

Ausgearbeitete Prüfungsfragen „Einführung in wissensbasierte Systeme.

1.1 Komponenten eines wissensbasierten Systems:

Wissensbasis:

Wissen ist hier in deklarativer Form abgelegt, Wissen umfasst Fakten und Regeln, generisches (Problemlösungsfall) bzw. fallspezifisches (Problemereich) Wissen

Interferenzkomponente:

Das Wissen von der Wissensbasis wird hier verarbeitet, dabei werden neue Fakten und Regeln abgeleitet, je größer die Wissensbasis desto komplexer

* Verarbeitet Wissen aus der Wissensbasis.

* Leitet Wissen aus Fakten und Regeln ab

Wissenserwerbskomponente:

Zwei Möglichkeiten der Erweiterung:

Manuell: Wissensingenieur über entsprechende Schnittstelle(nicht gleich Endbenutzer-Schnittstelle)

Automatisch: System lernt aus Problemlösungen selbstständig neues Wissen und erweitert die Wissensbasis

User-Interface:

Schnittstelle zum Endbenutzer und Wissensingenieur,

Erklärungskomponente:

Erteilt dem Benutzer 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 wurden

1.2 Zusammenspiel der Komponenten + Grafik + Benutzer:

Scan 1.13

2.1 Definition eines Suchproblems, einzelne Elemente erläutern

Suchproblem ist definiert durch:

1. einen Startzustand
2. eine nichtleere Menge von Zielzuständen(, die alle den Zieltest bestehen)
3. eine 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. eine Kostenfunktion für Pfade, die angibt, wie sich die Kosten eines Pfades aus den Kosten der Operatorenanwendungen ergeben

2.2 Nach welchen Eigenschaften kann man heuristische Suchverfahren klassifizieren? + Erläuterung der Eigenschaften

Eigenschaften von Suchverfahren:

- Vollständigkeit (werden alle Lösungen gefunden)
- Zeitkomplexität (Zeitverhalten bei wachsender Größe der Problemistanz)
- Speicherkomplexität (Speicherbedarfswachstum bei wachsender Größe der Problemistanz)
- Optimalität: (wird die beste Lösung gefunden, bez. der Kostenfunktion)

2.3 Eigenschaften für die in der VO behandelten heuristischen Suchverfahren

Greedy Search:

Charakteristisch für Greedy Search ist das Schätzen der Kosten vom aktuellen Knoten n zu einem Zielknoten ohne Berücksichtigung bisher angefallener Kosten.

- Greedy Search ist nicht optimal und nicht vollständig.
- Speicher und Zeit: b^m .
- Entstehung von Zyklen
- in einem lokalen Minimum kann es stecken bleiben od. zw. zwei hin und her wechseln

A* Suche:

Die tatsächlichen Kosten vom Start zum aktuellen Knoten gehen mit in die Entscheidung ein. Dadurch wird erreicht, dass Knoten nicht unendlich lange von der Expansion ausgeschlossen werden.

A* Suchen haben folgende Eigenschaften:

- Der Zustandsraum besitzt einen endlichen Verzweigungsgrad.
- Die Kosten JEDER Kante sind positiv und > 0 .

A* Suche ist optimal und vollständig.

Speicher und Zeit: exponentiell.

2.4 Berechnung von $f(n)$ für die in der VO behandelten heuristischen Suchverfahren:

Greedy: $f(n) = h(n)$

A*: $f(n) = g(n) + h(n)$

3.1 Regel-Produktionssystem (OPS 5): Elemente eines vorwärtsverketteten Systems, synthaktische Struktur, Operationen auf den Elementen

Regel- und Produktionssysteme bestehen auf 2 Elementen:

- Arbeitsspeicher
- Regelspeicher

Elemente des Arbeitsspeichers:

Das WM besteht aus einer Menge von Typen sowie deren Instanzen => WME (working memory element)

verschiedene Sichtweisen: WME entspricht:

- den „record“ Deklarationen in prozeduralen Programmiersprachen
- einer frameartige Repräsentation mit Namen und Wertfacette
- einem Tupel einer Datenbankrelation

Synthaktische Struktur:

(Person

↑name	Jonas
↑alter	5
↑strasse	Alhonrg 5
↑ort	irgendwo)

Operatoren auf den Elementen:

es gibt 3 Operationen auf WME's:

make, modify, remove

3.2 Wann ist eine Regel anwendbar

Eine regel ist anwendbar, wenn die Konjunktion ihrer Vorbedingungen erfüllt ist.

