

Einführung in Wissensbasierte Systeme

(04.10.2003, version 1.1)

Table of Contents

1. Einführung.....	2
Was ist ein wissensbasiertes System (Expertensystem, intelligentes System)?.....	2
Warum wissensbasierte Systeme?.....	2
Anforderungen an ein wissensbasiertes System.....	2
Architektur von wissensbasierten Systemen.....	3
2. Problemlösen & Suchen.....	4
Uninformierte ("blinde") Suche.....	4
Heuristische Suche.....	5
3. Wissensrepräsentation.....	6
Anforderungen an Wissensrepräsentation (erwünschte Eigenschaften).....	6
Arten von Wissen.....	6
Prozedurale Methoden.....	7
Logikbasierte Methoden.....	7
Reifizierung.....	11
Strukturierte Objekte.....	11
Objektorientierte Methoden.....	11
Einfache Framesysteme.....	12
Komplexere Frames.....	12
4. Logik & Inferenz.....	14
Inferenz in regelorientierten Repräsentationen.....	14
Unifikation.....	15
Hornklauseln & deren Abarbeitung.....	15
Allgemeine Resolution.....	16
5. Spezielle Aspekte der Wissensrepräsentation.....	17
Anwendungsprobleme der klassischen Logik.....	17
Methoden rationalen Schließens.....	17
Prinzipien bei nichtmonotonen Logiken.....	17
Minimierungsmethoden.....	18
Default-Logik.....	18
Answer Sets.....	19
6. Unsicherheit.....	20
Elemente der Wahrscheinlichkeitstheorie.....	20
Probabilistisches Schließen	20
Bayes'sche Netze.....	20
Formalismen quantitativen Schließens.....	21
Fuzzy Logik.....	21
7. Anwendungsbereiche wissensbasierter Methoden.....	22
Diagnose.....	22
8. Planen.....	23
Situationskalkül.....	23
Strips.....	23

1. Einführung

Was ist ein wissensbasiertes System (Expertensystem, intelligentes System)?

Computersystem, das Probleme lösen kann, für die spezielles Wissen erforderlich ist. Weltwissen wird in deklarativer Form gespeichert (Wissensbasis) & verarbeitet (deklarativ: beschreibend in einer geeigneten Sprache).

Wichtigstes charakteristisches Merkmal: Trennung von Problemwissen & Wissensverarbeitung.

Warum wissensbasierte Systeme?

In einer Wissensbasis werden über Fakten hinaus auch Zusammenhänge von Fakten in expliziter Form dargestellt. Daraus kann durch Wissensverarbeitung (Interferenz) neues Wissen gewonnen werden. Zusammenhänge zwischen Fakten werden durch Regeln beschrieben.

Vorteile dieser Organisation: Flexibilität, Änderbarkeit & Erweiterbarkeit, Transparenz.

Anforderungen an ein wissensbasiertes System

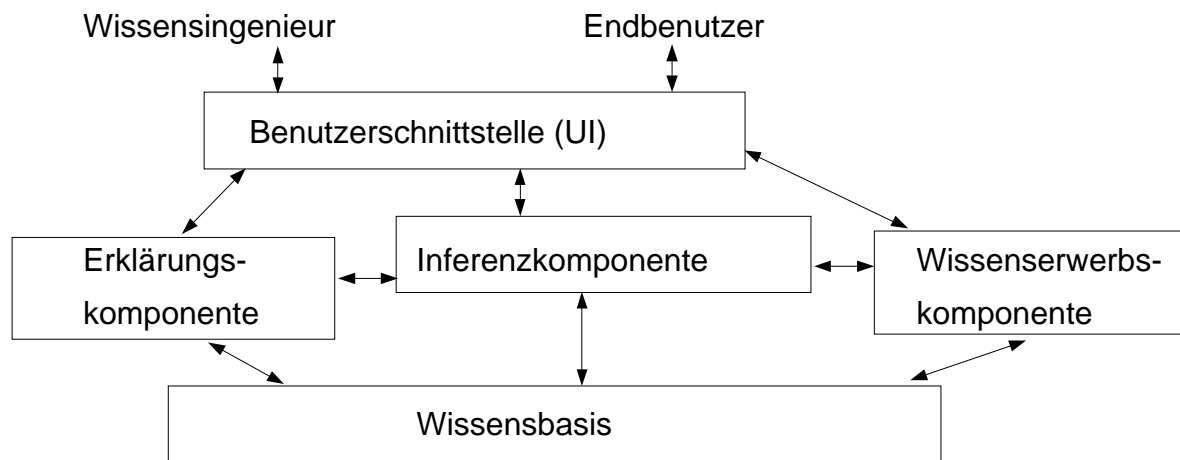
- Lösung von Problemen, die einen gewissen Anspruch haben
- Trennung von Problemwissen & Verarbeitung
- Speicherung & Organisation von Wissen
- Verständliche Wissensanzeige
- Transparenz der Lösungsfindung
- Erweiterbarkeit/Änderbarkeit für neues Wissen.

Kognitive AI: Wissensverarbeitung erfolgt nach Muster von Intelligenz. Voraussetzung: Theorie der Intelligenz.

Rationale AI: Ergebnisorientierte Wissensverarbeitung, Wissensverarbeitung muss nachvollziehbar & erklärbar sein.

Symbolverarbeitende Systeme: Basieren auf der Manipulation von Information & Wissen, das in symbolischer Form repräsentiert ist.

Subsymbolische Systeme: Signale das wesentliche Element für die Informationsverarbeitung.

Architektur von wissensbasierten Systemen

- **Wissensbasis:** Wissen ist hier in deklarativer Form abgelegt, Wissen umfasst Fakten & Regeln. Unterteilung in generisches (allgemeines Hintergrundwissen über Problembereich) & fallspezifisches (zur Bearbeitung eines konkreten Problems) Wissen.
- **Inferenzkomponente:** Das Wissen von der Wissensbasis wird hier verarbeitet, dabei wird neues Wissen aus Fakten & Regeln abgeleitet. Je größer die Wissensbasis desto komplexer. Erfordert methodisches Vorgehen.
- **Wissenserwerbskomponente:** Erweiterbarkeit. Zwei Möglichkeiten der Erweiterung:
 - Manuell: Wissensingenieur über entsprechende Schnittstelle (nicht Endbenutzerschnittstelle), auch Wartung. Ev. automatische Konsistenzprüfung.
 - Automatisch: System lernt aus Problemlösungen selbständig neues Wissen & erweitert die Wissensbasis.
- **User-Interface:** Schnittstelle nach außen, also zum Endbenutzer (Dialogkomponente) & zum Wissensingenieur.
- **Erklärungskomponente:** Auskunft über die Lösungsfindung, damit sie der Benutzer nachvollziehen kann.
2 Funktionen:
 - How-Funktion: Wie wird die Lösung gefunden (Schritte).
 - Why-Funktion: Warum werden bestimmte Schritte unternommen bzw. bestimmte Informationen verlangt.

2. Problemlösen & Suchen

Suchproblem ist definiert durch:

1. Startzustand.
2. Nichtleere Menge von Zielzuständen, die alle den Zieltest bestehen.
3. Nichtleere Menge von Operatoren, wobei durch Anwendungen von Operatoren auf einen gegebenen Zustand Nachfolgezustände generiert werden. Die dabei entstehenden Kanten haben positive Kosten > 0 .
4. Kostenfunktion für Pfade, die angibt, wie sich die Kosten eines Pfades aus den Kosten der Operatoranwendungen ergeben.

Eigenschaften von Suchverfahren:

- Vollständigkeit (Werden Lösungen gefunden?)
- Zeitkomplexität (Zeitverhalten bei wachsender Größe der Problemistanz?)
- Speicherkomplexität (Speicherbedarf bei wachsender Größe der Problemistanz?)
- Optimalität (Wird die beste Lösung gefunden, bezügl. der Kostenfunktion?)

Uninformierte ("blinde") Suche

Es werden keine problemspezifischen Informationen ausgenutzt, um Teile des Suchraums anderen Teilen vorzuziehen.

Breitensuche (breadth-first search)

Ebenenweise Expansion der Zustände. Zustand der Tiefe d wird erst expandiert, wenn alle Zustände der Tiefe $d-1$ expandiert wurden. Systematische Strategie.

