Wissensmanagement als Basis für wissensbasiertes Management

Einführung in die technischen Grundlagen von Wissensbasierten Systemen und die Möglichkeiten der Wissensrepräsentation

Jörg Freistätter

Einleitung

Die vorliegende Arbeit soll einen Weg zur Einführung eines, durch den Einsatz von Informationstechnologie zu unterstützenden, Wissensmanagementsystems aufzeigen. Da Wissen, das in jeder Organisation, so auch in unserer, vorhanden ist, ständig wächst und damit ein nicht unwesentlicher Faktor für die Effizienz einer Organisation ist, wird es immer wichtiger, dieses zu sammeln, zu strukturieren und letztendlich wieder der Organisation nutzbar zu machen. Durch den Einsatz von modernen Technologien und der stetigen Steigerung der Verarbeitungsleistung und –geschwindigkeit wird es möglich, komplexe Strukturen für ein effizientes Wissensmanagement zu schaffen, das es ermöglicht, dieses Wissen auch entsprechend zu verwerten. Die ersten Ansätze sind mit der Einführung des Intranets und des KIS-ELAK bereits passiert. Wer nun aber darauf vertraut, dass solche Systeme das Allheilmittel darstellen, wird spätestens bei der ersten Nutzung der Medien eines Besseren belehrt.

Das Hauptproblem im Bereich der Wissensverarbeitung ist die Gratwanderung zwischen einer effizienten Struktur zur Wissensrepräsentation und einer effizienten Suche, die in einer entsprechenden Zeit ein korrektes Suchergebnis liefert. Die derzeit verwendete Volltextsuche ist für kleine Datenmangen durchaus geeignet, stößt aber bei verteilten Systemen und der ständig wachsenden Informationsflut bald an ihre Grenzen. Aus diesem Grund verwenden große Informationssammlungen wie zum Beispiel die Zentral-Dokumentation des BMLV (ZentDok) spezielle Verfahren zur Organisation der Information. Hierbei werden die Dokumente nach bestimmten Kriterien beschlagwortet. Dies ermöglich der verwendeten Suchmaschine (Excalibur) eine Suche mit einem entsprechenden Aufwand (siehe Teil 2: Suchverfahren). Diese Beschlagwortung stellt jedoch den größten Aufwand bei der Implementierung eines solchen Systems dar, ist aber der Ausschlag gebende Faktor für die Qualität und damit die Nutzbarkeit der Sammlung. Die Generierung von Metadaten in Wissenssammlungen ist aber derzeit noch unumgänglich. Für semantische Netze wie das

Internet werden daher jetzt standardisierte Verfahren für strukturierte Beschreibung der vorliegenden Dokumente entwickelt. Diese Beschreibungen werden dann im Dokument selbst abgelegt. In dieser Arbeit wird auf einen neuen, derzeit in Entwicklung befindlichen, Standard eingegangen, das Resource Description Framework. Die Verwendung von XML ermöglicht dabei die Austauschbarkeit und maschinelle Verarbeitung der Daten. Die Ressourcen (Dokumente, Begriffe), Statements und Properties werden mittels URIs abgebildet. Damit ist es für Suchmaschinen möglich, in verteilten Systemen wie im Intranet möglichst effizient zu suchen. Für eine Organisation wie das Bundesheer müssten hierbei, aufgrund der spezifischen Terminologie, eigene Definitionen standardisiert, die Intranetsites auf XML umgestellt und die Anwender bzw. Intranetverantwortlichen entsprechend geschult werden.

Wissensmanagement mittels wissensbasierten Systemen geht jedoch weit über die reine Informationssammlung und Organisation hinaus. Das Ziel hierbei ist es, das in einer Organisation vorhandene, Wissen so zu strukturieren und nutzbar zu machen, dass es möglich ist, aus dem gesammelten Wissen durch Inferenz, also entsprechende Algorithmen zur Ableitung, neues Wissen zu generieren, bzw. auf eine bestimmte Fragestellung eine Antwort zu bekommen.

In der Folge werden nun die Grundlagen für diese Systeme sowie die Ansätze der AI vorgestellt. Die Idee dahinter ist es, Computer so zu programmieren, dass sie aus dem fundierten Wissensstand der in einer Wissensrepräsentation dargestellt ist, auf „intelligente“ Weise Schlüsse und Ableitungen treffen bzw. finden, d.h. menschliche Intelligenz durch geeignete Computerprogramme zu simulieren.

Struktur der Arbeit

Im ersten Teil dieser Arbeit wird kurz auf die Grundlagen der Definition und Identifikation von Wissen eingegangen, Wissensmanagement dargestellt und vor allem die Unterstützung der Informationstechnologie aufgezeigt. Der zweite Teil der Arbeit

beschäftigt sich mit den Ansätzen der Künstlichen Intelligenz (KI; Artifical Intelligence, AI) und den Methoden, die in Wissensbasierten Systemen (WBS) Verwendung finden. Der dritte Teil beschreibt einen Standard (Ressource Description Framework, RDF) des „World Wide Web Consortium“ (W3C), der ein Konzept der systematischen Beschreibung von Dokumenten für eine effiziente Suche im Internet bzw. Intranet implementierte. Dieser Standard beruht auf dem Prinzip eines semantischen Netzes und ist deshalb so interessant, weil auch in unserer Organisation eine umfassende Wissenssammlung im 3.VE-Intranet bereits verfügbar ist, eine effiziente Suche in dieser Information jedoch bis dato noch nicht implementiert wurde. Der vierte und letzte Teil beschäftigt sich mit einem Ausblick auf die Zukunft des Wissensmanagements.

Wissen und Wissensmanagement

Was versteht man unter Wissen?

Es gibt unzählige Definitionen von Wissen. Viele sind der Meinung, Wissen sei nur die Sammlung von Informationen und Daten. Wissen ist jedoch viel mehr. Man könnte Wissen als die Summe aller Informationen und Daten auf einer höheren Aggregatstufe bezeichnen. Diese Informationen und Daten werden für die Entscheidungsfindung, in Abhängigkeit vom jeweiligen Kontext, von den Mitarbeitern vernetzt und unterliegen dabei einem Lebenszyklus.
Im Folgenden sind einige Definitionen von Wissen angeführt:

> *„In einem Prozess wählt der Mitarbeiter Informationen aus, die er vor seinem persönlichen Hintergrund bewertet, verbindet und transformiert, um ein Ziel zu erreichen.“ (Heroo)*

> *„Knowledge is organized information applicable to problem solving“(Woolf)*

"Knowledge is information that has been organized and analyzed to make it understandable and applicable to problem solving or decision making." (Turban)

Allein in diesen Definitionen von Wissen wird die Hauptproblematik der heutigen Informationsgesellschaft sichtbar. Wissen wird implizit dem Mitarbeiter zugeordnet. Das heißt, jeder Mitarbeiter setzt sein Wissen für seine Aufgabe und die Entscheidungsfindung in seinem Arbeitsbereich im Unternehmen ein. Damit ist das in der Organisation vorhandene Wissen das aggregierte Teilwissen aller Mitarbeiter im eingesetzten Umfeld. Wird ein Mitarbeiter von seinem Aufgabengebiet abberufen, sei es im Zuge eines Aufstiegs auf einen höheren Arbeitsplatz, einer Umschulung oder einer Reorganisation, nimmt er sein Wissen mit. Damit steht dieses für etwaige Nachfolger nur mehr beschränkt zur Verfügung (in Abhängigkeit von vorhandenen Aufzeichnungen). Dramatisch wird es jedoch, wenn besagter Mitarbeiter die Organisation komplett verlässt. In diesem Fall ist das Wissen für die Organisation verloren.

Aus diesem Grund kann Wissen auch als zusätzlicher Produktionsfaktor angesehen werden, der es durchaus wert ist, sich mit Maßnahmen für seine Institutionalisierung zu beschäftigen. Einen Ansatz dafür liefert das Schlagwort, das heute für die Wettbewerbsfähigkeit von Unternehmen als unabdingbare Voraussetzung gilt: Das Wissensmanagement.

Wissensmanagement (Knowlegde Management, KM)

Wissensmanagement ist mit einem Alter von ca. 10 Jahren eine relativ junge Wissenschaftsdisziplin. Der Begriff des KM wurde erstmalig von K.Wiig 1986 bei einer Konferenz geprägt. Das Wissensmanagement ist nicht allein auf den Einsatz von Informationstechnologie (IT) reduzierbar, sondern ein interdisziplinäres Fach, das sich auch mit den organisatorischen,

sozialen, psychologischen und informationstechnischen Aspekten und Faktoren auseinandersetzt.

> *„KM is the systematic, explicit, and deliberate building, renewal, and application of knowledge to maximize an enterprise's knowledge-related effectiveness and returns from knowledge assets. "*

Wissensmanagement beschäftigt sich also mit der zielorientierten Gestaltung des Wissens von Mitarbeitern. Voraussetzung zur Einführung eines Wissensmanagements ist eine Bewusstseinsänderung in der Unternehmenskultur. Wissen soll nicht mehr das „Privateigentum" des Mitarbeiters sein, sondern muss Teil der Organisation werden. Dazu muss das in der Organisation vorhandene Wissen vernetzt und geteilt werden. Durch die systematische Entwicklung von Kompetenznetzwerken soll sich die Organisation in eine lernende weiterentwickeln. Damit wird Wissen zu einem wichtigen Element der strategischen Unternehmensführung. Diese Entwicklung kann durch die Einführung von entsprechenden IT-Systemen wesentlich erleichtert und unterstützt werden.

Diese Evolution zur Steigerung der Innovationsfähigkeit ist der zentrale Schlüssel beim Übergang vom Informations- zum Wissensmanagement:

quantitativ, stark strukturierte Information	qualitative, schwach strukturierte Information
sammeln, zentrales Speichern, verteilen	strukturieren, verdichten, Indizieren, diffundieren
Aufbau von Datenbanken	Aufbau von vernetzten Wissensstrukturen
Bezug auf Vergangenheit	Bezug auf die Gegenwart und Zukunft
Bestandsfokus	Prozessfokus
„Haben-und-Halten"-Kultur	„Teilen/Mitteilen"-Kultur
Unterstützung des operativen Management	Unterstützung des strategischen Management
Steigerung der Effizienz	**Steigerung der Innovationsfähigkeit**

Der Nutzen, der dem Unternehmen aus der Einführung eines Wissensmanagements entsteht, lässt sich grob in folgende Teilaspekte einteilen:

- Strategischer Nutzen:
  - Wettbewerbsvorteil durch bessere Leistungen
  - Verbesserte Reaktionsfähigkeit auf neue Situationen durch Zeitvorsprung bzw. Wissensvorsprung.
- Operativer Nutzen
  - Zeit- und Kostenvorteile (z.B. bei Mitarbeiterfluktuation)
  - Verkürzung von Projektdauern
  - Verhinderung von Fehlern in Projekten („lessons learned“)
  - Beschleunigung von Prozessen.

IT-gestützte Wissensverarbeitung

Durch die ständige Evolution im Bereich der Informations- und Kommunikationstechnologie wird die Einführung eines IT-gestützten Systems für das Wissensmanagement bzw. die Verarbeitung von Wissen zusehends erleichtert. Das Erfassen, Speichern und Verteilen von Wissen soll dabei in der Organisation zu einem der Kernprozesse erklärt und dementsprechend behandelt und strukturiert werden. Einer ständigen Kontrolle der Effizienz kommt, wie bei jedem Geschäftsprozess, eine eminente Bedeutung zu. Abhängig von der vorhandenen IT Infrastruktur ist bei der Einführung einer der folgenden Ansätze zu wählen:

- **Integriertes Wissensmanagement:** Einsatz verschiedener unterschiedlicher Werkzeuge, die in ein System integriert werden müssen.
- **Generelles Wissensmanagement:** Einsatz eines globalen, vernetzten Systems.

Im Folgenden werden, durch die Darstellung der Grundlagen verschiedener Ansätze, die Möglichkeiten einer Unterstützung eines Wissensmanagementsystems durch den Einsatz von Methoden der

„Artificial Intelligence“ bzw. durch Implementierung eines Wissensbasierten Systems aufgezeigt.

Wissensbasierte Systeme

Was sind wissensbasierte Systeme?

Im allgemeinen Sprachgebrauch versteht man unter einem wissensbasierten System oder Expertensystem ein Computersystem, das Probleme lösen kann und für das spezielles Wissen (sog. Expertenwissen) erforderlich ist. Dieser Begriff ist jedoch zu weitläufig und unscharf, um ein wissensbasiertes System im eigentlichen Sinn von anderen Systemen abzugrenzen. Aufgrund dieser Definition wäre jedes Buchhaltungsprogramm, geschrieben in einer beliebigen Programmiersprache, wie z.B. C, Java oder Delphi, ein Expertensystem, da ja auch zur Fakturierung von Rechnungen bestimmtes Wissen erforderlich ist.

Eine Definition für ein wissensbasiertes System ist jedoch schwierig. Die Problematik dabei liegt bereits in der Definition von Wissen selbst. Nicht einmal hier gibt es eine zufriedenstellende allgemein akzeptierte Definition. In der Literatur selbst gibt es eine Vielzahl von Begriffbildungen zu wissensbasierten Systemen (WBS).

> *„The emphasis on the importance of knowledge in applications such as these prompts us to use the phrase knowledge-based systems to describe programs that reason over extensive knowledge bases.“ (NIL88)*
>
> *“AI programs that achieve expert level competence in solving problems by bringing to bear a body of knowledge are called knowledge-based systems. Often, the term expert system is reserved for programs whose knowledge base contains the knowledge used by human experts in contrast to knowledge gathered from textbooks or non-experts. More often than not,*

the two terms expert system and knowledge based systems are used synonymously." (NIL88)

Aus diesen Zitaten kann man entnehmen, dass wissensbasierte Systeme über eine Wissensbasis operieren. Die Wissensbasis kann als eine Art Datenstruktur angesehen werden, die die Zustände einer „Welt" in deklarativer Form beschreibt. Diese deklarative Beschreibung erfolgt dabei in einer geeigneten Sprache. Eines der charakteristischen Merkmale eines wissensbasierten Systems ist jedoch die Trennung von Problemwissen und Wissensverarbeitung. In der klassischen Datenverarbeitung besteht die Eingabe zur Lösung eines Problems aus Daten, die elementare Fakten darstellen. Zusammenhänge zwischen diesen werden dabei nicht explizit erfasst, sondern implizit im Algorithmus zur Problemlösung verwendet. Zur Lösung bestimmter Aufgaben ist diese Form der Organisation jedoch unzureichend.(FAH79)

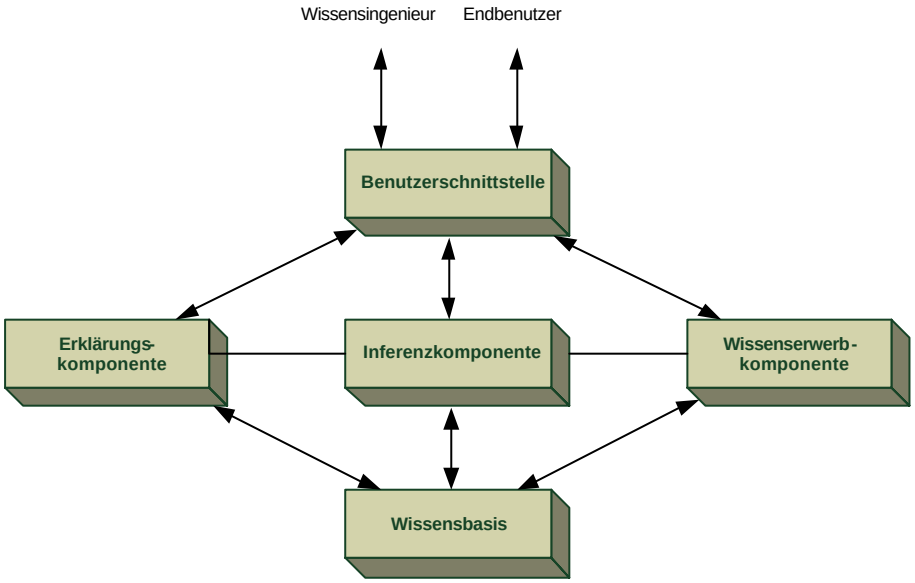

Abb.1: Architektur eines wissensbasierten Systems

In einer Wissensbasis werden über die Fakten hinaus auch Zusammenhänge von Fakten in expliziter Form dargestellt. Dieses Wissen kann dazu verwendet werden, um aus dem bestehenden Wissen durch Wissensverarbeitung (Inferenz) neues Wissen zu gewinnen. Die Zusammenhänge zwischen den Fakten werden zumeist in Regeln beschrieben. Vorteile dieser Organisation sind:

- **Flexibilität:** Verschiedenste Aufgaben können gelöst werden.
- **Änderbarkeit und Erweiterbarkeit:** Änderungen des Wissen können leichter in das System eingebracht werden.
- **Transparenz:** Das Wissen liegt „offen".