3.3 Schaubild vom Steuerungsalgorithmus, Wie werden Regeln ausgewählt

Scan 4.5

- Regelauswahl => Regelinstanz wird aus der Konfliktmenge ausgewählt zur Verarbeitung. Dabei gibt es mehrere Merkmale die zur Regelauswahl herangezogen werden:
 - Verhinderung mehrfacher Ausführung gleicher Regeln
 - Auswahl aufgrund zeitlicher Bedingungen
 - Auswahl aufgrund syntaktischer Struktur
 - Bevorzugte Regeln aufgrund von Meta Wissen.
 - Zufällig Auswahl

4.1 Was ist CWA formal:

Sei T eine Theorie (dh eine Menge von geschlossenen Formeln) und $\text{cons}(T)$ der logische Abschluss von T . Wir definieren:

- 1.) $T_{asm} := \{ \neg P \vee P \mid P \text{ ist eine geschlossene Atomformel und } P \notin \text{cons}(T) \}$;
- 2.) $CWA(T) := \text{cons}(T \cup T_{asm})$

4.2 Was ist eine vollständige Theorie, ist $CWA(T)$ vollständig?

Man bezeichnet eine Theorie T als vollständig, wenn für jede geschlossene Atomformel P gilt, dass P oder $\neg P$ in T liegt. Somit ist die CWA eine vollständige Theorie, da für jede geschlossene Atomformel P , die nicht in $CWA(T)$ liegt, $\neg P \in CWA(T)$ gilt ($CWA(T)$ vervollständigt die Theorie durch Hinzunahme von negativen Fakten).

4.3 $T = \text{konsistent}$, ist $CWA(T)$ immer konsistent (wenn nein: Gegenbeispiel)

NEIN

Gegenbsp:

Sei $T := \{ p(a) \vee p(b) \}$. Da weder $p(a)$ noch $p(b)$ in $\text{cons}(T)$ liegen, gilt:
 $\neg p(a), \neg p(b) \in CWA(T)$. Somit ist auch $\neg p(a) \wedge \neg p(b) \in CWA(T)$, was mit $p(a) \vee p(b)$ inkonsistent ist. $\Rightarrow T \cup \{ \neg p(a) \vee p(b) \}$ besitzt kein Modell.

5.1 Frames: Struktur von Frames systematisch beschreiben, Bestandteile?

Frame = RECORD OF

FRAME ID: object-id;
GENERIC: boolean;
AKO: list of frame-id's;
INSTANCES: list of frame-id's;
A1: W1;
...;
Am : Wm;

END;

5.2 Mehrfachvererbung, Probleme, Lösungen

- Mehrfachvererbung:
Es ist allgemein zulässig, dass ein Konzept mehrerer Superkonzepte hat. In diesem Fall erbt das Konzept die Eigenschaften von jedem Subkonzept.
- Probleme: Tritt auf, wenn dasselbe Attribut von mehreren T_i geerbt wird. In diesem Fall muss entschieden werden, von WELCHEM T_i das Attribut geerbt wird.
- Lösung: zB durch explizite Festlegung der Vererbung durch den Benutzer oder durch Konventionen in der Beschreibungssprache (zB automatisch durch Reihenfolge bei der Auflistung von Superkonzepten).

5.3 Überdeckung:

- Überdeckung:
Eigenschaften werden vom allgemeinen Fall auf den speziellen übertragen. Falls für einen speziellen Fall diese Eigenschaft nicht zutrifft, so wird sie explizit redefiniert. \Rightarrow Überdeckung

5.4 Wie werden Prozeduren in Framesystemen integriert / aufgerufen

Der Mechanismus des „procedural attachment“ erlaubt es, Prozeduren in die Framestruktur einzuhängen, die nach bestimmten Aktivierungsmustern (Triggering) ausgeführt werden.

6.1 Diagnoseproblem formal / was ist Diagnose?

Diagnose

Soll fehlerhafte Komponenten in einem System erkennen. Dies schließt sowohl technische Fehlersuche, als auch medizinische Systeme ein, die erkennen an welcher Krankheit ein Patient leidet.