Vollständig & optimal, wenn Pfadkostenfunktion streng monoton steigende Funktion der Tiefe. Speicherbedarf & Zeitkomplexität: b^d (b = Anzahl der Nachfolgezustände, d =Tiefe der Lösung), also exponentiell.

Uniform Cost Search

Modifikation der Breitensuche. Bevorzugt Zustände, die niedrige Kosten verursachen. Zuerst wird der Knoten mit den geringsten Kosten expandiert. Falls Lösung gefunden wird, werden die anderen Lösungspfade nur mehr solange verfolgt, bis die Kosten des Pfades die Kosten der günstigsten gefundenen Lösung übersteigen. Falls günstigere Lösung gefunden, wird sie weiter verglichen.

Vollständig & optimal, wenn Pfadkostenfunktion streng monoton steigende Funktion der Tiefe. Speicherbedarf & Zeitkomplexität: b^d (b = Anzahl der Nachfolgezustände, d =Tiefe der Lösung), also exponentiell.

Tiefensuche (depth-first search)

Generiert immer die Nachfolger eines tiefsten Zustands. Wenn alle Zustände der Tiefe d keine Nachfolger mehr besitzen, dann werden alternative Zustände der Tiefe $d-1$ expandiert. In immer größere Tiefen abtauchen, dann mittels Rücksetzen (backtracking) Alternativen suchen. Achtung: unendlich absteigende Pfade, auf denen keine Lösung

liegen => terminiert nicht!

Nicht vollständig & nicht optimal. Zeitbedarf: b^m , also exponentiell. Speicherbedarf gering: $b \cdot m$ (b = Verzweigungsgrad, m = maximale Pfadlänge).

Tiefensuche mit wachsender Tiefenschränke (depth-first search with iterative deepening)

Wie Tiefensuche, es wird aber eine globale Tiefenschränke eingeführt. Falls die Tiefenschränke erreicht wird, wird zurückgesetzt => damit terminiert die Tiefensuche immer. Aber: falls Schranke zu gering gewählt, wird keine Lösung gefunden! (=> dynamisch erzeugte Schranke)

Vollständig & optimal (=> Vorteile der Breitensuche). Speicherbedarf gering: $b \cdot d$ (=> Vorteil der Tiefensuche). Zeitkomplexität: b^d .

Heuristische Suche

Bestbewerteter Zustand wird zuerst expandiert, wobei die Bewertung auf Heuristiken basiert (best-first search).

Greedy Search

Schätzt Kosten vom aktuellen Knoten n zu einem Zielknoten ohne Berücksichtigung bisher angefallener Kosten.

Nicht optimal, ohne Vorkehrungen (wie zB Überprüfung auf Zyklen) ev. keine Termination => nicht vollständig. Zeit- & Speicherkomplexität: b^m (b = Verzweigungsgrad, m = Länge eines maximalen Suchpfades).

A*-Suche

Die tatsächlichen Kosten des Pfades vom Startknoten zum aktuellen Knoten gehen mit in die Auswahlentscheidung ein.

A*-Suche ist ein best-first Verfahren mit den Eigenschaften:

- Zustandsraum besitzt endlichen Verzweigungsgrad.
- Kosten jeder Kante sind positiv & > 0 .
- Heuristische Schätzfunktion muss zulässig sein (dh darf die tatsächlichen Kosten nie überschätzen).

Finden einer brauchbaren Schätzfunktion mittels Vereinfachung des zugrunde liegenden Modells.

Optimal & vollständig. Speicher- & Zeitkomplexität exponentiell.

3. Wissensrepräsentation

Anforderungen an Wissensrepräsentation (erwünschte Eigenschaften)

- Ausdrucksstärke: Formalismus muss hinreichend mächtig sein, damit das Wissen überhaupt dargestellt werden kann.
- Verarbeitbarkeit: Aus bestehendem Wissen muss auf systematische Weise neues Wissen gewonnen werden können. Plus ev. Automatisierbarkeit der Wissensverarbeitung & Schlüsse in endlicher Zeit möglich. Aber: polynomielle Verarbeitbarkeit steht normalerweise im Gegensatz zu großer Ausdrucksstärke.
- Flexibilität: Wissen aus verschiedenen Anwendungsbereichen. Spezifische Informationen sollen "effizient" dargestellt werden können.
- Modularität: Wissensbasis modular aufgebaut, also in Bereiche unterteilt.
- Verständlichkeit: Wissensingenieur & Benutzer müssen das dargestellte Wissen leicht erfassen können.
- Unvollständige Information: Muss oft dargestellt & verarbeitet werden.
- Erfassung unsicheren Wissens: Vage (ungefährer Sachverhalt) & unsichere (Wahrscheinlichkeit) Information. Es kann auch widersprüchliches Wissen vorliegen.

Arten von Wissen

- Faktenwissen: Fakten sind Beschreibung von elementaren Sachverhalten, zB Eigenschaften eines Objekts. Auch können Beziehungen zwischen Objekten beschrieben werden.
- Zusammenhänge (Beziehungen):
 - Statische: Regeln (zB "alle Fische können schwimmen") oder Bedingungen (zB "Ventil kann nicht gleichzeitig offen & zu sein").
 - Zeitliche: Entwicklung von Eigenschaften (zB Rechtsfähigkeit einer Person nach Alter).
 - Kausale: Ursache-Wirkung Beziehungen (zB Masern kann rote Punkte im Gesicht hervorrufen).
- Methodisches Wissen: Gibt Auskunft darüber, wie zur Problemlösung vorgegangen werden kann. Kann in Zerlegungsstrategien oder in Schrittabfolgen vorliegen.
- Meta-Wissen: Wissen über Wissensbasis, also Aufbau, Struktur & Verwendbarkeit des Wissens. Unterteilung in verschiedene Arten wie zB Faktenwissen, Beziehungsteil, methodisches Wissen (beschreibt hier die Eignung des in der Wissensbasis vorhandenen Wissens zur Problemlösung). Wird verwendet, um die Auswahl von Daten & Strategien zur konkreten Problemlösung zu steuern.

Methoden der Wissensrepräsentation: logik-basierte, prozedurale & objekt-orientierte Methoden.

Prozedurale Methoden

Wissen in Form von Prozeduren dargestellt. Aufruf der Prozedur ist von der Inferenzkomponente steuerbar. Explizite Berücksichtigung aller Lösungsmöglichkeiten notwendig.

Vorteile:

- Bei Formulierung der Prozedur kann explizit auf die Abfolge der Suche eingegriffen werden. => Suchalgorithmen können optimiert werden.

Nachteile:

- Lösungen können vergessen bzw übersehen werden.
- Mangelnde Modularität.
- Eingeschränkte Flexibilität.

Logikbasierte Methoden

Verwenden die Logik & Methoden der formalen Logik.

Komponenten:

- Syntax: Sätze müssen in einer Sprache vorliegen.
- Semantik: Bedeutung der Sätze.

Weitverbreitete Referenzsprache zur logischen Wissensrepräsentation ist die Prädikatenlogik (1. Stufe). Vor/Nachteile:

- + Wichtigster Formalismus der formalen Logik, viele theoretische Resultate.
- + Ausdrucksstark.
- + Einfache, natürliche Syntax & intuitive Semantik.
- Es können nicht berechenbare Funktionen spezifiziert werden.
- Rechenzeit zur Lösung von Problemen kann sehr hoch sein.

Aussagenlogik

Subformalismus der Prädikatenlogik. Gegenstand sind in sich geschlossene Aussagen, die mit wahr oder falsch bewertet werden können. Aussagen werden durch logische Verknüpfungen zu komplexeren Aussagen verbunden. Aussagen - Symbole, Verknüpfungen - Formeln.

Syntax: Ausgangspunkt ist Menge A von elementaren Aussagen (Atome).

Semantik: Wird durch Interpretationen & Modelle festgelegt.

Interpretation: Abbildung $I : A \rightarrow \{t, f\}$ die jedem Atom $a \in A$ einen Wahrheitswert $I(a)$ zuordnet. $\text{Int}(A)$ bezeichnet die Menge aller möglichen Interpretationen.