Ansätze in der Wissensverarbeitung

Die Verarbeitung des, in einer Wissensbasis gespeicherten, Wissens soll auf eine Art und Weise erfolgen, die als „intelligentes" Kombinieren und Verknüpfen von Fakten und Regeln angesehen werden kann. Diesbezüglich werden in der Artificial Intelligence (AI) zwei unterschiedliche Ansätze verfolgt [GIN93]:

- **Kognitive AI:** Die Wissensverarbeitung erfolgt nach einem Muster von Intelligenz. Dies setzt eine Theorie der Intelligenz voraus.
- **Rationale AI:** Diese Richtung vertritt eine ergebnisorientierte Wissensverarbeitung. Es wird keine direkte Entsprechung zu einem Denkmuster postuliert, die Verarbeitung muss jedoch nachvollziehbar sein.

Für wissensbasierte Systeme ist der Ansatz der rationalen AI maßgebend. Für die Repräsentation des Wissens und die Wissensverarbeitung muss dabei ein formales und automatisierbares Modell vorhanden sein. Für diese Zwecke bieten sich die Formalismen der Logik an. Fakten und Regeln lassen sich als Sätze in der Sprache der Logik darstellen, und die Verarbeitung des Wissens erfolgt durch Ableitung, d.h. durch Inferenzsysteme, die das logische Schließen modellieren. Problematisch dabei ist, dass nicht immer alles aus der Wissensbasis ableitbar ist, da auch diese nie vollständig sein kann, bzw. die Wissensbasis so komplex wird, dass keine

Inferenzprozedur in vertretbarer Zeit zu einem Ergebnis kommen kann.(GOT90)

Im weiteren Verlauf dieser Arbeit werden Formalismen zur Wissensrepräsentation und der Verarbeitung beschrieben, welche die Grundlage zum Bau eines wissensbasierten Systems bilden.

Ansätze in der Artificial Intelligence

Im Bereich der AI gibt es grundsätzlich zwei prinzipielle Sichtweisen, wie durch das Verarbeiten von Informationen ein intelligentes Verhalten entstehen kann. Nach diesen Sichtweisen kann man symbolverarbeitende Systeme (symbol processing systems) und subsymbolische Systeme (subsymbolic systems) unterscheiden.

Symbolverarbeitende Systeme

Bei diesen Systemen werden Informationen und Wissen in symbolischer Form repräsentiert. Dass durch die Verarbeitung in solchen Systemen eine hinreichende Basis für den Anspruch an ein intelligentes System gegeben sei, stützt sich auf die „Physical Symbol System Hypothesis“ von Newell und Simon. „A physical symbol system has the necessary and sufficient means for generating intelligent action“ *(New76)*

Symbolverarbeitende Systeme folgen der Tradition der klassischen Logik und den damit verbundenen Formalismen. Das Wissen wird hierbei in deklarativer Form repräsentiert, die Verarbeitung erfolgt auf Basis von Interferenzprozessen auf Symbolebene. Dabei geht das Wissen und Verhalten vom Ganzen zu den Teilen beziehungsweise wird von oben nach unten zerlegt. Aus diesem Grund wird dieser Ansatz oft auch als „Top-Down-Ansatz“ bezeichnet.

Subsymbolische Systeme

Bei den subsymbolischen Systemen sind Signale das wesentliche Element für die Informationsverarbeitung. Als Grundlage dazu dient die von R. Brooks (1990) formulierte Antithese zur „Physical Symbol Hypothesis".

„Physical Grounding Hypothesis"

> *„To build a system that is intelligent, it is necessary to have its representations grounded in the physical world. Our experience with this approach is that once this commitment is made, the need for traditional symbolic representations soon fades away entirely. The key observation is that the world is its own best model."*

Diese Hypothese verlangt von einem System, dass dessen Sensoren und Aktoren mit der realen Welt verbunden sind. Dateneingaben über Tastaturen werden dabei konsequent abgelehnt. Komplexes Verhalten ergibt sich daher aus der Interaktion von individuellen Teilsystemen mit der Umwelt sowie untereinander. Intelligentes Verhalten wird so zum kollektiven Phänomen, das mehr als die Summe seiner Teile ist.

Der Ansatz von Symbol verarbeitenden Systemen wird in der Wissensverarbeitung als „Bottom-up" Ansatz bezeichnet, der auch in engem Zusammenhang mit einer evolutionären Theorie der Intelligenz steht. Prototypische Vertreter von subsymbolischen Systemen sind Neuronale Netzwerke, auf die in dieser Arbeit in weiterer Folge noch eingegangen werden wird.

Problemlösung und Suchen

Die Suche stellt eine der wesentlichen Methoden in der Informatik und im Speziellen in der AI dar. Im Folgenden sollen die wichtigsten Suchverfahren vorgestellt werden. Dabei wird prinzipiell zwischen zwei Verfahrensgruppen unterschieden, den uninformierten und den heuristischen Verfahren. Letztere zeichnen sich durch die

Verwendung von problembezogenem Wissen bei der Lösungsfindung aus.

Die einfachste Form eines Suchproblems ist die Pfadsuche in einem gegebenen Graphen. Dabei wird ein Weg zwischen einem Ausgangspunkt und einem Endknoten gesucht, der gewisse Kriterien und Bedingungen erfüllt.

Beispiel:
Gegeben sei ein Suchraum (Graph) mit Knoten und Kanten. Die Kanten dürfen nur in Pfeilrichtung „befahren“ werden. Existiert ein Pfad von A nach G, der diese Richtungsbindung berücksichtigt?

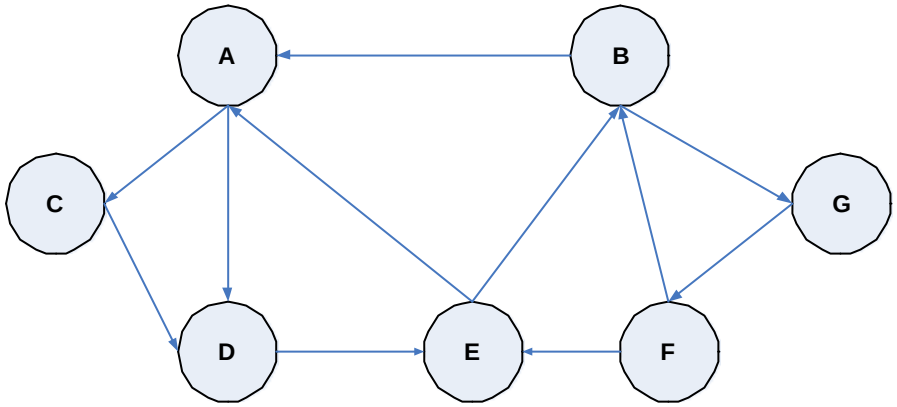

Abb.2: Suchraumrepräsentation durch einen gerichteten Graph

Offensichtlich existiert ein solcher Pfad, da der Pfad A, D, E, B, G den Startknoten A mit dem Zielknoten G verbindet und die Pfeilrichtung berücksichtigt.
Im angeführten Beispiel wurden bereits implizit wesentliche Elemente der Repräsentation verwendet:

1. Eine Datenstruktur als Grundlage (Suchraum)
2. ein Startknoten
3. ein (mehrere) Zielknoten, die vorgegebene Endbedingungen erfüllen.

Der Suchraum wird normalerweise nicht explizit vorgegeben, sondern wird während der Lösungsfindung dynamisch generiert. Diese Vorgehensweise wird deshalb notwendig, da Suchräume im Allgemeinen sehr groß sind und mit der Größe der Probleminstanz exponentiell anwachsen.

Ein weiteres Beispiel wäre eine Landkarte. Dabei repräsentieren die Städte die Knoten, die Kanten werden durch die Straßenverbindungen dargestellt. Hierbei kommt es nun auf die Bewertung der Kanten an. Repräsentiert wird diese durch die Angabe der Straßenkilometer. Eine mögliche Aufgabe könnte es sein, den kürzesten Weg zwischen dem Startknoten (Stadt A) und dem Zielknoten (Stadt Z) zu finden. Diese Problemstellung findet sich in jedem Navigationssystem.

Definition

Ein Suchproblem ist charakterisiert durch:

1. einen Startzustand (initial state)
2. eine nichtleere Menge von Zielzuständen (goal states), die alle den Zieltest (goal test) bestehen.
3. eine nichtleere Menge von Operatoren (Regeln, Aktionen), wobei durch Anwendung von Operatoren auf einen gegebenen Zustand Nachfolgezustände generiert werden. Die dabei entstehenden Kanten haben positive Kosten > 0 (Sind keine Kosten angegeben, werden die Einheitskosten (1) angenommen.)
4. einer Kostenfunktion für Pfade, die angibt, wie sich die Kosten eines Pfades aus den Kosten der Operatoranwendung ergeben.

In den folgenden Unterabschnitten werden nun verschiedene, wichtige Suchverfahren vorgestellt. Dabei soll untersucht werden, wie gut die einzelnen Verfahren folgende wichtige Eigenschaften erfüllen:

- **Vollständigkeit**: Wird durch das Suchverfahren eine Lösung gefunden, sofern diese existiert?

- **Optimalität:** Findet das Verfahren die optimale Lösung (kürzester Weg, geringste Kosten, etc)?
- **Zeitkomplexität:** Wie verhält sich der Zeitbedarf bei zunehmender Größe der Probleminstanz?
- **Speicherkomplexität:** Wie wächst der Speicherbedarf bei zunehmender Größe der Probleminstanz?

Uninformierte Suche

Ein Suchverfahren heißt dann uninformiert, wenn es während der Lösungsfindung keine problemspezifischen Informationen ausnutzt, um gewisse Teile des Suchraums anderen vorzuziehen. Die wichtigsten Vertreter dieser Verfahrensgruppe sind:

- Breitensuche (breath-first search, bfs),
- Uniform Cost Search (ucs),
- Tiefensuche (depth-first search, dfs) und
- Tiefensuche mit wachsender Tiefenschranke (depth-first search with iterative deepening, dfid).

Breitensuche

Die Breitensuche zeichnet sich durch das Charakteristikum der ebenenweisen Expansion aus. Das heißt, dass bei der Suche ein Zustand der Tiefe d erst dann expandiert wird, wenn alle Zustände der Ebene d-1 expandiert sind. Demnach ist die Breitensuche eine systematische Strategie, da zuerst alle Pfade der Länge 1, dann alle Pfade der Länge 2 usw. betrachtet werden. Existiert eine Lösung, so wird diese durch die Breitensuche gefunden (vollständig). Existieren mehrere Lösungen, so werden zuerst diejenigen mit der geringeren Tiefe gefunden, dann die mit den größeren Tiefen. Die Breitensuche ist dann optimal, wenn die Pfadkostenfunktion eine streng monoton steigende Funktion der Tiefe ist. Das ist dann der Fall, wenn alle Operatoren gleiche, positive Kosten >0 besitzen und sich die Pfadkosten durch die Summe der Operatorkosten ergeben.

Für die Betrachtung der Zeit- und Speicherkomplexität gehen wir von der vereinfachten Betrachtung aus, dass jeder Zustand dieselbe Anzahl b von Nachfolgeknoten besitzt. Diese Anzahl b wird auch Verzweigungsfaktor (branching factor oder branching degree) genannt. Die Anzahl der Blätter in einem Suchbaum mit der Tiefe d und konstantem Verzweigungsfaktor b ist demnach b^d. Damit ergibt sich ein maximaler Speicherbedarf von:

$$1 + b + b^2 + ... + b^d = \sum_{i=0}^{d} b^i = \frac{b^{d+1} - 1}{b - 1} \approx b^d$$

Die Zeitkomplexität ist, durch den Bedarf von mindestens einem Berechnungsschritt je Beschreibung eines Speicherelementes, ebenfalls exponentiell.

Uniform Cost Search

Die „Uniform Cost Search“ ist eine Verallgemeinerung der Breitensuche. Während bei der Breitensuche, durch die Einheitlichkeit der Kosten, alle Zustände quasi gleichwertig sind und es damit irrelevant ist, welcher als erstes expandiert wird, werden bei der Uniform Cost Search bestimmte Zustände, die kleinere Kosten verursachen, bevorzugt expandiert. Das Prinzip lässt sich am Besten anhand eines Beispiels erklären: Betrachten wir das in Abbildung 3 dargestellte Routenproblem. Gesucht wird eine Verbindung von S nach Z, wobei minimale Kosten anfallen sollen.

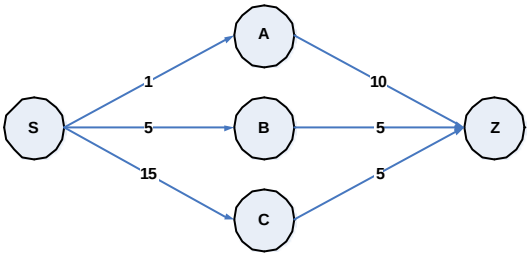

Abb.3: Suchraumrepräsentation durch einen bewerteten Graphen

Als erstes wird nun der Knoten S expandiert, da er mit 0 die geringsten Kosten aufweist. Das Resultat ist der Zustandsraum S, A, B, C.

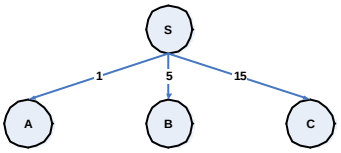

Abb.4: Erster Expansionsschritt

Im nächsten Schritt wird der Knoten A expandiert, da dieser mit 1 nun die geringsten Kosten besitzt. Man erzeugt damit den Knoten Z mit den Gesamtkosten von 11.

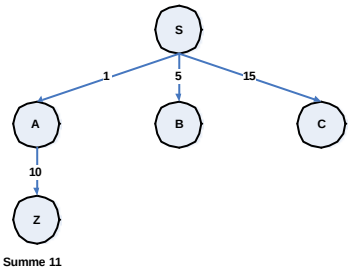

Abb.4: Zweiter Expansionsschritt

Bei einer Nichtberücksichtigung der Kosten hätte man bereits die erste Lösung gefunden. Diese wäre jedoch nicht minimal. Um Optimalität zu gewährleisten, müssen nun noch die restlichen Knoten expandiert werden, bis alle Pfadlängen Kosten ≥ 11 verursachen. Im vorliegenden Fall betrifft dies nur mehr den Knoten B, da C bereits Kosten von 15 verursacht. Diese letzte Expansion ergibt nun die optimale Lösung S-B-Z mit den zugehörigen Kosten von 10.

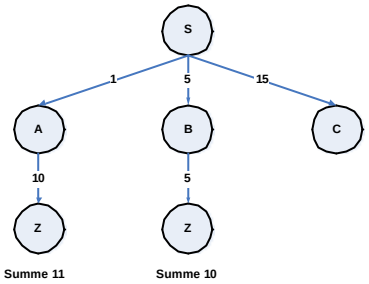

Abb.5: Dritter Expansionsschritt

Dabei ist leicht ersichtlich, dass die UCS sowohl vollständig als auch, unter den bei der Breitensuche bereits angesprochenen Voraussetzungen, optimal ist. Die Speicher- und Zeitkomplexität entspricht ebenfalls dem bei der Breitensuche Gesagten.

Tiefensuche

Im Gegensatz zur Breitensuche wird bei der Tiefensuche die Expansion nicht ebenenweise durchgeführt, sondern immer der Nachfolger bis zur Tiefe d expandiert. Erst wenn alle Zustände der Tiefe d keine Nachfolger mehr besitzen, werden die alternativen Zustände der Tiefe d-1 expandiert. So wird zuerst in die Tiefe vorgestoßen und erst danach mittels „backtracking“ Alternativen untersucht. Die Problematik bei diesem Verfahren liegt in der Gefahr, dass man in unendlich absteigende Pfade gelangt, die keine Lösung besitzen und der Suchvorgang nicht terminiert wird. Damit werden keine Lösungen gefunden, auch wenn dieselben existieren und auf einem anderen Pfad liegen. Die Tiefensuche ist demnach weder

vollständig noch optimal. Der große Vorteil der Tiefensuche ist jedoch der relativ geringe Speicherbedarf, da lediglich die Zustände des verfolgten Pfads und deren alternative Zustände gespeichert werden müssen. Die Zeitkomplexität bleibt jedoch weiterhin exponentiell.

Tiefensuche mit wachsender Tiefenschranke

Eine einfache Möglichkeit, die Verfolgung eines unendlichen Pfads zu verhindern ist die Einführung einer Tiefenschranke. Dabei gibt es grundsätzlich zwei Möglichkeiten. Eine globale, vom Benutzer vorgegebene Tiefenschranke, die aber das Problem in sich birgt, eventuell keine Lösung zu finden, wenn eine solche in einer größeren Tiefe vorhanden ist. Sinnvoller ist eine dynamisch erzeugte Tiefenschranke, die bei einer vorgegebenen Konstante beginnt und nach einem negativen Suchresultat schrittweise erhöht wird. Damit ist es möglich, die Vorteile einer Tiefensuche mit den Eigenschaften einer Breitensuche zu kombinieren. Das Verfahren ist sowohl vollständig als auch optimal und benötigt nur den Speicherbedarf einer Tiefensuche (maximale Pfadlänge m mal Verzweigungsgrad b).

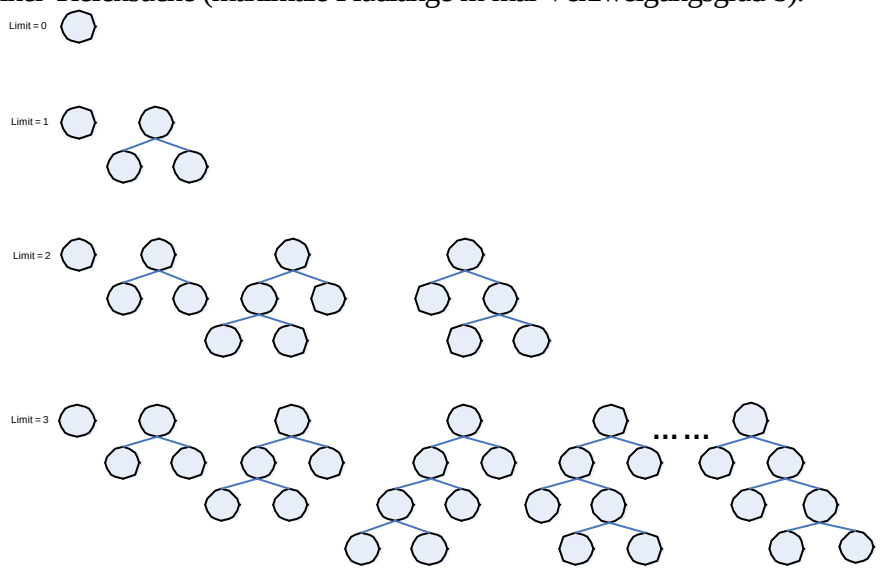

Abb.6: Tiefensuche mit wachsender Tiefenschranke

Heuristische Suche

Die im vorigen Kapitel besprochenen Suchverfahren nutzen keinerlei problemspezifisches Wissen, um Präferenzen beim Expandieren von speziellen Knoten zu treffen. Die Suchverfahren, die in weiterer Folge vorgestellt werden, verfügen über Algorithmen, so genannte Heuristiken, die als eine Art Daumenregel angesehen werden können. Diese dienen als Schätzregeln, mit deren Hilfe „Kosten“ geschätzt und auf dieser Basis bestimmte Knoten beim Expandieren bevorzugt werden. Dabei ist eine Heuristik eine spezielle Form einer Evaluations-Funktion. Mit ihr wird versucht, ein mathematisches Abbild der menschlichen Intuition zu schaffen. Um eine gute Heuristik zu einem Problem zu finden, bedarf es vieler Phantasie und Erfindungsgabe. Zudem lässt sich eine Heuristik fast nur durch Ausprobieren klassifizieren. Die Leistung der Verfahren ist sehr stark von der gewählten Heuristik abhängig. Heuristiken werden als „zulässig“ bezeichnet, wenn sie „optimistisch“ sind. Das bedeutet, dass eine zulässige Heuristik die Kosten unterschätzen muss.

Die Verfahren die hier vorgestellt werden, werden in der Literatur oft unter dem Begriff „best-first search“ zusammengefasst. (KAI89)

Greedy Search

Bei diesem Algorithmus wird der Knoten zuerst expandiert, der aufgrund einer Heuristik dem Ziel am nächsten ist. Charakteristisch für die „Greedy Search“ ist das Schätzen der „Kosten“ vom aktuellen Knoten zum Zielknoten ohne die Berücksichtigung der bisher angefallenen „Kosten“. Im angegebenen Beispiel (siehe Abbildung 7) soll eine Reise von ARAD nach BUKAREST geplant werden. Ziel ist dabei ein Weg, der die kürzeste Fahrtstrecke (km) aufweist. Als Heuristik (Schätzfunktion) wird dabei die nebenstehende Tabelle verwendet, welche die direkten Entfernungen (Luftlinie) zwischen einer Stadt und Bukarest angibt.

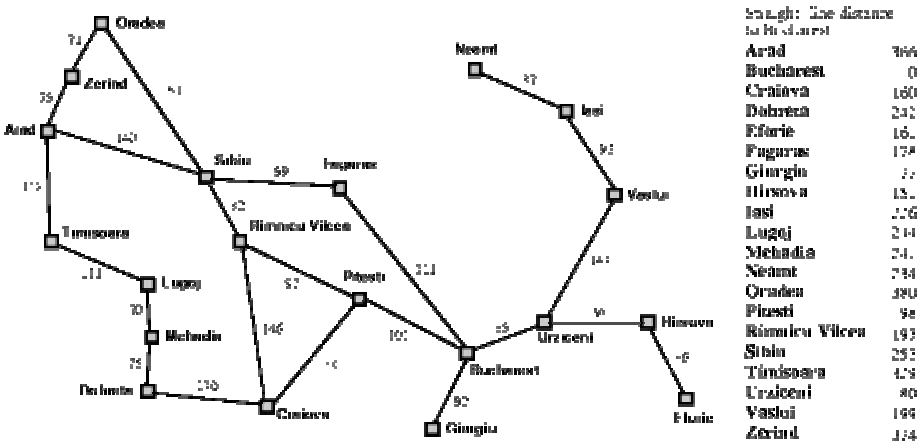

Abb.7: Suchraumrepräsentation durch einen gewichteten Graphen und direkte Entfernungen als Heuristik

Als erster Knoten wird der Startknoten ARAD expandiert. Im Anschluss werden aufgrund der Heuristik die möglichen Nachfolger bewertet und derjenige mit der geringsten Luftlinienentfernung nach BUKAREST expandiert. Im vorliegenden Fall ist dies SIBIU. Nach Abschluss des Verfahrens liegt dann als mögliche Lösung der Pfad Arad-Sibiu-Fagaras-Bukarest mit einer Pfadlänge von 450 km vor.

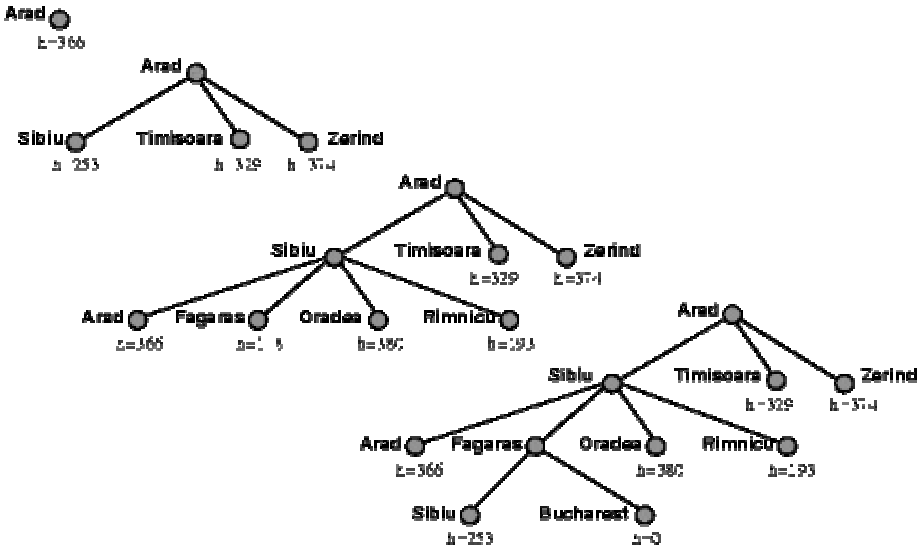

Abb.8: Expansion des Suchraums

Dabei wird ein Problem dieses Verfahrens sichtbar. Die gefundene Lösung ist nicht optimal. Der Weg Arad-Sibiu-Rimnicu-Vilcea-Pitesti-Bukarest ist mit 418 km um 32 km kürzer.

Neben der fehlenden Optimalität gibt es bei diesem Verfahren eine zweite Schwachstelle. Dadurch, dass bereits angefallene Kosten nicht berücksichtigt werden, ist es möglich, dass es zu Zyklen kommt und das Verfahren nicht terminiert wird. In seinem Verhalten erinnert der Algorithmus sehr an die Tiefensuche, da auch hier erst in die Tiefe gesucht wird, und nur wenn sich der Weg als Sackgasse erweist, wird an höherer Stelle nach Alternativen gesucht.
Die Greedy Search ist demnach weder vollständig noch optimal. Die Speicherkomplexität liegt im Bereich von bm, die Zeitkomplexität im Bereich von b^m.

A*-Search

Bei A* wird als Evaluations-Funktion die Summe aus den Kosten bis zum aktuellen Knoten und einer Heuristik genommen.

$$f(n) = g(n) + h(n)$$

Wobei f die heuristische Schätzfunktion ist, h(n) die minimalen Kosten der Schätzfunktion vom Knoten n zu einem Zielzustand und g(n) die bisher angefallenen Kosten repräsentiert.
Da man die Heuristik weitestgehend auch mit einer Abschätzung der Kosten bis zum Ziel identifizieren kann, ist die hier benutzte Evaluations-Funktion gleich den geschätzten Kosten der Lösung, die auf diesem Weg zu finden wäre. Wichtig bei der Heuristik ist hier, dass sie eine so genannte „optimistische Heuristik“ ist. Das bedeutet, dass ihre Abschätzung stets besser ist als der wirkliche Wert. Auf diese Weise kann sichergestellt werden, dass entlang jedes Weges im Graphen die Kosten stets steigen.
Anhand der Eigenschaften, die die A^*-Suche definieren

- Der Zustandraum besitzt einen endlichen Verzweigungsgrad,
- Die „Kosten“ jeder Kante sind positiv und >0 und

- f(n) = g(n) + h(n) wobei h(n) zulässig sein muss

findet die A*-Suche im oben angeführten Beispiel die optimale Lösung, da sie im Gegensatz zur Greedy Search im dritten Schritt den Knoten mit der Bezeichnung Rimnicu-Vilcea expandiert.

Setzt man die Heuristik gleich null, so erhält man die Uniform-Cost-Search. Von dieser Verwandtschaft erbt die A*-Suche ihre Optimalität und Vollständigkeit. Die Komplexitäten sind bm für den Speicherbedarf, bzw. b^m für die Zeitkomplexität.

Zusammenfassung Suchalgorithmen

Die folgende Tabelle gibt einen Überblick über die Eigenschaften der bisher besprochenen Suchverfahren. Dabei kennzeichnet b wieder den konstanten Verzweigungsgrad, d die Tiefe der Lösung und m die maximale Pfadlänge im Suchraum, wobei eine unendliche Pfadlänge möglich ist (m=).

Algorithmus	Vollständig	Optimal	Speicherkomplexität*	Zeitkomplexität*
Breitensuche	Ja	Ja	$O(b^d)$	$O(b^d)$
Uniform-Cost-Search	Ja	Ja	$O(b^d)$	$O(b^d)$
Tiefensuche	Nein	Nein	O(b*m)	$O(b^m)$
Tiefensuche m. wachsender Tiefenschranke	Ja	Ja	O(b*d)	$O(b^d)$
Greedy Search	Nein	Nein	O(b*m)	$O(b^m)$
A*-Search	Ja	Ja	O(b*m)	$O(b^m)$

*worst case

Wissensrepräsentation

Der zentrale Aspekt in wissensbasierten Systemen ist die Repräsentation des vorliegenden Wissens. Dazu gibt es verschiedene Ansätze, die im Folgenden kurz erläutert werden sollen.

Anforderungen an die Wissensrepräsentation

Um die Ansätze zur Wissensrepräsentation verstehen zu können, ist es erforderlich, im Vorfeld Überlegungen anzustellen, welche Eigenschaften ein Formalismus zur Wissensrepräsentation erfüllen sollte. Im Folgenden werden einige wünschenswerte Eigenschaften vorgestellt. In der Regel werden diese jedoch nie vollständig durch einen bestimmten Formalismus abgedeckt werden können.

Ausdruckstärke

Der Formalismus muss hinreichend mächtig sein, damit das Wissen überhaupt dargestellt werden kann. Dazu ist es erforderlich, dass neben der Darstellung des reinen Faktenwissens auch die Zusammenhänge der Fakten in Form von Regeln dargestellt werden können.

Verarbeitbarkeit

Das gespeicherte Wissen muss verarbeitet werden. Im Konkreten bedeutet dies, dass aus vorhandenem Wissen auf systematische Weise neues Wissen generiert werden können muss. In der AI werden zu diesem Zweck oft logik-basierte Schlussfolgerungssysteme wie Kalküle der formalen Logik, speziell entwickelte Schlussfolgerungsverfahren wie analogie-basiertes Schließen oder abduktives Schließen verwendet. Die Verarbeitbarkeit soll dabei automatisierbar sein, Schlüsse sollen immer korrekt und vor allem in endlicher Zeit gezogen werden. Eine Verarbeitbarkeit in polynomieller Zeit wird dabei angestrebt. Dies steht normalerweise im Gegensatz zu großer Ausdrucksstärke. Ein weiteres Problem ist dabei, dass selbst für sehr eingeschränkte Formalismen heutzutage

nur Schlussverfahren mit exponentiellem Aufwand bekannt sind. Dies erfordert den Einsatz von unvollständigen Verfahren und Heuristiken.

Flexibilität

Der Ansatz zur Wissensrepräsentation soll allgemeiner Natur sein, dass heißt, die Wissensbasis soll Wissen aus den verschiedensten Domänen darstellen können. Darüber hinaus sollen spezifische Informationen, die zur Lösung unterschiedlicher Aufgabenstellungen erforderlich sind, auch effizient dargestellt werden können.

Modularität

Die Wissensbasis soll modular aufgebaut sein, da sie in der Praxis Änderungen unterliegt. Die Veränderung soll daher leicht durchführbar sein. Solche Veränderungen sind zum Beispiel das Einfügen bzw. Entfernen von bestimmten Wissensinhalten oder Informationen. Die modulare Bauweise unterstützt dabei das Hinzufügen von Wissen zur Lösung bestimmter Teilprobleme ohne die Beeinflussung des Rests der Wissensbasis. Ansätze deklarativer Wissensrepräsentation unterstützen diese Modularität in der Regel besser als prozedurale Methoden, da sich die Wissensbasis durch die explizite Bekanntgabe von Zusammenhängen zwischen Wissenselementen der Wissensbasis leichter in zusammengehörige Bereiche strukturieren lässt.

Verständlichkeit

Das Wissen muss in verständlicher Form abgelegt werden können, so dass der Wissensingenieur, der mit der Erstellung und der Wartung der Wissensbasis beauftragt ist, sowie auch alle Benutzer das dargestellte Wissen leicht erfassen können. Diese Forderung setzt jedoch nicht dieselbe Darstellungsform voraus. Der Benutzer und der Wissensingenieur werden demnach unterschiedliche Schnittstellen (User-Interfaces) haben.

Unvollständige Information

Erfassung unsicheren Wissens

In vielen Fällen kann die Wirklichkeit nicht durch Fakten beschrieben werden, die hundertprozentig korrekt sind, sondern nur mit Fakten, die mit einer bestimmten Wahrscheinlichkeit zutreffen. Intelligentes Verhalten besteht darin, aus einer Situation das Beste zu machen, in der das richtige Verhalten nicht eindeutig vorgeschrieben ist. Daher müssen intelligente Systeme in der Lage sein, mehrdeutige Aussagen zu verarbeiten. Diese Unsicherheiten können auf mehrere Arten entstehen:

- Inhärente Unsicherheit der Information
- Unvollständigkeit der Information
- Unsicherheit von Schlussfolgerungen
- Zusammenfassung von Informationen, die aus mehreren, eventuell einander widersprechenden Quellen stammen

Bei vielen Anwendungen wird jedoch zur Vereinfachung der Verarbeitung auf die Repräsentation von unsicherem Wissen verzichtet.

Prozedurale Methoden

Beim Ansatz der Prozeduralen Wissensrepräsentation wird das Wissen in Form von Prozeduren dargestellt. Die Inferenzkomponente ist sich dabei jedoch des in der Prozedur gespeicherten Wissens bewusst, der Aufruf der Prozedur unterliegt ihrer expliziten Kontrolle. Die Inferenzkomponente ist mit dem Wissen über Problemlösungsmöglichkeiten ausgestattet, dass heißt sie weiß exakt, welche Prozeduren es gibt und welche Probleme diese beantworten können. Sie verwendet dieses Wissen, um für ein ad hoc gestelltes Problem, eine geeignete Lösungsprozedur zu bestimmen.