Modell: Eine Interpretation I ist ein Modell für eine Aussage φ ($I \models \varphi$), wenn $I(\varphi) = t$. $\text{Mod}(\varphi)$ bezeichnet die Menge aller Modelle von φ .

Arten von Aussagen:

- Tautologie: Wenn jede Interpretation ein Modell von \mathcal{I} ist ($\text{Mod}(\varphi) = \text{Int}(\mathcal{A})$).
- Kontradiktion: Widersprüchlich. Wenn keine Interpretation ein Modell von φ ist ($\text{Mod}(\varphi) = \emptyset$).
- Erfüllbare Aussage: Wenn sie ein Modell besitzt ($\text{Mod}(\varphi) \neq \emptyset$).

Logische Folgerung: Aussage ψ folgt logisch aus einer Aussage φ ($\varphi \models \psi$, ψ folgt logisch aus φ), wenn jedes Modell von φ ein Modell von ψ ist ($\text{Mod}(\varphi) \subseteq \text{Mod}(\psi)$).

Logische Äquivalenz: Zwei Aussagen φ & ψ sind logisch äquivalent, wenn $\varphi \models \psi$ & $\psi \models \varphi$ gilt.

Elementare Resultate: Doppelte Negation, Gesetz von De Morgan, Kontraposition, Implikation als Disjunktion, Implikation u. Äquivalenz, Kompaktheit, Deduktionstheorem, Widerspruchsbeweis, Modus Ponens, Fallanalyse.

Erfüllbarkeitsproblem: Es ist durch einen Algorithmus entscheidbar, ob eine gegebene Formel φ der Aussagenlogik erfüllbar ist. Es sind aber keine Algorithmen bekannt, die das Erfüllbarkeitsproblem in polynomieller Zeit lösen können (Erfüllbarkeitsproblem ist NP-vollständiges Problem).

Prädikatenlogik

Stärkere Ausdrucksmittel als bei Aussagenlogik vorhanden. Gegenstand sind Objekte & deren Eigenschaften sowie Beziehungen zueinander. Über logische Quantoren kann die Betrachtung einer Aussage für bestimmte Belegungen der Variablen ausgedrückt werden.

Syntax: Vokabular für Objekte, Eigenschaften & Beziehungen, logische Grundzeichen.

Vokabular:

- Konstantensymbole: ZB c . Damit wird ein bestimmtes Objekt angesprochen. Es muss festgelegt sein, welches Objekt durch c bezeichnet ist ($= c$ muss interpretiert werden).
- Funktionssymbole: ZB f mit Stelligkeit n . Funktion, die jeder "Eingabe" von Objekten o_1, \dots, o_n als "Ausgabe" genau ein Objekt $o = f(o_1, \dots, o_n)$ zuordnet. Muss interpretiert werden.
- Variablensymbole: ZB x . Objekt, das zur Auswertung einer Formel festgelegt werden muss. Darauf kann ein Quantor angewendet werden.
- Prädikatensymbole: Stelligkeit n , n -äres Prädikat. $n = 1$ bezeichnet eine Eigenschaft eines Objektes, $n > 1$ eine Beziehung zwischen Objekten. Muss interpretiert werden.
- Terme: Bezeichnen Objekte.
 - Jedes Konstantensymbol c & jedes Variablensymbol x ist ein Term.
 - Wenn t_1, \dots, t_n Terme sind & f ein n -stelliges Funktionssymbol $\Rightarrow f(t_1, \dots, t_n)$ ist ein Term.
- Atomare Formeln: P ist ein n -stelliges Prädikatensymbol, t_1, \dots, t_n sind Terme $\Rightarrow P(t_1, \dots, t_n)$ ist eine atomare Formel (Atom).

- Formeln:
 - Jede atomare Formel ist eine Formel in F .
 - φ_1 & φ_2 sind Formeln in F & x ist eine Variable \Rightarrow AND, OR, Implikation, Äquivalenz, NOT, Allquantifizierung, existentielle Quantifizierung sind ebenfalls Formeln in F .

Begriffe im Zusammenhang mit Formeln:

- Freie Variable: Vorkommen einer Variable x in einer Formel φ heißt frei, wenn x durch keinen Quantor gebunden ist. Und wenn x mindestens einmal in φ frei vorkommt.
- Satz: Geschlossene Formel. Formel ohne freie Variablen.
- Allquantifizierter Satz: Universieller Satz. Wenn nur Allquantoren vorkommen.
- Literal: Atomformel oder deren Negation.
- Klausel: Disjunktion (OR) von Literalen L_i .
- Horn-Klausel: Klausel, in der höchstens ein positives Literal vorkommt.
- Konjunktive Normalform (KNF): Wenn ein Satz aus Konjunktionen (AND) von Klauseln besteht.
- Klauselmenge: Klauselmenge einer KNF ist die Menge $\{K_i \mid 1 \leq i \leq m\}$ der Klauseln von φ .

Semantik: Wird über Interpretationen & Modelle definiert.

Interpretation: \mathcal{I} für ein Vokabular V hat die Komponenten:

- beliebigen nichtleeren Wertebereich D von Objekten
- Zuordnungen von Objekten, Funktionen & Relationen über D .

Modell, Tautologie, Kontradiktion, Erfüllbarkeit, Gültigkeit, logische Folgerung usw sind analog der Aussagenlogik definiert.

Elementare Resultate: Gleichen der Aussagenlogik. Und:

- $\forall x \varphi(x)$ logisch äquivalent zu $\neg \exists x \neg \varphi(x)$
- Spezialisierung: $\forall x \varphi(x) \models \varphi[x/t]$ (t variablenfreier Term)
- Existentialisierung: $\varphi[x/t] \models \exists x \varphi(x)$ (t variablenfreier Term)

Erfüllbarkeitsproblem: Probleme der Erfüllbarkeit & Gültigkeit nicht entscheidbar. Semi-entscheidbar, dh es gibt einen Algorithmus, der genau dann terminiert, wenn die Eingabe φ ein gültiger Satz ist.

Erweiterungen:

- Gleichheit (=): Vordefiniertes Prädikat.
- Uniqueness-Quantifier: "genau ein x " ($\exists!x$)
- Herbrand-Interpretation: Interpretationen, in denen Konstanten- & Funktionssymbole durch sich selbst interpretiert werden, dh. $\mathcal{I}(c) = c$. Verschiedene Konstantensymbole stehen für verschiedene Objekte.

Aspekte der logischen Wissensmodellierung

- **Axiome, Fakten & Regeln:** Die Beschreibung eines Problembereichs erfolgt über eine Menge von Sätzen, die wahre Sachverhalte im Problembereich ausdrücken. Solche Sätze nennt man auch Axiome, & die Menge der beschreibenden Sätze eine Theorie des Problembereichs (**domain theory**). Diese allgemeine Beschreibung wird zur Lösung von konkreten Problemen um fallspezifisches Wissen erweitert, das ebenfalls in Form von Sätzen (meist Fakten beschrieben ist). Fakten lassen sich als Literale repräsentieren.
- **Universe of Discourse:** In der Wissensrepräsentation, wo man Probleme aus konkreten Bereichen formalisieren will, ist in der Regel der Wertebereich eindeutig festgelegt (in der Prädikatenlogik kein spezieller Wertebereich). Dieser Wertebereich heißt universe of discourse. Auch hat man bei der Verwendung des Vokabulars für gewisse Elemente ev. eine fixe Interpretation vor Augen (zB geburts_jahr). Formal gesehen ist man nur an bestimmten Interpretationen der Theorie interessiert, in denen der Wertebereich D dem universe of discourse entspricht & bestimmte Symbole entsprechend interpretiert werden.
- **Unique Names Assumption (UNA):** Konstantensymbole sind im Kontext der Wissensrepräsentation oft Namen für Objekte (zB Tom). Intuitiv bezeichnen verschiedene Namen verschiedene Objekte; in einer beliebigen Interpretation \mathcal{I} können verschiedene Objekte jedoch gleich interpretiert werden. Somit ist der Satz $\text{Tom} \neq \text{Jerry}$ nicht gültig. Die verschiedene Interpretation von Konstantensymbolen $c_1, c_2, \dots, c_i, \dots$ kann durch Hinzunahme von Literalen $c_1 \neq c_2, \dots, c_i \neq c_j$, für alle $i \neq j$ sichergestellt werden. Dies wird als UNA bezeichnet. Unter UNA ist $\text{Tom} \neq \text{Jerry}$ gültig.
- **Domain Closure Axiom (DCA):** Der Wertebereich ist gelegentlich implizit durch jene Namen (bzw Konstantensymbole) c_1, c_2, \dots, c_n gegeben, die in einer Theorie vorkommen. Um bei Interpretationen weitere Objekte auszuschließen, kann der Wertebereich durch das DCA abgeschlossen werden: $\forall x(x = c_1 \vee x = c_2 \vee \dots \vee x = c_n)$. Dies setzt voraus, dass der Wertebereich endlich ist. Die Gültigkeit von Eigenschaften auf bestimmte Objekte kann ebenfalls eingeschränkt werden: $\forall x[P(x) \rightarrow (x = c_1 \vee x = c_2 \vee \dots \vee x = c_n)]$.