Beispiel:

```
function bird(x): boolean;
    if x='Tweety' then return true
    else if x='Sam' then return true
    else if penguin(x) then return true
    else return false
```

Im angeführten Beispiel wird streng nach einer einfachen Entscheidungsliste vorgegangen, die Schritt für Schritt geprüft wird und letztendlich eine Lösung findet. Dies entspricht der logischen Repräsentation:

$$\forall x(bird(x) \leftrightarrow (x = Tweety \vee x = Sam \vee penguin(x)))$$

Zur Feststellung der Eigenschaft penguin(x) zieht die Inferenzkomponente eine eigene Prozedur heran.

Logikbasierende Methoden

Bereits in den 50er Jahren ist von J. McCarthy formale Logik als Sprache für die Entwicklung von AI-Systemen vorgeschlagen worden. Unter formaler Logik soll hierbei die Thematik verstanden werden, aus einer Formulierung bestimmter Sachverhalte durch Aussagen bzw. Sätze in einem logischen System auf das Zutreffen weiterer Sachverhalte, die inhärent wahr oder falsch sind, systematisch zu schließen. Auf diese Weise wird versucht, den Vorgang des „logischen“ Schließens auf adäquate Weise zu beschreiben und zu untersuchen. Dabei sind folgende Komponenten auseinander zuhalten:

- **Syntax**: Die zu untersuchenden Sätze müssen in einer Sprache sein, die festlegt, welche Ausdrücke (Formeln) zulässige Sätze sind.
- **Semantik**: Die Semantik legt die Bedeutung der Sätze fest, d.h. was letztlich durch einen Satz bzw. eine Menge von Sätzen ausgedrückt wird. Dies erfolgt üblicherweise durch Konzepte wie Interpretationen und Modelle.

Aussagenlogik

Die Aussagenlogik ist die elementare Form der Logik und tritt als Subformalismus der Prädikatenlogik auf. Sie ist in der Praxis deshalb von Bedeutung, da viele Problemstellungen auch in ausdrucksstarken Formalismen wie der Prädikatenlogik unter bestimmten Voraussetzungen auf die Aussagenlogik zurückgeführt werden können. Gegenstand der Aussagenlogik sind in sich geschlossene Aussagen, die mit wahr oder falsch bewertet werden können, und deren logische Beziehung zueinander.

Beispiel:

A1: Das Lfz Saab S35 OE verfügt über ein Triebwerk.
A2: Ein Triebwerk verbraucht Kerosin (Jet A1).
A3: Ein Triebwerk ist laut.
A4: Ein Triebwerk läuft.

Durch die logische Verknüpfung entstehen komplexere Aussagen:

C1: Das Lfz Saab S35 OE verfügt über ein Triebwerk und ist laut.
C2: Wenn ein Triebwerk läuft, ist es laut.
C3: Das Triebwerk läuft nicht.
C4: Ein Triebwerk verbraucht Kerosin oder es läuft nicht.

Beschreibt man diese Zusammenhänge nun in logischen Formeln so erhält man:

C!: $A1 \wedge A3$ C2: $A4 \rightarrow A3$ C3: $\neg A4$ C4: $A2 \vee \neg A4$

Syntax

Ausgangspunkt zur Bildung von aussagenlogischen Formeln ist eine Menge A von elementaren Aussagen (Atome). Die Menge der Aussagen über A ist die kleinste Menge von Formeln *F*, die folgende Punkte erfüllen:

- Jedes Atom $a \in$ A ist in *F*.
- Wenn φ_1, φ_2 Formeln aus *F* sind, dann sind:

- $\varphi_1 \wedge \varphi_2$ (Konjunktion)
- $\varphi_1 \vee \varphi_2$ (Disjunktion)
- $\varphi_1 \rightarrow \varphi_2$ (Implikation)
- $\varphi_1 \quad \varphi_2$ (Äquivalenz)
- $\neg\varphi_2$ (Negation)

ebenfalls in *F*.

Semantik

Die Semantik von Formeln der Aussagenlogik wird durch Interpretationen und Modelle festgelegt. In der Abstraktion kann dabei jede atomare Aussage *a* den Wert wahr (t, true) oder falsch (f, false) annehmen.

Eine Interpretation ist eine Abbildung I: $A \rightarrow \{t,f\}$, die jedem Atom $a \in A$ einen Wahrheitswert $I(a)$ zuordnet. Die Menge aller Interpretationen wird mit *Int*(A) bezeichnet.

Eine Interpretation I ist ein Modell für eine Aussage φ (Notation: $I \models \varphi$), wenn $I(\varphi) = t$ gilt. I ist ein Modell für eine Menge von Aussagen S (geschrieben $I \models S$), wenn I ein Modell für jedes $\varphi \in S$ ist (d.h. $\forall \varphi \in S: I \models \varphi$).

In der Aussagenlogik und der Logik allgemein können auch Schlussfolgerungen durchgeführt werden. Folgende wichtige Inferenzmechanismen stehen zur Verfügung:

- Modus ponens
- Resolution

Der Modus ponens besagt: *Wenn eine Aussage A wahr ist und A eine weitere Aussage B impliziert (A ®B), dann ist auch B wahr.*

Eine Resolution ist eine einfach anzuwendende, syntaktische Umformungsregel. Voraussetzung dafür ist jedoch, dass die Formel in konjunktiver Normalform (KNF) vorliegt. Gegebenenfalls ist vor der Anwendung noch eine Transformation in die KNF durchzuführen.

Es ist offensichtlich durch einen Algorithmus entscheidbar, ob eine gegebene Formel φ der Aussagenlogik erfüllbar ist. Eine simple Vorgehensweise ist, systematisch alle Interpretationen I zu überprüfen, bis ein Modell von gefunden wird. Der Test $I \models \varphi$ ist

einfach und kann in linearer Zeit der Länge von φ durchgeführt werden. Im schlimmsten Fall müssen 2^n Interpretationen getestet werden, wobei n die Anzahl der Atome in A ist, d.h. dieser einfache Algorithmus hat eine exponentielle Laufzeit. Derzeit sind keine Algorithmen bekannt, die dieses Problem in polynomieller Zeit zu lösen vermögen. Die Existenz solcher Algorithmen ist äquivalent zum berühmten offenen P=NP Problem, da das Erfüllungsproblem ein NP-vollständiges ist (Theorem von Cook, 1971). Zur Lösung großer Instanzen dieses Problems müssen demnach heuristische Verfahren verwendet werden, um ein einigermaßen akzeptables Laufzeitverhalten zu bekommen, selbst auf die Gefahr hin, dass diese nicht immer irgendein bzw. ein richtiges Ergebnis liefern.

Prädikatenlogik

Die Prädikatenlogik ist ein Teilgebiet der Logik. Man kann sie als Erweiterung der Aussagenlogik ansehen, die zusätzlich zur Verknüpfung von Aussagen (z.B. durch und oder oder) auch die Eigenschaften von Objekten und des Geltungsbereiches betrachtet, wobei erstere durch Prädikatssymbole und Funktionssymbole, letztere durch Quantoren beschrieben werden. Aus elementaren Aussagen über Eigenschaften und Beziehungen von Objekten können durch logische Verknüpfungen komplexere Aussagen gebildet werden. Objekte bleiben dabei mitunter unbestimmt und werden durch einen Platzhalter (Variable) repräsentiert (z.B. soldat(x), offizier(x)). Über logische Quantoren kann die Betrachtung einer Aussage für bestimmte Belegungen der Variablen ausgedrückt werden (z.B. in $\forall x(\text{offizier}(x) \rightarrow \text{treu}(x))$).

Damit stellt die Prädikatenlogik wesentlich stärkere Ausdrucksmittel zur Verfügung. Die Grundlagen für eine formale Sprache der Prädikatenlogik (erster Ordnung) wurde von Ludwig Gottlob Frege 1879 in seiner „Begriffsschrift“ gelegt.

Wie jeder Logikkalkül besteht auch die Prädikatenlogik aus

- Angaben, wie man systematisch formal korrekte Aussagen konstruiert,
- einer Menge von Axiomen, von denen jedes einzelne Axiom ebenfalls eine formal korrekte Formel darstellt,
- einer Menge von Regeln, die erlauben, Sätze (Theoreme) aus früher hergeleiteten Sätzen oder den Axiomen herzuleiten.

Syntax

Ausgangspunkt zur Bildung von Formeln in der Prädikatenlogik ist ein Vokabular für Objekte, Eigenschaften und Beziehungen, sowie ein Satz von logischen Grundzeichen. Üblicherweise ist folgendes Vokabular vorhanden:

- **Konstantensymbole**: Mit Konstantensymbolen wird ein bestimmtes Objekt angesprochen.
- **Funktionssymbole:** Ein Funktionssymbol *f* der Stelligkeit n steht intuitiv für eine Funktion, die jeder Eingabe von Objekten $o_1,.....,o_n$ als Ausgabe genau ein Objekt $o=f(o_1,....,o_n)$ zuordnet.
- **Variablensymbole:** Ein Variablensymbol steht für ein Objekt, das zur Auswertung einer Formel festgelegt werden muss. Anders als bei einem Konstantensymbol kann hierauf auch ein Quantor angewandt werden.
- **Prädikatensymbole:** Ein Prädikatensymbol der Stelligkeit n steht für ein n-äres Prädikat. Bei n=1 ist dies eine Eigenschaft eines Objekts, bei n>1 eine Beziehung zwischen Objekten.
- **Terme:** Terme bezeichnen Objekte und sind induktiv wie folgt definiert:
    - Jedes Konstantensymbol c und jedes Variablensymbol x ist ein Term.
    - Wenn $t_1,.....,t_n$ bereits enthaltene Terme sind und *f* ein n-stelliges Funktionssymbol, dann ist $P(t_1,.....,t_n)$ ein Term.
- **Atomare Formeln:** Wenn *P* ein n-stelliges Prädiktensymbol ist, und $t_1,.....,t_n$ Terme sind, dann ist $P(t_1,.....,t_n)$ eine atomare Formel (Atom).

- **Formeln:** Jede atomare Formel ist eine Formel in F. Wenn φ_1, φ_2 Formeln in F[nur 1 A] sind, und x eine Variable ist, dann sind
  - $\varphi_1 \wedge \varphi_2$ (Konjunktion)
  - $\varphi_1 \vee \varphi_2$ (Disjunktion)
  - $\varphi_1 \rightarrow \varphi_2$ (Implikation)
  - $\varphi_1 \quad \varphi_2$ (Äquivalenz)
  - $\neg\varphi_1$ (Negation)
  - $\forall x \varphi_1$ (Allquantifizierung: Der ALL-Quantor sagt, dass für alle betrachteten Elemente oder Elementkombinationen eine (zusammengesetzte) Aussage zutrifft.)
  - $\exists x \varphi_1$ (Existentielle Quantifizierung: Der EXISTENZ-Quantor sagt, dass mindestens für ein Element der betrachteten Elemente oder Elementkombinationen eine (zusammengesetzte) Aussage zutrifft.)

ebenfalls Formeln in F. Die Formel φ_1 ist dabei der Bindungsbereich des Quantors $\forall x$ bzw. $\exists x$.

Semantik

Wie bei der Aussagenlogik wird die Semantik der Prädikatenlogik über Interpretationen und Modelle von Formeln definiert. Interpretationen sind hier allerdings wesentlich komplexere Strukturen. Formal hat eine Interpretation I für ein Vokabular V zwei Komponenten:

1. Einen beliebigen nichtleeren Wertebereich D von Objekten (domain, universe)
2. Zuordnungen von Objekten, Funktionen und Relationen über D zu allen Konstanten-, Funktions- und Prädikatensymbolen, d.h.:
   - Jedem Konstantensymbol c aus V wird ein Objekt $I(c)$ aus D zugeordnet.
   - Jedem n-stelligen Funktionssymbol aus V wird f eine Funktion $I(f):D^n \rightarrow D$ zugeordnet.

- Jedem n-stelligen Prädikatensymbol P aus V wird eine n-stellige Relation $I(P) \in D^n$ zugeordnet.

Häufig spricht man präziser von Prädikatenlogik erster Stufe (englisch: first-order predicate calculus oder first order logic, FOL). Diese zeichnet sich dadurch aus, dass Sätze des Typs „für jede Eigenschaft E, gilt folgendes...“ nicht behandelt werden. Trotz dieser Einschränkung lässt sich aber mit der Prädikatenlogik erster Stufe die ganze Mengentheorie formalisieren und damit gewissermaßen fast das ganze Gebiet der Mathematik. Die Prädikatenlogik ist die klassische Logik, der die Mathematik zugrunde liegt.

Objektorientierte Methoden

Bei der Konzeptualisierung, im Schritt also, in dem der abzubildende Ausschnitt der Welt festgelegt wird, werden die relevanten Objekte, deren Eigenschaften und die Beziehungen der Objekte untereinander ermittelt. Durch die Darstellung in den Formalismen der prozeduralen und logikbasierten Konzepten geht die syntaktische Struktur des Objekts selbst verloren und dieses Phänomen wird in der Wissensbasis quasi verstreut gespeichert.

Bei objektorientierten Methoden geht man von der Idee aus, dass die Struktur der Objekte bei der Wissensrepräsentation beibehalten werden sollte. Die ermittelten Objekte sollten dabei auf möglichst natürliche Weise durch formale Objekte dargestellt werden können. Charakteristische Merkmale von objektorientierten Methoden zur Wissensrepräsentation sind demnach:

- **Zentrale Beschreibung:** Die Objektstruktur wird zentral beschrieben, d.h. die Beschreibung ist über das Objekt lokalisierbar. Mit dem Objekt selbst sind die Eigenschaften (Attribute) des Objekts im direkten Zusammenhang abgelegt.
- **Deklarative Strukturbeschreibung:** Strukturen und deren Zusammenhänge werden in deklarativer Form erfasst.

- **Objektsstruktur:** Auf Ebene des konkreten Objekts werden die Eigenschaften eines Objekts explizit durch Attributwerte beschrieben.
- **Klassenstruktur:** Gleichartige Objekte sollen durch abstrahierende Beschreibungen erfasst werden, und die Ähnlichkeiten von Objekten sollten aus deren Beschreibung hervorgehen.

Weiters unterstützt die objektorientierte Wissensrepräsentation auch andere Konzepte, die aus den objektorientierten Programmiersprachen bekannt sind, wie zum Beispiel Vererbung und die damit in Zusammenhang stehende Überdeckung. Dabei kann es im Gegensatz zu Programmiersprachen bei Wissensrepräsentationen zu Inkonsistenzen (z.B. Nixon-Diamond) kommen, die mit geeigneten Mitteln behandelt werden müssen.

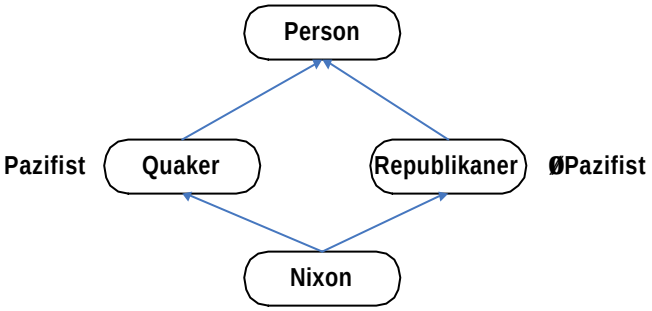

Abb.9: Der Nixon-Diamond

Durch die Vererbung entsteht hierbei ein Widerspruch. In einer logikbasierten Darstellung kann das Wissen durch die Regeln $\forall x(Q(x) \rightarrow P(x))$, $\forall x(R(x) \rightarrow \neg P(x))$ und die Fakten $Q(Nixon)$, $R(Nixon)$ beschrieben werden. Daraus lässt sich $P(Nixon)$ und $\neg P(Nixon)$ ableiten.

Framesysteme

Framesysteme basieren auf einem objektorientierten Ansatz zur Wissensrepräsentation, der in den 70er Jahren konzipiert worden ist, ungefähr zu der Zeit, als erste Ideen für eine objektorientierte Programmiersprache wie Smalltalk verfolgt worden sind. Zentral ist bei diesem Ansatz das *Frame-Konzept*, das von M. Minsky (1975) vorgestellt worden ist.

> *„When one encounters a new situation (or makes a substantial change in one's view of the present problem) one selects from memory a substantial structure called a frame. This is a remembered framework to be adapted to fit reality by changing details as necessary…."*

Die Kernidee dieses Ansatzes ist, dass stereotype Situationen durch einen Frame beschrieben werden, der die üblichen Eigenschaften der Situation beinhaltet und so eine abstrakte, generische Beschreibung darstellt. Durch das Festlegen der konkreten Eigenschaften wird aus der stereotypischen Situation eine spezielle Situation, die eine Instanz des generischen Konzepts ist.

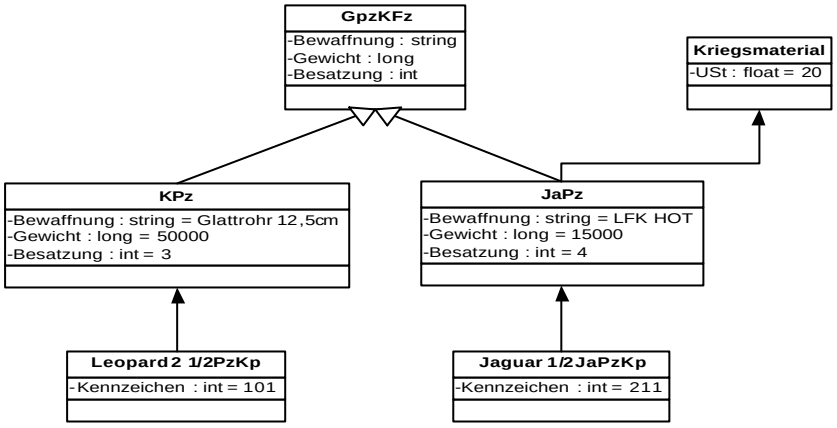

Abb.10: Einfaches Framesystem

Zur Beschreibung verwendet man für jeden generischen Frame ein Prädikatensymbol und für jeden Instanz-Frame ein Konstantensymbol. Jedes Attribut wird durch ein zweistelliges Prädikat *A(x,y)* beschrieben, wobei *x* das Objekt ist, dem das Attribut zugeordnet ist, und *y* der Wert des Attributs. Jede Instanzierungs-Beziehung wird durch einen Fakt beschrieben, und jede Subkonzept/Superkonzept Beziehung durch eine Regel:

KPz (Leopard 2 1/2PzKp)
Kriegsmaterial (Jaguar 1/2JaPzKp)
JaPz (Jaguar 1/2JaPzKp)
$\forall x(\text{KPz}(x) \rightarrow \text{GpzKFz}(x)$
$\forall x(\text{JaPz}(x) \rightarrow \text{GpzKFz}(x)$
$\forall x(\text{JaPz}(x) \rightarrow \text{Kriegsmaterial}(x)$

Die Attribute eines Instanz-Frames werden durch Fakten beschrieben, die Attribute eines generischen Frames durch Regeln:

Kennzeichen (Leopard 2 1/2PzKp,101)
Kennzeichen (Jaguar 1/2JaPzKp,211)
$\forall x$(KPz(x)→Bewaffnung(x, *Glattrohr 12,5cm*)
$\forall x$(KPz(x)→Gewicht(x, *50000*)
$\forall x$(KPz(x)→Besatzung(x, *3*)
usw.

Komplexere Frames

In richtigen Framesystemen werden wesentlich komplexere Framestrukturen verwendet, die eine facettenreiche Wissensrepräsentation erlauben, in der prozedurale und deklarative Elemente vermischt werden. Beispiele für solche Framesysteme sind:

- **FRL (Frame Representation Language):** Diese Sprache wurde in den 70er Jahren am MIT entwickelt (Roberts & Goldstein, 1977).
- **RLL (Representation Language Language):** Eine frame-basierte Meta-Sprache, die zur Definition von spezifischen Sprachen zur Wissensrepräsentation gedacht war. Die Sprache ist selbst-deskriptiv.
- **KEE (Knowledge Engineering Environment):** KEE ist ein kommerzielles hybrides Werkzeug für den Bau von WBS, das von der Firma Intellicorp in den 80er Jahren entwickelt wurde.
- **FrameWork**: Ein frame-basiertes System, das auch Konzepte der objekt-orientierten Programmierung wie Message Passing und die Vererbung von Methoden unterstützt (Kantrowitz, 1993).

Semantische Netze

Ein semantisches Netz (engl. semantic network) besteht aus Knoten, die Konzepte repräsentieren und Verbindungen (Kanten), die Beziehungen (Relationen) zwischen den Knoten herstellen. Semantische Netze werden zur formalen Wissensrepräsentation

genutzt. Semantische Netze können also in Form von verallgemeinerten Graphen dargestellt werden.
Die Relationen zwischen einzelnen Graphenknoten können u.a. sein:

- Hierarchische Relationen
- Vererbungsrelation (z.B. ist ein Hund ist ein Vertreter der Klasse Tier)
- Instanzrelation (z.B. ist Bello ein Instanz der Klasse Hund)
- Partitive Relation (z.B. ist Bellos Fell ein Teil von ihm)
- Eigenschaftsrelation (z.B. ist Bellos Fell wuschelig)

Da die Relationen eines semantischen Netzes immer Teil desselben sind, kommt es immer auf den Einzelfall an, welche Relationen vorhanden sind und was sie bedeuten. Eine einfache Form eines semantischen Netzes mit beschränktem Satz von Relationen ist ein Thesaurus.
Semantische Netze wurden in den frühen 60er Jahren von dem Sprachwissenschaftler R. R. Quillan als Repräsentationsformen von semantischem Wissen vorgeschlagen.

Nach Tim Berners-Lee soll das heutige Datennetz WWW in einem semantischen Netz („Semantic Web“) aufgehen, indem dessen Inhalte mittels RDF und anderer Standards mit wohldefinierten Bedeutungen versehen werden, die eine inhaltsbezogene Informationssuche ermöglichen werden.

Einige bekannte semantische Netzformalismen sind sind zum Beispiel:

- **Conceptual Graphs** (Sowa 1976;1984): Ein Formalismus, der eine graphische Notation für Prädikatenlogik mit Typen zur Wissensrepräsentation vorsieht. Dieser Ansatz ist in der weiteren Folge in einem breiteren Umfeld zu „Conceptual Structures“ ausgebaut worden. Anwendungen liegen vor allem in den Bereichen natürliche Sprachverarbeitung, fallbasiertes Schließen und Modellierung von Informationssystemen.
- **SNePs (Semantic Network Processing System** (Shapiro 1979): Dieses System, das in LISP geschrieben wurde, ist als System zur Wissensrepräsentation für einen kognitiven

Agenten konzipiert worden, der natürliche Sprachen verwendet.

- **NETL** (Fahlman 1979): Ein Netzformalismus, der auf Parallelverarbeitung in einer SIMD Architektur (Single Instruction, Multiple Data) ausgerichtet ist. Inferenz wird durch Propagierung von Markierungen im Netz betrieben.
- **KL-ONE** (Bachman 1979): Dieses System ist stark vererbungsbezogen. Die Semantik der Formalismen ist über eine Umsetzung in logik-basierter Darstellung definierbar. Entsprechende syntaktische Konstrukte haben zu einer eigenen Beschreibungslogik geführt.

Mögliche Formate zur Speicherung semantischer Netze sind das „Ressource Description Framework“ RDF, auf das im weiteren Verlauf der Arbeit noch genauer eingegangen wird, da damit Informationen für eine effiziente Suche im Internet bzw. Intranet realisiert werden können, und Topic Maps

Inferenz

Ein wesentliches Merkmal wissensbasierter Systeme ist die Trennung von Problemwissen und Wissensverarbeitung. Während das Wissensmanagement zum Teil mit konventionellen Techniken der Informatik wie beispielsweise Datenbanken bewerkstelligt werden kann, spielen Methoden der Inferenz eine zentrale Rolle in der Wissensverarbeitung. Die bisher beschriebenen Formalismen zur Repräsentation von Wissen bedingen spezielle Verfahren, mit denen auf dieses Wissen zugegriffen werden und aus diesem Wissen neues, nicht explizit repräsentiertes Wissen generiert werden kann. Eine zentrale Rolle spielen hierbei Methoden der Inferenz.

Inferenz in regelorientierten Repräsentationen

Die Inferenzstrategien in regelorientierten Systemen lassen sich in zwei Gruppen aufteilen:

- **Vorwärtsverkettende Systeme:** Die Verarbeitung findet datengetrieben statt, d.h. ausgehend von den Daten wird nach Wissen gesucht, das anwendbar ist.
- **Rückwärtsverkettende Systeme:** Die Verarbeitung findet zielorientiert statt, d.h. ausgehend vom Gesamtziel wird nach Wissen gesucht, das anwendbar ist und das das Gesamtziel in ein oder mehrere, einfachere Unterziele zerlegt.

Vorwärtsverkettende Systeme

Viele der wichtigen und auch wirtschaftlich erfolgreichen Expertensysteme wurden als so genannte Regel- oder Produktionensysteme realisiert, so zum Beispiel MYCIN zur Diagnose von bestimmten Infektionskrankheiten oder XCON zur Konfiguration von VAX-Rechnern der Firma DEC. Neben diesen wurden solche Produktionensysteme auch in Programmen wie SOAR und ACT eingesetzt, welche das kognitive Verhalten von Menschen simulieren. Solche Regel- oder Produktionensysteme bestehen normalerweise aus folgenden Elementen:

- Dem Arbeitsspeicher (working memory, WM)
- Dem Regelspeicher (rule memory, RM)

Die Elemente des Arbeitsspeichers sind die Fakten des Produktionensystems. Im Prinzip besteht das WM aus einer Menge von Typen sowie deren Instanzen, den Elementen des WMs (WME). Es gibt dabei unterschiedliche Sichten auf diese WME. Eine assoziiert mit den WMEs eine frameartige Repräsentation, eine andere sieht die WMEs als Datenbanktupel.
Die Elemente des Regelspeichers kodieren zumeist das Expertenwissen über die Zusammenhänge der Fakten in der Form „wenn Bedingung, dann Aktion“, wobei implizit angenommen wird, dass, wenn mehrere Bedingungen auftreten, diese konjunktiv

verknüpft sind. Die Verarbeitung in einem solchen System erfolgt in folgenden Arbeitsschritten.

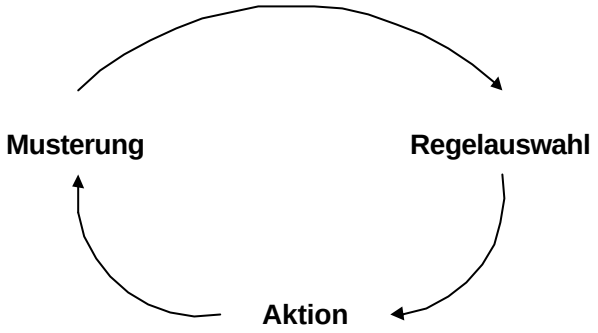

Abb.11: Verarbeitungsablauf in einem vorwärtsverkettenden System

In der ersten Phase werden die Bedingungen aller Regeln daraufhin untersucht, ob diese durch Elemente des WM erfüllt werden. Resultat dieser Musterung ist die so genannte Konfliktmenge, die die Regelinstanzen enthält. Die gebunden WMEs sind genau jene, die die Bedingungen der Regeln erfüllen. Im zweiten Schritt wird eine Regelinstanz aus dem „Conflict Set“ (CS) zur Verarbeitung ausgewählt. Die Auswahl kann durch verschiedene Merkmale erfolgen:

- Verhinderung mehrfacher Ausführung von Regeln mit gleichen Daten zur Verhinderung von Endlosschleifen.
- Auswahl aufgrund zeitlicher Bedingungen.
- Auswahl aufgrund der syntaktischen Struktur der Regel.
- Bevorzugte Regel aufgrund von Meta-Wissen.
- Zufällige Auswahl.

Danach wird die Aktion der im zweiten Schritt ausgewählten Regel ausgeführt. Dies bewirkt zumeist eine Änderung des WM. Der Abarbeitungszyklus beginnt von vorne. Dieser „Recognize-Act-Cycle“ bildet die grundlegende Kontrollkomponente in einem Produktionensystem. Die gesamte Verarbeitung terminiert, wenn entweder eine explizite „halt“-Anweisung ausgeführt wird, oder aber das CS nach der Musterung leer ist.

Rückwärtsverkettende Systeme

Die Alternative zur Abarbeitungsstrategie der vorwärtsverkettenden Produktionensysteme sind die rückwärtsverkettenden. Der vielleicht bekannteste zielorientierte Formalismus hierbei ist der der Hornklauseln. Diese haben zwei mögliche Interpretationsformen: einerseits eine logische, andererseits eine prozedurale. Das bekannteste Hornklauselsystem ist die Programmiersprache PROLOG („Programming in Logic"). Eine wesentliche Forderung an Kalküle, die zur Automatisierung und Mechanisierung der Inferenz dienen, ist die zielgerichtete Generierung von Instanzen. Bei den rückwärtsverkettenden Systemen gibt es einen der Musterung ähnlichen Mechanismus, die Unifikation.

Den Ausgangspunkt bildet der Zielzustand. Es wird nun versucht, diesen Zielzustand aus der Wissensbasis abzuleiten, d.h. zu bestimmen ob eine Aussage wahr oder falsch ist. Zuerst wird in der Wissensbasis gesucht, ob die dem Zielzustand entsprechende Aussage als Fakt in der Wissensbasis enthalten ist. Danach wird untersucht, ob es die Prämissen jener Regel als Fakten gibt, die den Zielzustand als Konklusion enthalten. Gibt es diese ebenfalls nicht, wird rückwärtsgehend versucht, die für die Generierung der Zielaussage notwendigen Fakten aus der Wissensbasis abzuleiten. Letztendlich feuern jene Regeln in umgekehrter Reihenfolge, die zur Erzeugung der Aussage des Zielzustandes bzw. deren Widerlegung notwendig sind.

Allgemeine Resolution

Während die Hornklausen aufgrund der Definition ein eingeschränkter Formalismus sind, stellt der Resolutionskalkül eine Inferenzstrategie für allgemeine Klauselmengen dar. Dieser wurde 1965 von J.A. Robinson zusammen mit dem Unifikationsprinzip eingeführt. Der Kalkül arbeitet auf einer syntaktischen Unterklasse der Formeln der Prädikatenlogik erster Stufe. Durch die Verwendung

von Klauselformen wird sowohl die Struktur der Eingabeformel als auch die Anzahl der vorkommenden Konnektive und Quantoren eingeschränkt, was zu einem sehr einfachen und vielfach praktisch effizienten Kalkül führt. Dabei werden zuerst beliebige Formeln der Prädikatenlogik in eine einfache Klauselform (Konjunktive Normal-Form, KNF) transformiert. Da ein Widerlegungsbeweis konstruiert werden soll, wird in der weiteren Folge versucht, aus dem Negat der zu beweisenden (geschlossenen) Formel mit Hilfe der Resolutionsprozedur einen Widerspruch in Form der leeren Klausel herzustellen. Möglich wird dies, da ja bekanntlich gilt:

$\neg\varphi$ ist unerfüllbar $\Leftrightarrow$ φ ist Tautologie

Der Resolutionskalkül

Der Resolutionskalkül ist sehr kompakt und besteht im Wesentlichen nur aus zwei Regeln. In der Aussagenlogik sind dies:

$$\frac{L \vee C_1 \qquad \neg L \vee C_2}{C_1 \vee C_2}\,\mathrm{Re}\,s \qquad \text{und} \quad \frac{L \vee L \vee C_1}{L \vee C_1}\,Fak$$

Wobei Res die (aussagenlogische) Resolutionsregel und Fak die (aussagenlogische) Faktorisierungsregel bezeichnet. In der Resolutionsregel heißen die beiden Klauseln der Prämissen die Elternklauseln, die Klausel $C_1 \vee C_2$ der Konklusion heißt Resolvent über das Literal L. In der Faktorisierungsregel heißt die Konklusion $L \vee C_1$ der Faktor der Prämisse $L \vee L \vee C_1$.
Der prädikatenlogische Resolutionskalkül besitzt die folgenden beiden Inferenzregeln:

$$\frac{L \vee C_1 \qquad \neg K \vee C_2}{C_1 \boldsymbol{s} \vee C_2 \boldsymbol{s}}\,\mathrm{Re}\,s \qquad \text{und} \quad \frac{L \vee K \vee C_1}{L\boldsymbol{m} \vee C_1 \boldsymbol{m}}\,Fak$$

Die Namensgebung der einzelnen Klauseln und Operationen erfolgt analog zu den Regeln für die Aussagenlogik.

Default-Logik

Nicht-monotones Schließen bzw. Default-Schließen bedeutet, dass ein gezogener Schluss unter Umständen ungültig sein, bzw. werden kann. Das Erkennen neuer Fakten kann alte Schlüsse ungültig machen.
Dies kann in unterschiedlichen Situationen nötig sein:

- Unvollständiges Wissen
- Änderung der Fakten
- Eine Darstellung, die es erlaubt, Ausnahmen anzugeben; ermöglicht unter Umständen eine dichtere Repräsentation (und damit effizienteres Schließen).

Beispiel: Tweety ist ein Vogel. Unter der Annahme, dass Vögel fliegen können, kann man auch davon ausgehen, dass Tweety fliegen kann. Obwohl Vögel üblicherweise fliegen können, kann Tweety nicht fliegen, weil er ein Pinguin ist und Pinguine zwar Vögel sind, jedoch nicht fliegen können.