Methodologie der Modellierung

- **Klarheit von Namen, Konzepten:** Die Namen von Konstanten, Prädikaten usw sollten nach Möglichkeit sprechend sein. ZB elternteil(x,y).
- **Offenlegen von Zusammenhängen:** Die Verwendung von zu spezifischen Prädikaten, die komplexere Beziehungen unausgesprochen in sich kodieren & so der Verwendbarkeit bei der Wissensverarbeitung entziehen, soll vermieden werden.
- **Validierung:** Bei der Formulierung eines Satzes ist es ratsam zu hinterfragen, warum dieser Satz wahr ist.
- **Allgemeinheit:** Wie allgemein ist ein Satz, der einen Zusammenhang beschreibt? Kann die entsprechende Beziehung allgemeiner ausgedrückt werden?
- **Erfordernis von Prädikaten:** Ist ein neues Prädikat nötig? Wie steht es in Beziehung zu anderen Prädikaten? Falls das Prädikat eine Klasse von Objekten beschreibt, ist diese eine Ober/Unterklasse von anderen Klassen?

Mögliche Vorgehensweise:

1. Konzeptualisierung: Entscheidung, was repräsentiert werden soll.
2. Wahl des Vokabulars: Übersetzung der abstrakten Konzepte in die Sprache der Prädikatenlogik.
3. Kodierung der domain theory.
4. Für die Problemlösung: Kodierung des fallspezifischen Wissens.

Darstellung von Klassen & Objekten

Klassen von Objekten können in der Prädikatenlogik durch einstellige Prädikate beschrieben werden (zB $\text{fisch}(x)$). Beziehungen zwischen Klassen werden dann durch Sätze beschrieben. Die Subklassenbeziehung wird durch Implikation beschrieben (zB $\forall x(\text{fisch}(x) \rightarrow \text{tier}(x))$). Auf diese Weise können Hierarchien von Klassen, insbesondere Taxonomien von Objekten gebildet werden. In der Hierarchie werden Eigenschaften vererbt (zB $\forall x(\text{fisch}(x) \rightarrow \text{schwimmt}(x))$).

Reifizierung

Alternative zur Verwendung eines Prädikats. Die Klasse wird durch ein Objekt dargestellt. ZB statt $\text{vogel}(x)$: Konstantensymbol vogel & Mitgliedsbeziehung $\text{member}(t, \text{vogel})$. Subklassenbeziehung lautet dann $\forall x(\text{member}(x, \text{vogel}) \rightarrow \text{member}(x, \text{tier}))$.

Vorteile:

- Man kann über Eigenschaften & Beziehungen von Klassen sprechen, die nicht Eigenschaften bzw Beziehungen von Mitgliedern der Klasse sind (zB $\text{experte}(\text{Graham}, \text{vogel})$).
- Beziehungen zwischen Kategorien können mit entsprechendem Vokabular leicht ausgedrückt werden. Allgemeine Prädikate zur Beschreibung von Beziehungen können definiert & auf spezielle Fälle angewendet werden (zB $\text{disjunkt}(x)$ kann so beschrieben werden, dass mit dem Fakt $\text{disjunkt}(\text{tier})$ beschrieben wird, dass die Klassifikation der Tiere disjunkt ist).

Strukturierte Objekte

Strukturierte Objekte, die sich aus anderen Objekten zusammensetzen, müssen durch Prädikate beschrieben werden (zB $\text{teil_von}(x, y)$). Über weitere Prädikate kann beschrieben werden, in welcher Beziehung Teile eines Objektes zueinander stehen. ZB Fahrrad hat 1 Rahmen & 2 Räder.

Objektorientierte Methoden

Man geht von der Idee aus, dass die Struktur der Objekte bei der Wissensrepräsentation beibehalten wird. Die in der Konzeptualisierung ermittelten Objekte sollten auf möglichst natürliche Weise durch formale Objekte dargestellt werden.

Merkmale:

- Zentrale Beschreibung: Objektstruktur wird zentral beschrieben, dh die Beschreibung ist über das Objekt lokalisierbar. Mit dem Objekt selbst sind seine Eigenschaften (Attribute) im direkten Zusammenhang abgelegt.

- Deklarative Strukturbeschreibung:
 - Objektstruktur: Bei konkreten Objekten werden seine Eigenschaften explizit durch Attributwerte beschrieben.
 - Klassenstruktur: Gleichartige Objekte werden durch abstrahierende Beschreibungen erfaßt. Die Ähnlichkeit von Objekten geht aus deren Beschreibung hervor. Jedem Objekt wird ein Typ zugeordnet, so ist eine Menge von gleichartigen Objekten durch ihren Typ beschrieben.

Vererbung

- Subkonzept/Superkonzept-Vererbung: Objekte des Subkonzeptes werden durch die Eigenschaften des Superkonzeptes beschrieben (& weiterer Eigenschaften des Subkonzeptes).
- Instanzierungs-Vererbung: Instanzen eines Typs erben alle Eigenschaften, die ein prototypisches Objekt dieses Typs hat.
- Überdeckung: Explizite Redefinition einer Eigenschaft, die das Superkonzept hat, die für das Subkonzept aber nicht ganz zutrifft (zB Sitzplätze). Kann zu Inkonsistenzen führen (zB Pinguin ist ein Vogel, kann aber nicht fliegen).
- Mehrfachvererbung: Ein Subkonzept hat mehrere Superkonzepte. Problem, wenn dasselbe Attribut von mehreren Superkonzepten geerbt wird. Es muss dann entschieden werden, von welchem Superkonzept das Attribut geerbt wird. Ev. ist eine Vererbungsstrategie nötig (zB Depth-First mit Reihenfolge der Superkonzepte). Es können ebenfalls Inkonsistenzen auftreten (zB Vererbung widersprüchlicher Boolescher Attribute).

Framesysteme

Basieren auf dem objektorientierten Ansatz. Stereotype Situationen werden durch ein Gerüst bzw Rahmen (frame) beschrieben, das die üblichen, möglichen Eigenschaften der Situation beinhaltet & so eine abstrakte, generische Beschreibung darstellt, die ein Prototyp für eine Situation ist.

Einfache Framesysteme

Ein Frame ist ein Objekt, das durch die Objekt-ID angesprochen wird. Es hat eine Liste von Attributen, die Werte aus Wertebereichen annehmen können. Die Unterscheidung zwischen einem generischen & einem Instanz-Frame kann durch ein Attribut erfolgen (zB GENERIC). Die Erfassung der Instanzierungs- & Subkonzept/Superkonzept Beziehungen erfolgt ebenso durch Attribute (zB AKO - a kind of, INSTANCES).

Komplexere Frames

Es werden prozedurale & deklarative Elemente vermischt. Mechanismus des procedural attachment erlaubt, also Prozeduren in die Framestruktur einzuhängen, die nach einem Aktivierungsmuster (triggering) ausgeführt werden. ZB Frame Representation Language (FRL).

Semantische Netze

Wissen wird vereinfacht gesehen in Form eines gerichteten Graphen repräsentiert. Knoten sind Konzepte & stellen Objekte der realen Welt, Klassen, Behauptungen, Ereignisse usw. dar. Kanten sind die Beziehungen zwischen den Konzepten (konzeptuelle Relationen).

Unterschied semantische Netze & objektorientierte Ansätze:

- Bei den OO-Ansätzen steht die Unterordnung der Eigenschaften eines Objektes (bzw. Konzepts) unter das Objekt (bzw. Konzept) im Vordergrund. => Datenkapselung
- Bei semantischen Netzen werden Eigenschaften von Objekten (bzw. Konzepten) als gleichwertige Objekte (bzw. Konzepte) dargestellt & in Beziehung gesetzt. => Offenlegung der Beziehungen

Unterschied semantische Netze & logik-basierter Ansatz:

- Darstellung der Wissensinhalte in semantischen Netzen ist für Benutzer besser erfassbar & überschaubarer.
- Wissen um Vererbung wird in semantischen Netzen auf modulare Art erfaßt.
- Struktur eines semantischen Netzes erlaubt direkte Verarbeitung durch Algorithmen, die auf diesen operieren. Verarbeitung erfolgt durch Propagieren von Werten bzw. Markierungen von Knoten (marker propagation) durch das Netz, sowie durch pattern matching.
- Semantische Netze haben oft keine rein deklarative Semantik, sondern die Interpretation hängt von Abarbeitungsalgorithmen ab (zB bei Mehrfachvererbung, Überdeckung).

Relationale Graphen: Knoten sind atomare Konzepte, Kanten drücken Beziehungen zwischen den verbundenen Konzepten aus. ZB "Mann schüttelt lachend den Kopf" => Knoten: Mann, Schütteln, Kopf, Lachen.

Propositionale Netze: Sind ausdrucksstärker als relationale Graphen. Es können komplexere Sachverhalte (Aussagen) dargestellt werden, Netze können geschachtelt werden. ZB "Jane glaubt, dass John Henry gerufen hat" => Knoten: Jane, Glauben, John, Rufen, Henry.

Logische Formeln mit Quantoren: Oft treten implizit logische Quantoren auf. Darstellungsmöglichkeiten sind die Verwendung von Knoten zur Darstellung von Variablen & Quantoren oder die Verwendung spezieller Kanten für Quantoren. ZB "wenn Schlange giftig ist, dann ..." enthält einen impliziten Allquantor.

Vererbung: Erfolgt durch spezielle Kantentypen (zB ISA-Kante).

4. Logik & Inferenz

Methoden der Inferenz spielen in der Wissensverarbeitung eine zentrale Rolle.

Inferenz in regelorientierten Repräsentationen

Vorwärtsverkettende Systeme

Verarbeitung findet datengetrieben (data driven) statt, dh ausgehend von den Daten wird nach Wissen gesucht, das anwendbar ist. Nachteil: Ziel wird während der Abarbeitung nicht berücksichtigt.

Vorwärtsverkettung ist vorteilhaft, wenn in das System neu hinzukommende Daten (zB aktualisierte Meßwerte) Aktionen auslösen sollen.

Ein oft angewendeter Formalismus ist die Repräsentation des Wissens in Form von Regeln (bzw Produktionen). Ein solches Regel- oder Produktionensystem besteht aus dem Arbeitsspeicher (working memory, WM) & dem Regelspeicher (rule memory).

Elemente des Arbeitsspeichers: Fakten. Besteht aus einer Menge von Typen sowie deren Instanzen. Die eigentlichen Elemente des Arbeitsspeichers (working memory element, WME) sind konkrete Instanziierungen.

Elemente des Regelspeichers: Regeln kodieren zumeist Expertenwissen über Zusammenhänge. Form: *wenn* Bedingungen *dann* Aktionen. Mehrere Bedingungen werden konjunktiv verknüpft. Der Bedingungsteil heißt left hand side (LHS), der Aktionsteil right hand side (RHS).

Dynamisches Abarbeitungsmodell: Initiierung passiert durch Ausführen einer Regel mit leerer LHS. Die RHS dazu enthält Initialisierungen, zB WME erzeugen => Initialisierung des WM. Danach startet der Abarbeitungszyklus: Musterung -> Regelauswahl -> Aktion.

Abarbeitungszyklus (recognize-act-cycle):

1. Musterung (matching): Die Bedingungen aller Regeln werden untersucht, ob sie durch Elemente des WM erfüllt werden. Das Resultat der Musterung ist die Konfliktmenge (conflict set, CS), die Regelinstanzen enthält. Eine Regelinstanz ist eine Datenstruktur, die den Namen der Regel & alle gebundenen WMEs (jene, die die Bedingung der Regel erfüllen) enthält.
2. Regelauswahl: Eine Regelinstanz wird aus dem CS zur Verarbeitung ausgewählt. Alternativ können Auswahlstrategien angewendet werden. Merkmale zur Regelauswahl:
 - Verhinderung von Endlosschleifen durch Verhinderung mehrfacher Ausführung von Regeln mit gleichen Daten.
 - Auswahl aufgrund zeitlicher Bedingungen. Aktuelle (neuere) Information bevorzugen oder Zeitpunkt der Aufnahme der Regelinstanz ins CS berücksichtigen.
 - Auswahl aufgrund der syntaktischen Struktur der Regel.
 - Bevorzuge Regeln aufgrund von Meta-Wissen. Meta-Regeln oder Prioritäten.
 - Wähle zufällig.
3. Aktion: Die Aktion der ausgewählten Regel wird ausgeführt. Bewirkt zumeist Änderung des WM. Abarbeitungszyklus beginnt von vorne.

=> Vorwärtsverkettendes Verfahren: Anwendung von Regeln auf den Startzustand, der dadurch geändert & in einen "neuen" Startzustand überführt wird. Dieser dient dann als Ausgangspunkt für die nächste Runde der Regelverarbeitung. Abgebrochen wird, wenn der Zielzustand erreicht ist oder das CS nach Musterung leer ist.

Rückwärtsverkettende Systeme

Verarbeitung findet zielorientiert (goal driven) statt, dh ausgehend vom Gesamtziel wird nach Wissen gesucht, das anwendbar ist & das das Gesamtziel in eine oder mehrere einfachere Unterziele zerlegt.

Vorteile: Einbeziehung des zu erreichenden Ziels in Suchentscheidungen, Fokussierung auf relevantes Wissen.

Unifikation

Unifikationsprinzip: Methode zur Ermittlung des anwendbaren relevanten Wissens (ähnl. Musterung). Zielgerichtete Generierung von Instanzen.

Substitution: Abbildung $\sigma: V \text{ (Variablen)} \rightarrow T \text{ (Terme)}$. $\sigma(x) = t$.

Komposition von Substitutionen: $x(\sigma \mu) = \mu(\sigma(x))$.

Grundsubstitution: Wenn die Terme im Nachbereich von σ keine Variablen enthalten ($\sigma(x)$ ist variablenfrei).

Variablenumbenennung: Spezielle Substitution, Bijektion auf der Menge der Variablen.

Unifikator: Zwei Terme oder Literale E_1 & E_2 sind unifizierbar, wenn eine Substitution σ existiert, sodaß $E_1\sigma = E_2\sigma$. σ heißt dann Unifikator von E_1 & E_2 . Sie heißt allgemeinsten Unifikator (most general unifier, mgu), wenn sie allgemeiner ist als jeder andere Unifikator. Die Frage, ob ein allgemeinsten Unifikator existiert ist entscheidbar.

Hornklauseln & deren Abarbeitung

Hornklauseln können als Regeln interpretiert werden. $q \leftarrow p_1, \dots, p_n$ (Konjunktion von Atomen). Atom q ist der Kopf der Regel, die Konjunktion der Rumpf. Eine Einer-Hornklausel (Fakt; fact) ist eine Hornklausel mit leerem Rumpf ($q \leftarrow$). Eine Hornklausel, die nur aus einem Rumpf besteht ($\leftarrow p_1, \dots, p_n$) heißt Anfrage (query) bzw Ziel (goal). Die leere Klausel (\leftarrow , ohne Kopf & Rumpf) symbolisiert den Widerspruch. Der Oberbegriff für Fakten & Regeln ist definite Klausel (definite clause). Ein logisches Programm ist eine Menge von definiten Klauseln (ist erfüllbar).