Fuzzy-Logik

Die Fuzzy-Logik beschäftigt sich mit dem Begriff der Vagheit. Vagheit bedeutet, dass Aussagen nicht exakt bestimmt, d.h. nicht genau wahr oder falsch sind. Nehmen wir an, eine Person ist 170 cm groß. Ist diese Person nun groß oder klein? Die Fuzzy-Logik beschäftigt sich nun damit, das Problem der Vagheit in ein Regelsystem zu packen und zu kodieren. Fuzzy-Systeme kodieren demnach direkt strukturiertes Wissen (Regeln) in numerischer Form. Die Fuzzy-Set-Theorie (Unscharfe Mengen = Fuzzy-Sets) wurde 1965 von Prof. Zadeh (Universität Berkeley, Kalifornien) eingeführt. Prof. Zadeh stellte fest, dass herkömmliche (Computer-) Logik keine Manipulation von Daten kennt, die vage oder subjektive Konzepte repräsentieren. (z.B. Es ist ziemlich kalt, eine schöne Frau). Die Fuzzy-Set-Theorie geht von der Annahme aus, dass alle Dinge nur zu einem gewissen Grad zutreffen und reduziert die herkömmliche

Logik auf einen Sonderfall. Gerade der Mangel an Präzision ermöglicht: Ein Treffen von Entscheidungen, selbst in Situationen, in denen unvollständige oder teilweise widersprüchliche Informationen vorliegen. Die Zugehörigkeitsfunktion zur scharfen Menge ist durch zugehörig („1“) oder nicht zugehörig („0“) bestimmt. Sind Werte der Zugehörigkeitsfunktion nicht nur Null oder Eins, sondern beliebige Werte zwischen 0 und 1, dann spricht man von einer unscharfen Menge (Fuzzy-Set). Ein Fuzzy-Set wird durch die Zugehörigkeitsfunktion immer eindeutig dargestellt. Dadurch werden beliebig feine Abstufungen zwischen „gehört dazu" und „gehört definitiv nicht dazu" vorgenommen.

Zur Formulierung von menschlichem Erfahrungswissen werden Fuzzy-Regeln verwendet.
Beispiel:
Regeln beim Autofahren:

- Wenn der Abstand zum vorderen Auto klein ist und die Geschwindigkeit groß, dann bremse mit großer Kraft.
- Wenn der Abstand zum vorderen Auto mittel ist und die Geschwindigkeit groß, dann bremse mit mittlerer Kraft.

Die Aufgabe der Inferenz ist es, Eingabewerte mit dem sprachlich formulierten, unscharfen Regelwerk zu verknüpfen und daraus Folgerungen zu ziehen. Im allgemeinen Fall können die Eingabewerte unscharfe Fuzzymengen sein; hier soll aber weiterhin davon ausgegangen werden, dass es sich bei den Eingabewerten um scharfe Messwerte handelt.
Seit Aristoteles benutzt man in der binären Logik den Modus-Ponens: Aus der allgemeinen Regel

- "Alle Menschen sind sterblich." und der konkreten Vorgabe
- "Sokrates ist ein Mensch." schließt man
- "Sokrates ist sterblich."

Implikation:	**x ist ein Mensch ⊗**	**x ist sterblich**
Prämisse:	Sokrates ist ein Mensch	
Folgerung		Sokrates ist sterblich

Die allgemeine Regel lässt sich als Implikation formulieren: „Wenn x ein Mensch ist, dann ist x sterblich“.

Bei der Fuzzylogik wird dieser binäre Modus-Ponens zunächst aufgeweicht:
Weil hier unscharfe Begriffe benutzt werden, können wir bei der Folgerung nicht davon ausgehen, dass der DANN-Teil der Implikation die Folgerung exakt angibt. Darüber hinaus soll die Folgerung aber auch berücksichtigen, inwieweit die Prämisse den WENN-Teil der Implikation erfüllt. Insbesondere soll die Folgerung nur in geringem Maße zutreffen, wenn die Prämisse den WENN-Teil nur wenig erfüllt. Man beachte, dass im Gegensatz dazu beim aristotelischen Modus-Ponens die Prämisse „Sokrates ist kein Mensch." *nicht* zur Folgerung „Sokrates ist nicht sterblich." führt.

In den Prämissen von Regeln werden linguistische Werte durch logische Operationen miteinander verknüpft. Bevor dies jedoch geschehen kann, müssen die aktuellen Eingangswerte erst fuzzyfiziert werden. Unter Fuzzyfizierung versteht man die Ermittlung der Zugehörigkeitsgrade zu den unscharfen Mengen, ausgehend von aktuellen Werten der Eingangsgrößen. Der Zugehörigkeitsgrad wird mit Hilfe der Zugehörigkeitsfunktion $\mu(u)$ ermittelt, d.h. der Eingangswert führt über die Zugehörigkeitsfunktion zu einem Zugehörigkeitsgrad der entsprechenden unscharfen Menge.

Im nächsten Schritt werden die linguistischen Werte logisch miteinander verknüpft. In diesem Beispiel sind die linguistischen Werte UND-verknüpft (Mensch und sterblich). Daher ist der Grad der Vorbedingung durch das Minimum der Zugehörigkeitsgrade der Eingangsgrößen bestimmt. Der Grad der Vorbedingung muss nun auf den linguistischen Wert der Aktion der Regel umgelegt werden. Diesen Schritt nennt man wieder Inferenz. Die Inferenz erfolgt,

indem man das Minimum zwischen dem Grad der Vorbedingung und der Zugehörigkeitsfunktion der Aktion bildet. Diese Methode der Inferenzbildung stellt jedoch nur eine Möglichkeit dar. Eine andere Möglichkeit ist die Produktbildung zwischen dem Grad der Vorbedingung und der Zugehörigkeitsfunktion der Aktion. Für welche Art der Inferenzbildung man sich entscheidet, hängt von der Art der Anwendung ab. Als letzten Schritt in einem Fuzzy-Regel-System muß noch eine konkrete Ausgangsgröße aus der Gesamtzugehörigkeitsfunktion ermittelt werden. Diesen Vorgang nennt man Defuzzyfizierung.

Neuronale Netze

Mit Hilfe der neuronalen Netze versucht man, bestimmte Verarbeitungstechniken, z.B. des menschlichen Gehirns, den biologischen Strukturen entsprechend, nachzubilden. Dies kann beispielsweise ein assoziatives Erinnern sehr einfacher Strukturen sein. Dabei hat man zwei Ziele: Zum einen will man Maschinen konstruieren, die mehr können als die sequentiell arbeitenden Computer, wie sie von Babbage oder von Neumann entwickelt wurden. Außerdem versucht man, die Funktionsweise des Gehirns, nicht seiner lokalchemischen Arbeitsweise, sondern des gesamten Nervensystems, verständlicher zu machen.

Die Modelle der Neuronalen Netzwerke werden notwendig, wenn man sich überlegt, dass schon ein sehr primitives Insekt im Flug Koordinierungsleistungen vollbringt, die selbst der modernsten digitalen Technik nicht zuzutrauen sind. Selbst die Möglichkeiten einer logik- und wissensbasierten künstlichen Intelligenz sind nicht so umfassend, dass sich aus ihnen alle menschlichen Eigenschaften nachbauen ließen.
Folgende Variablen bestimmen den Zustand des Neuronalen Netzwerkes:

1. Eine Zustandsvariable wird jedem Neuron zugeordnet.

2. Jeder Verbindung zwischen zwei Neuronen wird ein Konnektionskoeffizient zugeordnet.
3. Jedes Neuron besitzt eine Anregungsschnittstelle.
4. Für jedes Neuron wird eine Entwicklungsfunktion definiert.

Lernen in Künstlichen Neuronalen Netzen

Der Schwerpunkt liegt auf Algorithmen, die die Gewichte automatisch setzen bzw. adaptieren, da bei den meisten Verbindungen eine explizite „Programmierung" nicht möglich ist (vor allem bei den „Hidden Units"). Dieser Vorgang wird Training, der Algorithmus Lernregel genannt.
Es gibt drei wesentliche Lernprinzipien:

- Die Verstärkung von Verbindungen zwischen Units, die gleichzeitig aktiviert sind (d.h. eine von 0 verschiedene Aktivierung haben).
- Die Änderung des Gewichtes, so dass ein vorher definierter Fehler am Output so gering wie möglich wird (Fehlerminimierung).
- Das Annähern eines Gewichtes an den Aktivierungswert der vor der Verbindung liegenden Units (Angleichungslernen).

Delta-Regel (Widrow-Hoff-Regel): Zum Fehlerminimierungs-Lernen muss angenommen werden, dass es eine Vorgabe (oder „Target") für den gewünschten Output, also einen „Lehrer", gibt. Der momentane Output wird mit diesem Zielwert verglichen, und die Verbindung proportional zur resultierenden Differenz geändert. Mit dieser Lernregel können allerdings nur Netze ohne Hidden Units trainiert werden, da für diese keine Vorgabe angegeben werden kann (da sie sozusagen nur Vermittler sind, aber nicht direkt in das Endergebnis eingehen).

Backpropagation: Erweiterung der Delta-Regel. Es können damit Netze mit Hidden Units trainiert werden. Es wird aus den Fehlern der Outputunits ein „Pseudofehler" der Hidden Units berechnet, mit dessen Hilfe ähnlich wie oben die Gewichte, die zu

den Hidden Units hinführen, geändert werden können. Der Fehler wird also „zurückpropagiert“, ein Vorgang, den man in komplexeren Netzen auch öfter durchführen kann.

Aus mathematischer Sicht entspricht Fehlerminimierungslernen einem Gradientenverfahren, indem immer der Weg des steilsten Abstiegs (gegeben durch die erste Ableitung der Fehlerfunktion) genommen wird.
Wichtige Anwendungsbereiche für Neuronale Netze sind:

- **Diagnose und Interpretation**: z.B.: Qualitätskontrolle in der Produktion, Fehlerdiagnose
- **Vorhersage, Prognose**: z.B.: Aktienkurse, Verkaufszahlen
- **Clustering**: z.B.: Spracherkennung, EKG/EEG, Betriebszustände eines Prozesses
- **Mustererkennung**: z.B.: Zeichenerkennung, Unterschriftenverifikation, Satellitenbilderkennung
- **Robotik, Steuerung**: z.B.: Armsteuerung, Fahrzeugsteuerung (Autopilot)
- **Optimierung**: z.B.: Topologie (VLSI), logistische Probleme (Ressourcenzuteilung)
- **Kognitionswissenschaften**: z.B.: Modelle der Begriffsbildung, Lernen

Resource Description Framework (RDF)

Die derzeit gängige Vorgangsweise, um das Problem der Informationssuche zu bewältigen, ist die Volltext-Indizierung. Bei der Suche nach wissenschaftlichen Dokumenten etwa, erweist sich diese Vorgangsweise aufgrund der Schwierigkeit der Klassifizierung als unzweckmäßig.

Eine vernünftige, automatisierte Suche nach Internetdokumenten verlangt daher nach einer präzisen Beschreibung solcher Dokumente. Diese Informationen müssen demnach in das Dokument eingebettet werden. Dies kann aber nicht in unstrukturierter Form erfolgen, sondern muss, um eine effiziente Suche zu ermöglichen, gewissen Regeln folgen. Um eine solche Struktur einzuführen, verwendet man so genannte Metadaten, also Informationen über Informationen. In weiterer Folge soll nun ein Überblick über das „Resource Description Framework“ (RDF) des World Wide Web Consortium (W3C) gegeben werden, das es erlaubt ein Internetdokument zu beschreiben, ohne sich dabei auf ein bestimmtes Format oder ein Schema für die Repräsentation der Metadaten festzulegen. (RDF03a);

Die ersten Entwürfe des Resource Description Framework (RDF) veröffentlichte das W3C 1997. Ziel war es, einen Mechanismus zu schaffen, der eine automatische Verarbeitung von Metadaten ermöglicht. (RDF03b)

Grundkonzept

RDF ist eine deklarative Sprache zur Beschreibung von Begriffen bzw. Entitäten (Resources), deren Beziehungen untereinander (Statements) und die Arten dieser Beziehungen (Properties). Begriffe bzw. Resources können beliebige Konzepte (z.B.: Dokumente, Begriffe) sein und werden durch „Uniform Resource Identifiers“ (URI) eindeutig identifiziert. RDF kennt keinen speziellen vordefinierten Wortschatz zur Beschreibung von Metadaten, sondern

bietet die Möglichkeit verschiedene Standards (z.B.: Dublin-Core) einzubinden oder gar neue Schemata zu definieren.

Der grundlegende Gedanke bei RDF ist es, eine Sprache zu entwickeln, mit der man Aussagen über Dinge trifft, die mittels gerichteter Graphen modelliert werden können. Diese Graphen bestehen aus Knoten, zugehörigen Paaren von Attributen und Werten. Knoten sind beispielsweise Quellen im Internet, Attribute sind die Eigenschaften der Knoten, deren Werte entweder atomar sind (also Text Strings oder Zahlen) oder wiederum Quellen bzw. Dokumente mit Attributen. Solche Modelle werden in XML (eXtensible Markup Language) abgebildet, sind somit austauschbar und können maschinell verarbeitet werden. Hierbei ist XML aber keinesfalls zwingend vorgeschrieben, sondern nur eine Empfehlung des W3C zur Serialisierung der abgebildeten Informationen. RDF bedient sich der Syntax von XML, ohne dabei Annahmen über ein bestimmtes Anwendungsgebiet vorauszusetzen ([XML_RDF).

Resources

Alle in RDF beschriebenen Begriffe bzw. Entitäten werden als Resource bezeichnet. Das können ein HTML-Dokument, mehrere verknüpfte Internetseiten oder auch nur ein Teil einer Seite sein. Genauso können aber auch gedruckte Bücher als Resource betrachtet werden und somit nach dem RDF Konzept beschrieben werden.

Resources werden in RDF mittels URI identifziert. Folgende Tabelle stellt mögliche Ausprägungen von Resourcen und die dazugehörigen URIs dar. (RDF03a)

Arten von Resources	URI
Medien mit digitaler Manifestation (z.B.: Webseite, Bild in Datenbank, ...)	URI ist URL, mit der auf die Resource zugegriffen werden kann. (z.B.: *www.xyz.com/song.mp3*)
Abstrakte Begriffe ohne digitale Manifestation	URI dient zur global eindeutigen Identifikation. Zuordnung der Bedeutung (semantischen Interpretation) der Resource erfolgt nicht im Rahmen von RDF (z.B.: *www.xyz.com/properties#title*).
Literal Resource (einfacher Wert - z.B. String)	Keine eigene Identität, deshalb auch keine URI. (z.B.: Deep Down & Dirty)

Statements

Wie in der natürlichen Sprache erfolgt die Beschreibung von Begriffen bzw. Entitäten durch Aussagen (z.B.: Dieses MP3 hat den Titel „Deep Down and Dirty"). In RDF entspricht eine Aussage einem Statement, welches sich wiederum aus drei Teilen zusammensetzt. Sie werden in Anlehnung an die Grammatik als Subject (bzw. Resource), Predicate (bzw. Property) und Object (bzw. Value) bezeichnet. Das Object (der Wert der Eigenschaft) eines RDF-Statements kann einer anderen Resource, einem einfachen String (Literal) oder einem anderen einfachen Datentypen entsprechen.

Eine Beschreibung erfolgt in der Regel durch eine Menge von Aussagen bzw. Statements der Form subject-predicate-object, wobei die Abbildung auf drei verschiedene Arten erfolgen kann (RDF03a, RDF03b):

- als gerichteter virtueller RDF-Graph,

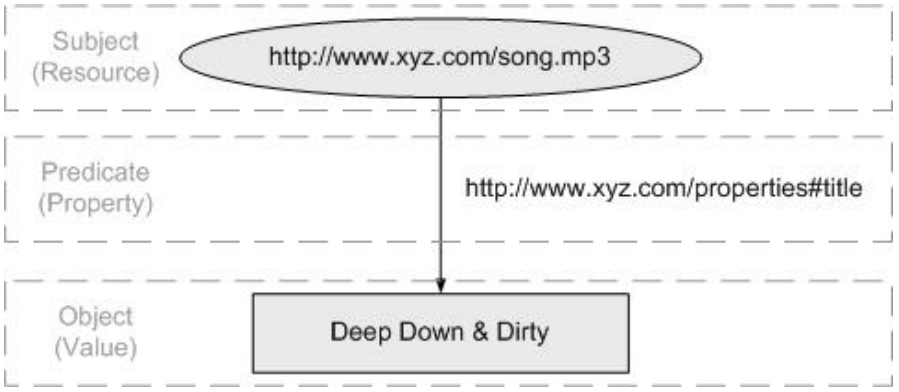

Abb.12: Einfacher RFD Graph

- in Form von Triples-Syntax,

<subject><predicate><object> bzw.

<resource><property><value>

<http://www.xyz.com/song.mp3>
<http://www.xyz.com/properties#title>
<http://www.xyz.com/title#DeepDown&Dirty>

- oder in einem XML-Format (siehe unten).