Abarbeitung: Fakten bzw Regeln in Programmreihenfolge suchen, auf die die Anfrage angewendet werden kann.

1. Auswahl des als nächstes zu bearbeitenden Atoms in der Anfrage.
2. Auswahl der Regel bzw des Fakts (zuerst Fakten suchen, dann Regelköpfe untersuchen & ev. unifizieren)

=> SLD-Resolution (selection-rule driven linear resolution for definite clauses)

Korrektheit: Wenn der Widerspruch mittels SLD-Resolution abgeleitet werden kann, dann ist die Abfrage für das Programm unerfüllbar.

Widerlegungsvollständigkeit: Wenn die Abfrage für das Programm unerfüllbar ist, dann

kann der Widerspruch mittels SLD-Resolution abgeleitet werden.

Allgemeine Resolution

Resolutionskalkül: Arbeitet auf einer syntaktischen Unterklasse der Formeln der Prädikatenlogik 1. Stufe, den Klauseln.

Transformation geschlossener Formeln in KNF

1. Ermitteln freier Variablen. Umbenennen von gebundenen Variablen, sodaß jede Variable von genau einem Quantor gebunden wird.
2. Negieren der geschlossenen Gesamtformel.
3. Äquivalenzen eliminieren. $A \leftrightarrow B = (A \rightarrow B) \wedge (B \rightarrow A)$.
4. Implikationen eliminieren. $A \rightarrow B = \neg A \vee B$.
5. NOT mittels Ersetzungen vor Atome bringen.
6. Doppelte Negation eliminieren. $\neg\neg A = A$.
7. Existenzquantoren eliminieren (Skolemisierung) & Allquantoren ganz nach vorne.
8. KNF erstellen. $A \vee (B \wedge C) = (A \vee B) \wedge (A \vee C)$.

Ursprungsformel & erzeugte KNF sind nicht logisch äquivalent, sondern erfüllungsäquivalent.

Resolutionskalkül

Besteht aus 2 Regeln.

Resolution in der Aussagenlogik:

Resolutionskalkül R_p mit den Inferenzregeln Resolution & Faktorisierung.

$$\begin{array}{ccc} \begin{array}{c} L \vee C_1 \\ \hline C_1 \vee C_2 \end{array} & \text{Res} & \text{und} & \begin{array}{c} L \vee L \vee C_1 \\ \hline L \vee C_1 \end{array} \\ & & & \text{Fak} \end{array}$$

Resolution in der Prädikatenlogik:

Resolutionskalkül R mit den Inferenzregeln Resolution & Faktorisierung.

$$\begin{array}{ccc} \begin{array}{c} L \vee C_1 \\ \hline C_1\sigma \vee C_2\sigma \end{array} & \text{Res} & \text{und} & \begin{array}{c} L \vee K \vee C_1 \\ \hline L\mu \vee C_1\mu \end{array} \\ & & & \text{Fak} \end{array}$$

Beachte:

- In Res dürfen die beiden Elternklauseln keine gemeinsamen Variablen enthalten.
- In Res ist σ der mgu der Atome L & K .
- In Fak ist μ der mgu der Literale L & K .

Lineare Resolution: Mit Einschränkung, dass in jedem Resolutionsschritt mindestens 1 Elternklausel eine (ev. umbenannte) Klausel aus der Eingabeklauselmeng ist.

5. Spezielle Aspekte der Wissensrepräsentation

Anwendungsprobleme der klassischen Logik

- Sprache: Sprache kann prinzipiell nicht all das ausdrücken, was wir über die Welt sagen wollen. Eine endliche Menge von Sätzen ist immer nur eine Approximation der realen Gegebenheiten.
- Vorläufige Aussagen: Klassische Logik ist korrekt im Sinne, dass alle herleitbaren Sätze semantisch gültig sind. Herleitungen aus einer gegebenen Wissensbasis können nie neues Wissen generieren, sondern alles ist bereits implizit in der Wissensbasis vorhanden. Es sind aber Mechanismen nötig, die im Fall von unvollständiger Information vorläufige Aussagen (Annahmen) treffen, um weiteres Wissen abzuleiten. Die Annahmen können später bestätigt oder revidiert werden. Klassische Logik ist monoton, also benötigt man Methoden, die vorläufige Aussagen treffen können.
- Wahrscheinlichkeit: Die Aussagen können nur entweder ganz wahr oder ganz falsch sein. Oft ist die Wahrscheinlichkeit wichtig/notwendig.

Für diese Probleme werden Formalismen benötigt, die flexibel auf sich ändernde Situationen (zB Wissenszuwachs) ausgelegt sind. Ein wichtiger Mechanismus ist die Möglichkeit, inkorrekte Schlüsse zu erlauben (Verfahren, die Aussagen herleiten, die plausibel, aber nicht notwendigerweise logisch herleitbar sind).

Methoden rationalen Schließens

2 Klassen von Formalismen für das Schließen bei unvollständiger Information:

- Quantitative Methoden: Basieren auf numerischen Verfahren. Verwenden meist Begriffe der Wahrscheinlichkeitstheorie. ZB Bayes'sche Netze, Überzeugungs-Theorien (Unwissen & Unsicherheit), Fuzzy Mengen.
- Qualitative Methoden: Versuchen Unsicherheit & Unwissen ohne Bezug zu jedweder numerischer Methodik zu beschreiben. Plausible Schlüsse: erfüllen nicht das Prinzip der Monotonie (ein Schluß kann für einen gegebenen Wissensstand plausibel sein, sich aber bei Erhalt genauerer Information als trügerisch herausstellen). => nichtmonotone Logiken.

Prinzipien bei nichtmonotonen Logiken

Qualifikationsproblem

Unmöglich, sämtliches Wissen über die reale Welt durch eine endliche Menge von Fakten beschreiben zu wollen. Jeder Begriff, der für einen bestimmten Zweck verwendet wird, ist potentiellen Änderungen ausgesetzt. ZB Vögel, die gerade geschlüpft sind, können nicht fliegen. Notwendig sind Schlussverfahren, die "temporäre" Annahmen treffen können (Default-Annahmen), die später ev. revidiert werden können, sobald zusätzliche Qualifikationen relevant werden.

Frame-Problem

Frage, welche Eigenschaften sich beim Ausführen einer Aktion ändern & welche gleich bleiben. Eine Aktion beeinflusst nur einen geringen Teil der betrachteten Welt. ZB Würfel

bemalen, alles bleibt gleich bis auf die Farbe. Um Invarianzeigenschaften darzustellen, benötigt man eine große Anzahl von speziellen Zusatzaxiomen (Frame-Axiome), was eine ineffiziente Methode ist.

Ramifikationsproblem: Eng mit dem Frame-Problem verwandt. Beschreibt implizite Folgerungen aus einer durchgeführten Aktion. ZB staubigen Würfel hochheben, Staub kommt mit, aber nicht extra spezifizieren müssen. ZB Lincoln, wo rechter Fuß begraben?

Minimierungsmethoden

Closed-World Assumption (CWA)

Methode, um negative Fakten effizient zu repräsentieren. Von der Beobachtung, dass die Zahl der negativen Fakten eines bestimmten Problembereichs meist weit größer als die Zahl der positiven Fakten ist. Es wird nur positive Information gespeichert, alle Fakten, die nicht daraus hergeleitet werden können gelten als falsch. ZB Zugfahrplan. CWA ist nichtmonoton, da durch Hinzunahme neuer Fakten die Menge der herleitbaren Fakten ev. kleiner wird. CWA einer Theorie kann inkonsistent u/od nicht vollständig sein. Vollständig heißt, dass entweder die geschlossene Atomformel P oder $\neg P$ in $CWA(T)$ ist.

Prädikatenvervollständigung

Man kann oft mH einer einzigen logischen Formel zum Ausdruck bringen, dass jene Objekte, von denen man zeigen kann, dass sie ein gewisses Prädikat erfüllen, die einzigen sind, die dieses Prädikat erfüllen.

Vervollständigungsformel: Umkehrung der Theorie $\forall x(x = a \rightarrow p(x))$, um die explizite Information über p in der Theorie vollständig zu machen, in $\forall x(p(x) \rightarrow x = a)$.

Vervollständigung von p ist dann $COMP(T; p) \equiv \forall x(p(x) \leftrightarrow x = a)$.

Eine Menge von Klauseln ist solitär in P , wenn jede Klausel mit einem positiven Vorkommen von P höchstens ein Vorkommen von P hat. Prädikatenvervollständigung wird nur für Klauseln definiert, die solitär sind.

Default-Logik

Nichtmonotone Schlüsse werden mittels spezieller Inferenzregeln, den Defaults, realisiert.

Default ist ein Schlußschema der Form

$$\frac{\varphi : \psi_1, \dots, \psi_n}{\gamma}, \quad n \geq 0.$$

$\varphi, \psi_1, \dots, \psi_n, \gamma$ sind prädikatenlogische Formeln. φ ... Vorbedingung, ψ_1, \dots, ψ_n ... Rechtfertigungen, γ ... Konsequenz des Defaults.

ZB Vogel(x) : fliegt(x) \Rightarrow fliegt(x).

Wissen wird in der Form von Default-Theorien repräsentiert. Das sind geordnete Paare der Form $\langle W, D \rangle$, wo W eine Menge von geschlossenen Formeln & D eine Menge von Defaults ist. Menge W stellt das sichere Wissen des beschriebenen Bereichs dar, das Hintergrundwissen der Default Theorie.

Answer Sets

Wegen der Komplexität von Default Logik im Vergleich zu klassischer Logik hat man versucht, syntaktische Fragmente zur Realisierung von Default-Beweisern heranzuziehen. Es wurden verschiedene deklarative Semantiken für logische Programme entwickelt, die ermöglichen, Default Negation im Sinne von negation-as-failure auszudrücken. Die wichtigste dieser Semantiken ist die Answer Set Semantik für erweiterte logische Programme.

Erweitertes logisches Programm P: Menge von Regeln der Gestalt

$L_0 \leftarrow L_1, \dots, L_m, \text{not } L_{m+1}, \dots, \text{not } L_n.$

L_i ... Literal (Atom oder Negation eines Atoms). "Not" bezeichnet negation-as-failure oder Default Negation.

Dh wenn L_1, \dots, L_m hergeleitet werden können & L_{m+1}, \dots, L_n nicht hergeleitet werden können, dann kann man auf L_0 schließen.

Answer Set Semantik: Die Regeln aus P werden als Defaults interpretiert.

$$\frac{L_1 \wedge \dots \wedge L_m : \neg L_{m+1}, \dots, \neg L_n}{L_0}$$

Die Extensionen der so aus dem Programm P erhaltenen Default Theorie bestimmen dann die Answer Sets von P.

6. Unsicherheit

Elemente der Wahrscheinlichkeitstheorie

Zufallsvariablen: V_1, \dots, V_n . Werte der Zufallsvariablen sind v_i .

Domänen: Boole'sche Werte, numerische Werte, kategoriale Werte.

Wahrscheinlichkeitsverteilung: $P(V_i = v_i)$.

Aussagen haben einen Wahrscheinlichkeitswert zwischen 0 & 1.

A priori (marginale) Wahrscheinlichkeit:

$P(V_i = v_i) = \text{Summe über alle } V_i = v_i \text{ von } P(V_1, \dots, V_n)$.

Bedingte Wahrscheinlichkeit: Wahrscheinlichkeit von Größen unter der Voraussetzung, dass andere Größen bestimmte Werte annehmen bzw die Werte schon bekannt sind.

Berechnungen auf Basis der bedingten Wahrscheinlichkeit werden unter dem Begriff "probabilistisches Schließen" zusammengefaßt.

Bedingte Wahrscheinlichkeitsfunktion: $P(V_i | V_j) = P(V_i, V_j) / P(V_j)$.

Kettenregel: $P(V_1, \dots, V_n) = \text{Produkt über } i \text{ von } 1 \text{ bis } n \text{ von } P(V_i | V_{i-1}, \dots, V_n)$.

Probabilistisches Schließen

Eine Zufallsvariable V von einer Menge V_i von Zufallsvariablen bei vorhandener Evidenz $E = e$ ist bedingt unabhängig, wenn $P(V | V_i, E = e) = P(V | E = e)$.

Bedingte Unabhängigkeit: $\mathcal{I}(V, V_i | E = e)$ bzw $\mathcal{I}(V, V_i | E)$: Wenn V von V_i bei beliebiger Evidenz E unabhängig ist. Diese Eigenschaft ist eine Beziehung zwischen Wahrscheinlichkeitsverteilungen, dh eine Relation zwischen Funktionen.

Bayes'sche Netze

Gerichteter, azyklischer Graph (directed acyclic graph), dessen Knoten mit Zufallsvariablen markiert sind & der in graphischer Weise gewisse Unabhängigkeitseigenschaften von Variablen repräsentiert. 2 Knoten V_1 & V_2 sind mit einem Pfeil von V_1 nach V_2 verbunden, wenn V_1 "direkten Einfluß" auf V_2 hat. Zu jedem Knoten V ist die bedingte Wahrscheinlichkeit $P(V | V_o)$ bekannt, wo V_o die unmittelbaren Vorgänger, dh die Elternknoten, von V sind.

Für eine komplette Spezifikation eines Bayes'schen Netzes sind sämtliche bedingte Wahrscheinlichkeiten aller Knoten in Abhängigkeit der jeweiligen Elternknoten notwendig.

Vor- & Nachteile:

- Große Datenmengen zur Angabe der erforderlichen marginalen & bedingten Wahrscheinlichkeiten notwendig.
- Inwieweit die Voraussetzungen zutreffen, dass gewisse Variablen tatsächlich unabhängig sind, kann iA schwer gesagt werden.
- Es kann kein Nicht-Wissen dargestellt werden.
- Widersprüchliche Informationen werden nicht entdeckt, sondern pflanzen sich weiter fort.

Formalismen quantitativen Schließens

Certainty Factors

Ansatz, um unsicheres Schließen mittels quantitativen Methoden in regelbasierten Systemen zu ermöglichen. Aussagen haben einen certainty factor zwischen -1 & +1 (nicht wie bei Wahrscheinlichkeitstheorie zwischen 0 & 1). -1 bedeutet, dass die entsprechende Aussage als falsch angesehen wird, bei +1 als richtig.

Certainty factor einer Konjunktion ist das Minimum der einzelnen Konjunkte A_i .

Ad hoc Ansatz, der durch geschicktes "Tuning" gute Resultate liefern kann, aber es kann auch zu Fehlaussagen kommen (wenn zu viele Aussagen hergeleitet werden können). Überholter Ansatz.

Dempster-Shafer Theorie

Arbeitet mit einem unvollständigen probabilistischen Modell, das keine definitiven Wahrscheinlichkeitswerte berechnet, sondern Intervalle angibt. Behandelt Methoden, um mit dem Unterschied von Unsicherheit & Unwissen umzugehen.

Fuzzy Logik

Beschreibt Vagheit. Objekte können einen beliebigen Wert aus dem Intervall $[0, 1]$ annehmen. Fuzzy Menge S ist eine Funktion $\mu_S: X \rightarrow [0, 1]$.

Für relativ kleine Regelbasen mit kurzen Kausalketten gut geeignet, bei komplexen Aufgabenstellungen aber Probleme.

7. Anwendungsbereiche wissensbasierter Methoden

Diagnose

Analytische Aufgabe. Es wird versucht, fehlerhafte Komponenten in einem nicht mehr ordnungsgemäß funktionierenden System zu erkennen. 2 Ansätze: regelbasierte & modellbasierte Diagnosemethoden. Regelbasierte Systeme sind schlechter zu warten, da die Regeln voneinander abhängen.

Bei der modellbasierten Diagnose hat man wieder eine Trennung zwischen Systembeschreibung & Lösungsfindung. Das korrekte Verhalten des Systems wird beschrieben; Fehler werden durch den Vergleich zwischen vorhergesagtem korrektem & tatsächlich beobachteten Verhalten gefunden.

Vorteil der modellbasierten Diagnose:

- Unabhängigkeit eines Diagnosesystems von spezifischen Bauteilen.
- Leichte Erweiterbarkeit.