Properties

Die Properties bzw. Eigenschaften einer Resource können spezielle Charakteristika, Attribute oder Beziehungen zu anderen Resources sein. Die spezifischen Bedeutungen der Properties legen fest, welche Werte sie annehmen können. Das heißt, dass diese Properties gemäß den Anforderungen des Anwendungsgebietes frei definierbar sind. Einige Properties sind bereits vom W3C vordefiniert (z.B.: *http://www.w3.org/1999/02/22-rdf-syntaxns#type* zur Deklaration des Typs einer Ressource).

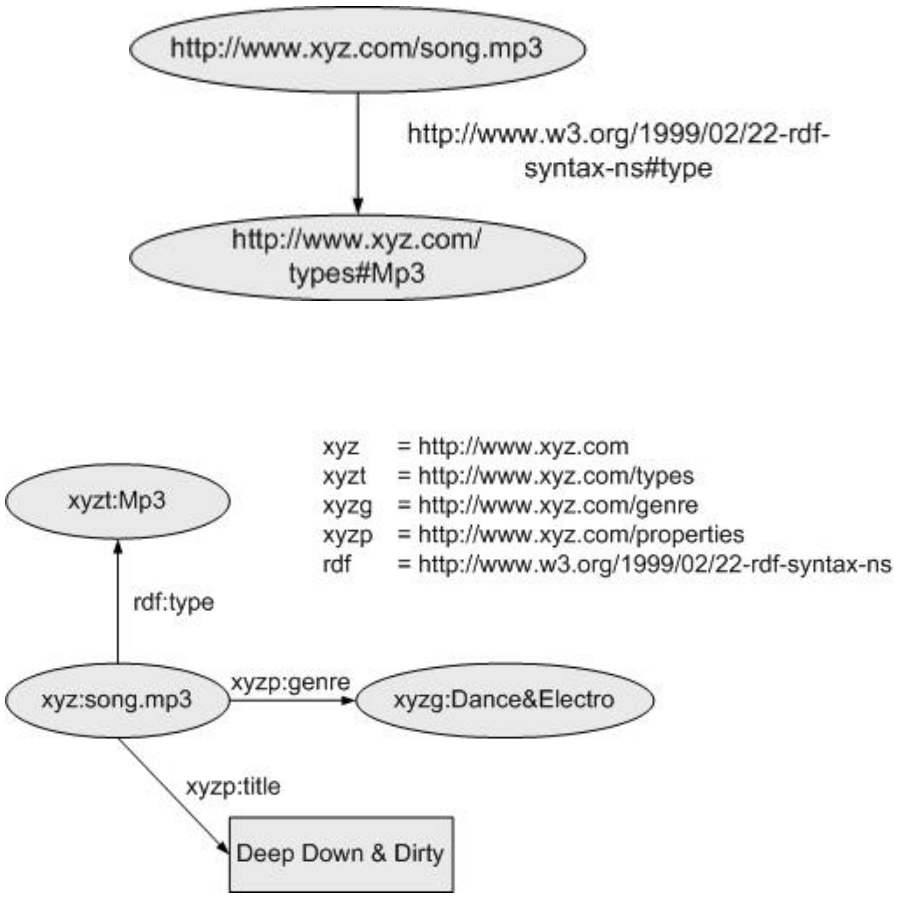

Abb 13: Kombination und Erweiterung des bisherigen Beispiels

XML Syntax: RDF/XML

Als maschinell verarbeitbares Austauschformat für die RDF-Graphen wird üblicherweise XML verwendet. In der RDF-Spezifikation wurde die dafür notwendige Syntax definiert. Im Folgenden wird das obige Beispiel in XML dargestellt (XML_RDF):

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf=http://www.w3.org/1999/02/22-rdf-syntax-ns#
        xmlns:xyzp="http://www.xyz.com/properties#">

    <rdf:Description rdf:about="http://www.xyz.com/song.mp3">
        <rdf:type rdf:resource="http://www.xyz.com/types#Mp3"/>
        <xyzp:genre rdf:resource="http://www.xyz.com/genre#DanceAndElectro"/>
        <xyzp:title>DeepDownAndDirty</xyzp:title>
    </rdf:Description>

</rdf:RDF>
```

Darstellung des letzten Beispiels in RDF/XML (Kurzversion):

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf=http://www.w3.org/1999/02/22-rdf-syntax-
ns#
    xmlns:xyzp="http://www.xyz.com/properties#"
    xmlns:xyzt="http://www.xyz.com/types#">

  <xyzt:Mp3 rdf:about="http://www.xyz.com/song.mp3">
    <xyzp:about
rdf:resource="http://www.xyz.com/genre#DanceAndElectro"/>
    <xyzp:title>DeepDownAndDirty</xyzp:title>
  </xyzt:Mp3>

</rdf:RDF>
```

Erweiterte Konzepte

Containers

Container ermöglichen die Gruppierung von Resources bzw. die Modellierung n-ärer Beziehungen durch die Einführung künstlicher Resources vordefinierten Typs. RDF definiert für diese Zwecke drei solche Containertypen:[RDF03b]

- Bag *(rdf:Bag)*: Entspricht einer ungeordneten Menge von Elementen. Ist die Reihenfolge der Resources nicht wichtig, wird dieser Typ verwendet.

- Sequenz *(rdf:Seq)*: Ist die Reihenfolge der Resources relevant, verwendet man eine Sequenz.
- Alternative *(rdf:Alt)*: Wird verwendet, wenn von mehreren gültigen Werten nur einer ausgewählt wird. Es muss mindestens ein Wert in der Auswahlliste stehen. Der erste Wert gilt als Defaultwert. Die Reihenfolge ist dabei wieder unwichtig.

Die Referenzierung der Mitglieder in einem Container erfolgt mittels vordefinierter Properties, wie zum Beispiel *rdf:_1, rdf:_2,..., rdf:_n*. Die entsprechende Ordnung (wenn die Reihenfolge beachtet wird) erfolgt dabei über die im Property enthaltene Zahl. Die folgende Abbildung soll diesen Sachverhalt anhand des begleitenden Beispiels veranschaulichen:

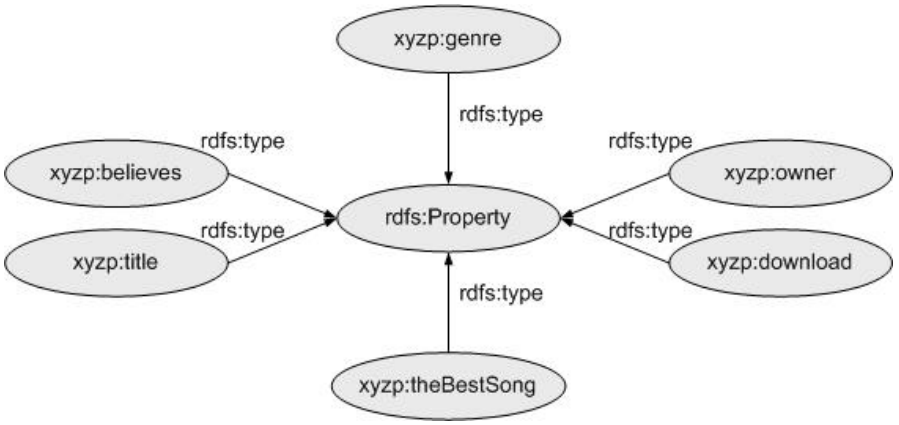

Abb. 14: Container – Verschiedene Orte für den Download des Songs: „song.mp3"

Collections

Container sind offene Strukturen, die ihre Gruppenmitglieder referenzieren. Es kann aber keine Aussage über die mögliche Existenz anderer Mitglieder getätigt werden. Collections hingegen sind geschlossene Gruppierungen in Form verketteter Listen. Die

Konstruktion einer solchen Liste erfolgt über das in RDF vordefinierte Vokabular:

- *rdf:List*: Gibt die Collection an
- *rdf:first*: Kennzeichnet das erste Element
- *rdf:rest*: Kennzeichnet den Rest der Liste
- *rdf:nil*: Als leere Liste, bzw. leerer Rest der Liste.

In der folgenden Abbildung wird eine zu unserem Beispiel passende Collection „The owners of the file song.mp3 are UniversalRecords, StereoMcs and AnyMusicPirate" als RDF Graph dargestellt.

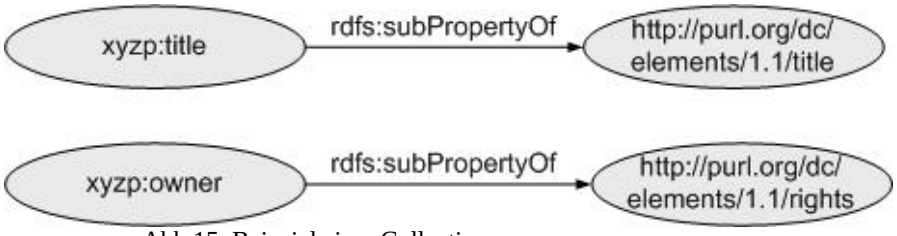

Abb.15: Beispiel einer Collection

Reifications

Als Reification in RDF bezeichnet man die Repräsentation eines Statements durch eine Resource, also Statements über Statements. Zu diesem Zweck wird eine künstliche Resource des vordefinierten Typs *rdf:Statement* eingeführt, die das zu reifizierende Statement repräsentiert. Ausgehend von dieser Resource wird die Referenzierung von Subject, Predicate und Object des Statements mittels vordefinierten Properties durchgeführt *(rdf:subject, rdf:predicate, rdf:object).* Unten stehende Abbildung zeigt eine solche, zu unserem Beispiel passende Reification: „The MusicExpert believes that song.mp3 is TheBestSong of the StereoMcs Album DeepDownAndDirty".

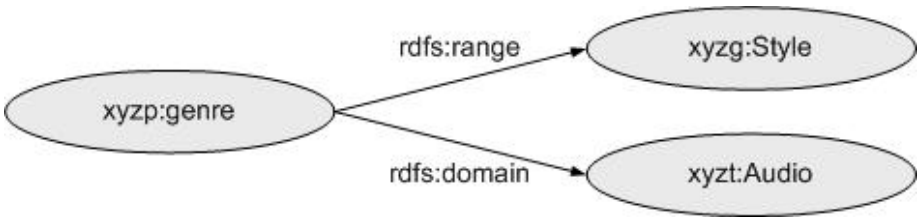

Abb.16 : Beispiel einer Reification

RDF-Schema

Grundkonzept

Wie weiter oben schon beschrieben wurde, sollten Properties (Eigenschaften) auch die Beziehung zwischen Resources beschreiben können. Nun wird aber in RDF keine Möglichkeit angeboten, diese Beziehungen zwischen Resources darzustellen. Diese Aufgabe übernimmt in diesem Falle das RDF Schema, das im Wesentlichen eine Definitionssprache des RDF-Vokabulars ist. Diese Schemadefinitionen sind dazu selbst wieder in RDF geschrieben. Das RDF Schema definiert dabei zu jedem Property, welche Werte es annehmen kann, welche Beziehung es zu anderen Properties besitzt und welche Art von Resources diese hat. Dadurch kann eine kontrollierte Nutzung von vordefinierten Properties und Resource-Type (Klassen) innerhalb eines gewissen Anwendungsbereiches sichergestellt werden. Diese Schemadefinition und dessen Ausprägungen bilden einen gemeinsamen, virtuellen RDF-Graphen und können somit nach den Vorgaben von RDF/XML in ein maschinenlesbares Austauschformat übersetzt werden.

Das RDF Schema als Klassensystem

Das RDF-Schema ähnelt sehr dem Konzept einer objektorientierten Sprache wie beispielsweise Java oder C++. Ein wesentlicher Unterschied besteht allerdings darin, dass bei objektorientierten

Sprachen eine Klasse durch die enthaltenen Eigenschaften und Methoden definiert wird. Beim RDF-Schema können Klassen auch ohne ihre vordefinierten Eigenschaften aufgefunden werden, da stets versucht wird, für alle Properties festzulegen, auf welche Klassen sie anwendbar sind. Das RDF-Schema ermöglicht es auch, bei Bedarf neue Properties zu definieren und diese dann einzelnen Klassen zuzuordnen, ohne bestehende Anwendungen zu beeinträchtigen.

Die Spezifikation von RDF-Schemata baut auf vordefinierten Ressourcen auf. Diese können dazu benutzt werden, Properties anderer Resources bzw. anderer Properties zu beschreiben. Die dazugehörige RDF-Definition wird mit dem namespace „rdfs“ bezeichnet und durch *http://www.w3.org/2000/01/rdf-schema/#* eindeutig identifiziert. Die Grundlage des RDF-Schemas bilden die Kernklassen *(rdfs:Resource, rdfs:Class* und *rdfs:Property)* und Kerneigenschaften *(rdf:type, rdfs:subClassOf, rdfs:PropertyOf, rdfs:subPropertyOf, rdfs:seeAlso und rdfs:isDefinedBy).*

Deklaration von Resource-Typen (Klassen)

rdfs:Class stellt eine Unterklasse von *rdfs:Resource* dar, welche wiederum die Klasse aller Resources, das heisst aller in RDF beschriebener Dinge, ist. Dabei ist zu beachten, dass *rdfs:Resource* selbst eine Instanz von *rdfs:Class* ist. Mit der Kerneigenschaft *rdf:type* wird ausgesagt, dass eine Ressource Instanz einer bestimmten Klasse ist.

Folgendes Beispiel soll den eben beschriebenen Sachverhalt veranschaulichen:

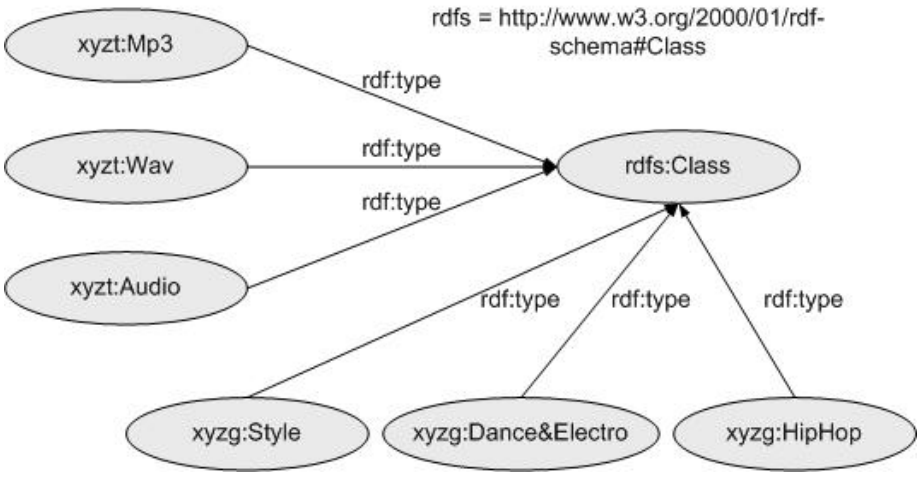

Abb.17 : Beispiel von Resource-Typen-Deklarationen

Dieses Beispiel kann durch eine Art Modularisierung, ähnlich der in objektorientierten Sprachen vorkommenden Vererbung, erweitert werden. Dabei wird auf eine vordefinierte Kerneigenschaft *rdfs:subClassOf* zurückgegriffen und folgende Spezialisierungsbeziehung zwischen zwei Unterklassen hergestellt:

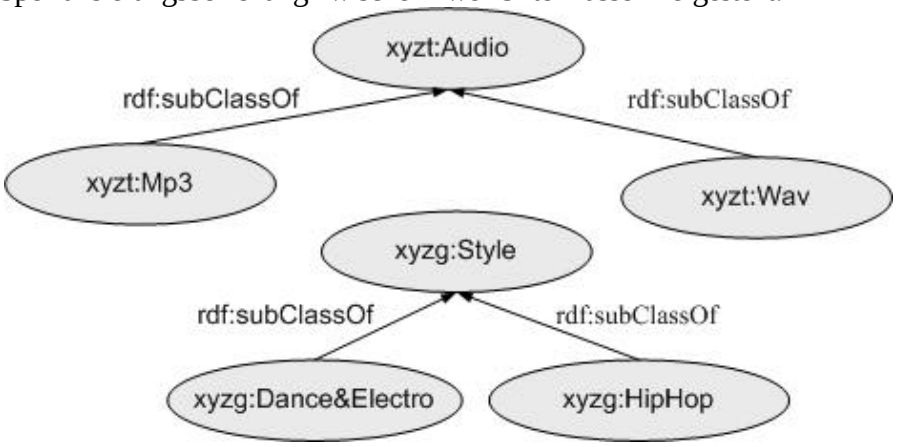

Abb.18 : Beispiel zu Spezialisierungsbeziehungen in RDFS

Deklaration von Properties

Die Klasse *rdfs:Property* wird analog dem Resource-Type *(rdfs:Class)* für Properties verwendet. Im folgenden Beispiel wird

mittels der Kerneigenschaft *rdf#type* bestimmten Resources die Property eines *rdfs:Property* zugewiesen.

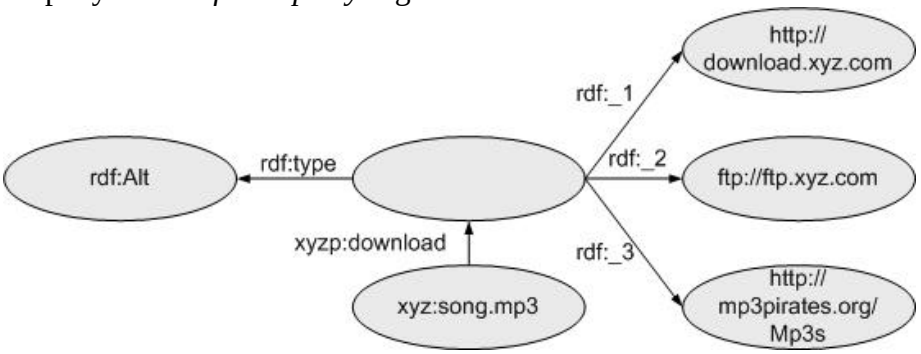

Abb.19 : RDFS Graph zur Deklaration von Properties

Dieses Beispiel kann nun wiederum durch Spezialisierungsbeziehungen zwischen gegebenen Properties erweitert werden. Im folgenden Beispiel wird dazu auf die Kerneigenschaft *rdfs:subPropertyOf* zurückgegriffen, um vererbte Eigenschaften darzustellen und wiederzugeben.

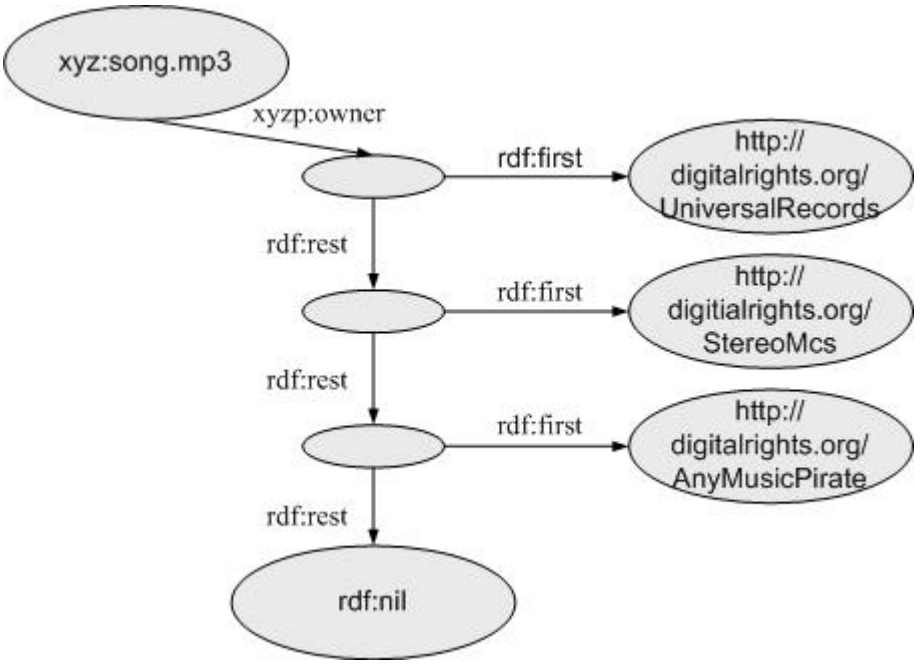

Abb.20 : Spezialisierungsbeziehung von Properties in RDFS

Einschränkungen von Properties

Ganz nach dem Konzept der Objektorientierung kann im RDF-Schema Vererbung stattfinden. Zusätzlich können Properties einer Beschränkung unterzogen werden. Mittels der zwei Kerneigenschaften *rdfs:domain* und *rdfs:range* kann nun ganz einfach der Definitions- und Wertebereich einer Resource festgelegt werden. Im folgenden Beispiel wird der Gültigkeitsbereich des Properties *xyzp#genre* auf die Resource-Typen *xyz#Audio* und *xyzg#genre* beschränkt. Dabei ist zu beachten, dass ein Property höchstens eine *rdfs:range* und eine *rdfs:domain*- Property besitzen darf.

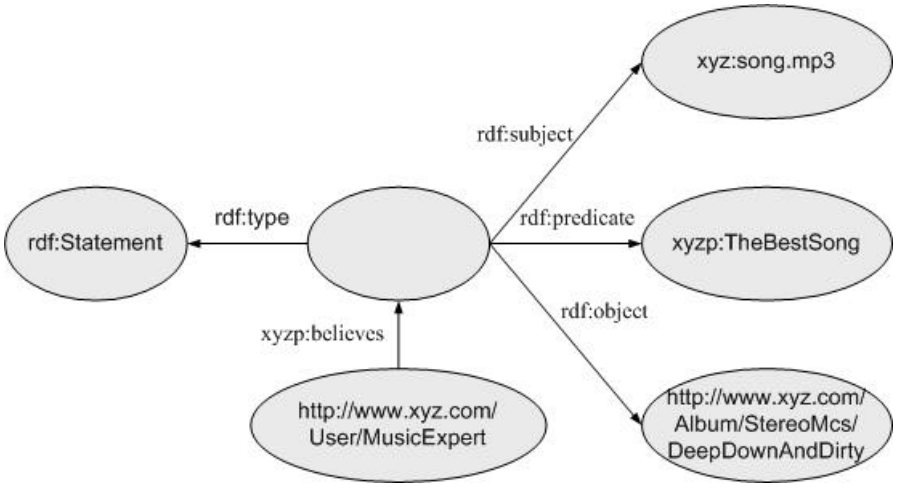

Abb.21 : Einschränkung des Properties *xyzp#genre*

Zusammenfassung RDF

RDF ist ein Metadaten-Framework, das zur Beschreibung von Resources auf semantischer Ebene herangezogen werden kann. Dabei ist es komplett unabhängig vom jeweiligen Bereich, auf den es angewendet werden soll. RDF umfasst sowohl ein graphenbasiertes Datenmodell, als auch eine XML-basierte Austauschsyntax. Das RDF-Schema ergänzt nun RDF um eine Definitionssprache für Beschreibungsschemata und basiert auf dem Datenmodell von RDF. Weiters stellt das RDF-Schema vordefinierte Kernressourcen und

Kerneigenschaften zur Verfügung, die für spezifische Anwendungsfälle weiter verwendet werden. Das RDF-Schema ist dabei ähnlich einem Klassenmodell einer objektorientierten Sprache zu verstehen.

Vorteile und Stärken von RDF und RDF-Schema

RDF ist ein sehr einfaches, dennoch äußerst mächtiges Datenmodell, das uns erlaubt, XML-Dokumente vielseitiger, d.h. auf semantischer Ebene, zu nutzen. Es können Tools darauf angewendet werden, die die Struktur erkennen und jedem Element dessen Eigenschaften und Typ zuordnen. Das RDF-Modell wird dabei formal spezifiziert. Es besitzt eine hohe Unabhängigkeit vom Anwendungsbereich, ist weiters schon sehr weit verbreitet und wird durch W3C stark unterstützt. Das RDF-Schema basiert auf RDF, wodurch eine sehr einfache, aber durchaus flexible Definitionssprache für Beschreibungsschemata geschaffen wurde. Eine weitere Stärke des RDF-Schemas ist, dass ähnlich wie bei objektorientierten Sprachen, Vererbungen, Mehrfachvererbungen, Spezialisierungen, sowie Einschränkungen von Properties zur Anwendung kommen können.

Nachteile und Schwächen von RDF und RDF-Schema

RDF stellt eine etwas gewöhnungsbedürftige Syntax des XML-Austauschformates dar und kann mittels herkömmlicher XML-Tools nur schwer verarbeitet werden. Eine weitere Schwäche ist die unklare Unterscheidung zwischen Medien und abstrakten Begriffen, was bedeutet, dass aufgrund einer URI nicht gesagt werden kann, welches Medium dahinter liegt. Gleichzeitig werden manche Eigenschaften nicht sehr sauber gelöst. Beispiele wären die Container, wo eine Ordnung über die Property-Bezeichnung angelegt wird, oder dass bei einem reifizierten Statement keine Reifikation möglich ist.

Zusammenfassung und Ausblick auf die Zukunft des Wissensmanagement

Knowledge-Management gilt heute als wichtiger Bereich für die effiziente Nutzung der Ressource Wissen und beschreibt den Schritt vom individuellen zum kollektiven Wissen eines Unternehmens. Bacons Satz „Wissen ist Macht“ wird durch das Wissensmanagement erweitert. „Wissen ist Macht, aber nur, wenn es auch weitergegeben wird“. Es reicht also nicht nur, hochqualifizierte Mitarbeiter einzustellen, wenn diese arbeits- und unternehmensrelevante Informationen nicht weitergeben. WMS- bzw. Knowledge-Management-Systeme werden auch in Zukunft vorrangig nur technische Unterstützung leisten können – der Input ins System muss von den Mitarbeitern selbst kommen.

Es ist auch nicht ausreichend, große Informationsmengen zu scannen und in Volltextdatenbanken abzulegen. Der entscheidende Schritt von der Information zu einem Wissensmanagement ist der von der Information zum Wissen. Dies wird durch Werkzeuge zur Verdichtung von Information, Erschließung von Zusammenhängen und Reduktion auf die wesentlichen Inhalte erreicht. Die Grundlagen des menschlichen Wissens wie die Einschätzung des Wertes einer Information in einem vorher noch nicht bekannten Zusammenhang, also das, was die Künstliche Intelligenz ausmacht, müssen den heutigen Lösungen erst noch implementiert werden. Zu den gewichtigsten Nutzaspekten eines Wissensmanagementsystems zählen die optimierte Wiederverwendung von schon existierenden Ergebnissen (lessons learned), bessere Ausschöpfung von best-practice Lösungen und die Reduktion von Fehlern.

Unabhängig von der verwendeten Software liegt die Herausforderung zur erfolgreichen Einführung eines Wissensmanagementsystems beim Management selbst. Entscheidungen für WM-Lösungen sind Entscheidungen der Unternehmensleitung. Es ist Aufgabe der Führung, zunächst die Voraussetzungen wie eine entsprechende Unternehmenskultur zu schaffen.

Abkürzungsverzeichnis

AI ..... Artificial Intelligence
FOL..... First Order Logic
FRL ..... Frame Representation Language
KEE..... Knowledge Engineering Environment
KI..... Künstliche Intelligenz
KNF..... Konjunktive Normalform
KNN ..... Künstliches Neuronales Netz
MIT..... Massachusetts Institute of Technology
PROLOG ..... Programming in Logic
RDF..... Resource Description Framework
RLL..... Representation Language Language
RM..... Rule Memory
SIMD ..... Single Instruction, Multiple Data
SNePs ..... Semantic Network Processing System
URI..... Uniform Resource Identifiers
URL..... Uniform Resource Locator
W3C..... World Wide Web Consortium
WBS..... Wissensbasiertes System
WME..... Working Memory Element
WMS..... Workflow Management System

Literatur

Wissensmanagement

[ANS66] Ansoff, H.I.: Management Strategie, München: Vahlen (1976).

[ARG78] Argyris, Chris; Donald A. Schön: Organizational Learning: A Theory of Action Perspective, Reading (Mass.) (1978).

[BAR80] Bartölke, Klaus: Organisationsentwicklung für entwicklungsfähige Organisationsmitglieder, in: Kappler, E. (1980): Unternehmensstruktur und Unternehmensentwicklung. Freiburg i.B., S. 319-344.

[BEC99] Beckman, T: The Current State of Knowledge Management. In: Knowledge Management Handbook. Hrsg. V. J. Liebowitz. Boca Raton (1999). S. 1.1-1.22.

[DUN79] Duncan, R.B.; A. Weiss: Organizational Learning: Implications for Organizational Design, in: Staw, B. (1979), S. 75-123.

[FUC73] Fuchs, Herbert; Erwin Grochla; Norbert Szyperski (Hrsg.): Systemtheorie und Organisation: Die Theorie offener Systeme als Grundlage zur Erforschung und Gestaltung betrieblicher Systeme, Wiesbaden: Gabler (1973).

[HAD81] Hedberg, Bo: How Organizations Learn and Unlearn, in: Nystrom, P. C.; W. H. Starbuck (Eds.): Handbook of Organizational design (1981), Bd. 1 S. 3-27.

[HEN91] Hentze, Joachim: Personalwirtschaftslehre 1, 5. Aufl. Bern, Stuttgart: Haupt (1991).

[KIR90] Kirsch, Werner: Unternehmenspolitik und strategische Unternehmensführung, München (1990).

[KLI79] Klix, Friedhart: Information und Verhalten: Kybernetische Aspekte der organismischen Informationsverarbeitung; Einführung in naturwissenschaftliche Grundlagen der allgemeinen Psychologie, 4. Aufl. Bern [u.a.] (1979).

[MIL86] Miller, M: Kollektive Lernprozesse. Studien zur Grundlegung einer soziologischen Lerntheorie, Frankfurt (1986).

[PIA73] Piaget, Jean: Einführung in die genetische Erkenntnistheorie, Frankfurt a.M.: Suhrkamp (1973).

[WII99] Wiig, K: An Emerging Discipline Rooted in a Long History Knowledge, Knowledge Insitute Inc., Draft of Chapter 1. In: Knowledge Management, Hrsg. V. D. Chauvel; C. Despres. (1999).

Wissensbasierte Systeme

[BRA90] Brachmann, R.J.: The Future of Knowledge Representation. Proceedings of AAAI'90, pp 1082-1092, (1990).

[BRO90] Brooks, R.: Elephants don't play chess, Robotics and Autonomous Systems, (1990).

[DEL79] Deliyanni, A.; Kowalski, R.A.: Logic and Semantic Networks. CACM, Vol.22, No.3, (1979).

[FAH79] Fahlman, S.: A System for Representing and Using Real-World Data. MIT Press, (1979).

[GIN93] Ginsberg, M.: Essentials of Artificial Intelligence. Morgan Kaufman, 1993.

[GOT90]: Gottlob, G.; Frühwirth, T; Horn, W.: Expertensysteme. Springers Reihe Angewandte Informatik. Springer (1990).

[HEL91] Helbig, H.: Künstliche Intelligenz und automatische Wissensverarbeitung. Berlin: Addison Wesley (1991).

[KAI89] Kaindl, H.: Problemlösen durch heuristische Suche in der Artificial Intelligence. Springer, (1989).

[KIN92] Kinnebrock, W.: Neuronale Netze, Grundlagen, Anwendungen, Beispiele. 2. verbesserte Auflage. München u.a.: Oldenbourg (1992).

[KÖH90] Köhle, Monika: Neuronale Netze. Wien: Springer (1990).

[KRU91] Kruse, H..; Mangold, R.; Melchler,B.; Penger, O.: Programmierung Neuronaler Netze. Bonn: Addison-Wesley (1991).

[LEH92] Lehmann, F.: Semantic Networks in Artificial Intelligence. Pergamon Press, (1992).

[NIL80] Nilsson, N.: Principles of Artificial Intelligence. Morgan Kaufmann. (1980).

[New76] Newell, A.; Simon, H.A.: Computer Science as Empirical Inquiry: Symbols and Search. In: Communications of the ACM 19 (1976).

[RUS95] Russel, S.; Norvig, P.: Artificial Intelligence – A Modern Approach. Prentice Hall, (1995).

[SOW91] Sowa, J.: Principles of Semantic Networks. Morgan Kauffmann. Los Altos/Palo Alto/San Francisco (1991).

[SEF95] Stefik, M.: Introduction to Knowledge Systems. Morgan Kaufman, 1995

RDF

[RDF03a] RDF Primer, W3C Working Draft, 10/2003http://www.w3.org/TR/2003/WD-rdf-primer-20031010/

[RDF03b] RDF Vocabulary Description Language 1.0: RDF Schema, W3C Proposed Recommendation, 12/2003 http://www.w3.org/TR/2003/PR-rdf-schema-20031215/

[NFO03] NFO Business Intelligence. Monitoring Information Economics. 2003 http://193.202.26.196/bmwi_english/Faktenbericht_6/index.htm

[ROS03] Andre Rosin, "RDF-Schema", Institut für Informatik, Lehrstuhl für Datenbanken und H. Reinsch. Semantic Web, RDF-Schema. http://www.helge-reinsch.de/ki2/rdfschema.htm

[RIE03] Tammo Riedinger. Seminar Metamodelle im Vergleich. RDF-Schema vs. UML vs. Topic Maps. http://www.inf.fu-berlin.de/lehre/WS01/netbasedIS/WBIS6_4RDF.ppt

[XML_RDF] XML Design (A Gentle Transition from XML to RDF) http://www.xfront.com/rdf/