Komponenten eines modellbasierten Diagnosesystems

- Diagnosemodell: Dient zur Beschreibung des jeweiligen Systems. Definiert über eine Menge von Bauteilen, einer Menge von Beobachtungen & einer Menge von Korrektheitsannahmen der jeweiligen Komponenten. Anzugeben sind die vorhandenen Bauteile des Systems, ihre Verbindungen & ihr Verhalten.
- Diagnosealgorithmus: Aufgabe ist es, entsprechend den durchgeführten Beobachtungen Mengen von fehlerhaften Bauteilen zu ermitteln. Eine Menge der als fehlerhaft erkannten bzw angenommenen Bauteile wird als Diagnose bezeichnet.
- Prozedur zur Auswahl von optimalen Messpunkten (Messpunktselektion): Soll die Messpunkte so bestimmen, dass mit möglichst wenigen Messpunkten eine eindeutige Diagnose gefunden werden kann. Jeder Messpunkt erweitert die Systembeschreibung & die Komplexität des Diagnoseproblems, da für jeden Messpunkt der Diagnosealgorithmus aufgerufen werden muss.

Diagnoseproblem

Besteht aus:

- Systembeschreibung (logische Theorie, SD)
- Beobachtungen (Menge von Fakten, OBS)
- Bauteilen (Menge von Konstantensymbolen, COMP).

Eine Diagnose für das Diagnoseproblem (SD, OBS, COMP) ist eine minimale Menge Δ von fehlerhaften Komponenten, sodass die Menge von Formeln $SD \cup OBS \cup \{\neg ok(c) \mid c \in \Delta\} \cup \{ok(c) \mid c \in COMP \setminus \Delta\}$ erfüllbar ist.
 $c \dots$ Bauteil, $ok(c) \dots$ Bauteil funktioniert korrekt.

8. Planen

Planungsproblem besteht darin, ein gewünschtes Ziel durch Ausführung von mehreren Aktionen zu erreichen. Jede dieser Aktionen kann durch einen Stimulus aktiviert werden & tätigt einen Effekt auf die "Welt" des Agenten, der die Aktion ausführt. Die Effekte müssen vor der Ausführung bedacht werden & auf "Unbedenklichkeit" hin überprüft werden (zB 1 Schritt nach vor gehen => ev. in einen Abgrund fallen). Die Erstellung eines geeigneten Plans ist iA nicht trivial.

Für ein formales Planungssystem sind erforderlich:

- ein Modell der "Welt"
- ein Modell für Aktionen & deren Effekte.

Situationskalkül

Konzepte:

- Zustände: Zustände des Systems werden als eigene Objekte betrachtet & durch Objektkonstanten repräsentiert. Menge aller Zustände: States. Auch in der beschriebenen Welt vorkommende reale oder virtuelle Objekte in der Formalisierung als Objekte repräsentiert.
- Fluents: Sind Prädikate, die Aussagen über Zustände des Systems machen. Aussagen können spezieller oder allgemeiner Natur sein. Fluents können zueinander in Beziehung stehen.
- Aktionen: Werden als Funktionen modelliert, die auf Objekten ausgeführt werden. Menge aller Aktionen: Actions.
- "do" Funktion: Spezielle Funktion $do: Actions \times States \rightarrow States$. Weist einem Paar aus einer Aktion a & einem Zustand S einen Zustand S' zu.

Probleme:

- Mangelnde Effizienz: Auch bei kleinen Planungsaufgaben ist der Beweisaufwand sehr groß (wegen komplexen Systembeschreibungen).
- "Representational Frame Problem": Zahl der Frameaxiome proliferiert mit der Zahl der Aktionen & der Fluents & bläht die Systembeschreibung auf. Dazu kommt, dass im Endeffekt alle Fluents vom momentanen Zustand in den Nachfolgezustand explizit übernommen werden müssen, auch wenn sich die Fluents nie ändern.
- Qualifikationsproblem.
- Ramifikationsproblem.

Strips

Verwendet logische Formeln für die Beschreibung von Planungsszenarien, die Effekte der Ausführung von Aktionen werden durch eine operationale Definition festgelegt. => Dieser Nachteil des Situationskalküls (wegen verwendeter Prädikatenlogik) ist bei Strips nicht vorhanden.

Elemente:

- Zustandsbeschreibung (state description)
- Aktion
- Goal.

Zustandsbeschreibung (state description)

Besteht aus einer Menge S von variablenfreien Literalen (ground literals) der Prädikatenlogik.

Aktion

Führt einen Zustand in einen anderen über. Jede Aktion wird durch einen eigenen Operator spezifiziert. Die Ausführung der Aktion wird durch Anwendung des Operators modelliert.

Operator: $op: StateDescr \rightarrow StateDescr$. $StateDescr \dots$ Menge aller Zustandsbeschreibungen. Anwendung des Operators ist nicht für jede state description definiert.

Komponenten:

- Precondition - $PC(op)$: Vorbedingung, Menge von variablenfreien Literalen (Konjunktion).
- Add List - $Add(op)$: Add-Liste, Menge von variablenfreien Literalen.
- Delete List - $Del(op)$: Delete-Liste, Menge von variablenfreien Literalen.

Resultat der Anwendung des Operators: Wenn die Vorbedingung erfüllt ist, dann werden Literale der Delete-Liste aus S entfernt & die aus der Add-Liste zu S dazugegeben ($op(S) = (S \setminus Del(op)) \cup Add(op)$); sonst undefiniert.

Trägheitsannahme (inertia assumption): Strips trifft bei der Ausführung einer Aktion diese Annahme. Dh der Status eines Literals, das nicht explizit durch Anwendung eines Operator verändert wird, bleibt bezüglich des Systemzustands unverändert. => Das Frame-Problem wird automatisch gelöst.

Einzelaktionen können durch Parameter schematisch beschrieben werden (zB $move(B, A, Floor)$). Variablen werden benützt, um Parameter zu beschreiben.

Operatorschema (Strips Regel):

- Name: $op(x_1, \dots, x_n)$. $V = x_1, \dots, x_n$ sind Variablen.
- Precondition - $PC(op)$: Menge von Literalen (Konjunktion). Alle vorkommenden Variablen sind aus V .
- Add-Liste - $Add(op)$: Menge von Literalen. Alle vorkommenden Variablen sind aus V .
- Delete-Liste - $Del(op)$: Menge von Literalen. Alle vorkommenden Variablen sind aus V .

Goal

Zielzustand, der durch einen Plan erreicht werden soll. Goal ist eine Konjunktion $\gamma = L_1 \wedge \dots \wedge L_n$ von Literalen L_1, \dots, L_n .

Plan

Plan für eine Zustandsbeschreibung S & ein Goal γ ist eine Folge $P = \alpha_1, \dots, \alpha_n$ von Aktionen, repräsentiert durch Strips Operatoren op_1, \dots, op_n , sodass

- die Anwendung der Aktionen α_i in der Reihenfolge des Plans in jedem Schritt definiert ist
- in der resultierenden StateDescr das Goal erfüllt ist.

Progressives Planen in Strips

Planen durch Suche nach einer Vorwärtsstrategie. Es wird ausgehend von S ein Plan P schrittweise über die Suche einer state description S' im konzeptuellen Suchgraphen konstruiert, die das Goal γ erfüllt. Suche selbst kann nach verschiedenen Methoden durchgeführt werden (zB breadth-first, depth-first, A^*).

Bei großer Zahl von Strips Regeln & großen Zustandsbeschreibungen aber nicht praktikabel (mangelnde Effizienz wegen zu vieler möglicher Verzweigungen). => Fokussierte Suche nötig.

Im Fall mit Variablen wesentlich einfacher als regressiver Ansatz (Behandlung der Variablen beim regressiven Ansatz kompliziert & aufwändig).

Regressives Planen in Strips

Rückwärtssuche, die ausgehend vom Goal γ in Richtung der anfänglichen state description S einen Plan P in umgekehrter Reihenfolge konstruiert.

Dieser Ansatz ist zur Lösung praktischer Probleme besser geeignet, da das Goal in der Regel klein ist & für die Erfüllung eines Subgoals wenige Operatoren vorhanden sind.