Diagnoseproblem Formal:

Diagnoseproblem (SD, OBS, COMP) besteht aus:

- logische Theorie – Systembeschreibung SD
- Menge von Fakten – Beobachtungen OBS
- Menge von Konstantensymbolen – Bauteile COMP

Eine Diagnose für das Diagnose Problem ist eine minimale Menge x von fehlerhaften Komponenten, sodass die Menge von Formeln

$$SD \cup OBS \cup \{\neg ok(c) \vee c \in x\} \cup ok(c) \vee c \in COMP/x\} \quad \text{erfüllbar ist.}$$

6.2 In welche Teile zerfällt die Repräsentation , Welche Art des Wissens ist in beiden vorhanden?

6.3 Warum benötigt man bei modellbasierter Diagnose schließen? Verfahren, dass eingesetzt werden kann

7.1 Wie werden Zustände in Strips repräsentiert? Aus welchen Komponenten besteht ein Operator? Wann ist ein Operator anwendbar?

- Zustandsbeschreibung
Besteht aus einer Menge von Variablenfreien Literalen der PL (Bsp: $S = \{on(B,A), on(A,C), on(C,F1), clear(B), clear(F1)\}$)
- Operator
ein Operator hat drei komponenten:
 - Precondition:
Eine Menge $PC(op)$ von variablenfreien Literalen, die als Vorbedingung bezeichnet wird.
 - Add List:
Eine Menge $Add(op)$ von variablenfreien Literalen, die als Add List bezeichnet wird .
 - Delete List
Eine Menge $Del(op)$ von variablenfreien Literalen, die als Delete List bezeichnet wird .
- Anwendbar ??

7.2 Wie wird ein Operator angewendet?

Die Ausführung einer Aktion wird durch Anwendung eines Operators modelliert. Formal gesehen ist ein Operator op eine partielle Funktion

$op : StateDescr \rightarrow StateDescr$

wobei hier StateDescr die Menge aller Zustandsbeschreibungen ist. Partiell heisst, dass die Anwendung des Operators nicht für jede state Description definiert ist.

7.3 Was ist ein Plan, Arten des Planens in Strips, Vor und Nachteile ?

Ein Plan für eine Zustandsbeschreibung S und ein Goal G ist eine Folge

$P = \alpha_1, \dots, \alpha_n$

von Aktionen, repräsentiert durch STRIPS Operatoren op_1, \dots, op_n , sodaß

- 1.) $S_i = op_i(S_{i-1})$ für alle $i=1, \dots, n$, wobei $S_0 = S$, dh die Anwendung der Aktionen α_i in der Reihenfolge des Plans in jedem Schritt definiert
- 2.) $S_n \models y$, dh. In der resultierenden State Description ist das Goal erfüllt.

Arten (Vor&Nachteile)

Planungsmethoden:

- Progressives Planen
- Progressives Strips Planen mit rekursiver Teilplanung
- Sussman Anomalie
- Regressives Planen

Progressives Planen:

Planen durch Suche nach einer Vorwärtsstrategie. Dabei wird ausgehend von S , ein Plan P schrittweise über die Suche einer state description S' im konzeptuellen Suchgraph konstruiert, die das „Goal“ erfüllt.

Bei einer großen Zahl von STRIPS Regeln und großen Zustandsbeschreibungen ist dieser naive Ansatz allerdings nicht praktikabel. Grund: Mangelnde Effizienz durch zu große Verzweigungen der Möglichkeiten Operatoren anzuwenden. Eine fokussierte Suche ist hier erforderlich.

Progressives Strips Planen mit rekursiver Teilplanung:

Eine Möglichkeit die Suche zu fokussieren, ist das „Goal“ in Teile zu zerlegen und ausgehend von einer Lösung für einen Teil eine Gesamtlösung zu konstruieren. Am einfachsten ist hierbei, zunächst einen Plan P_1 für das Subgoal L_1 zu finden. Ist ein solcher Plan konstruiert, so versucht man, durch weitere Aktionen das nächste Subgoal zu erfüllen, bis alle Subgoals erfüllt sind. Der Algorithmus terminiert aber nicht immer. Es kann daher zu Zyklen kommen. Will man dies vermeiden, muß eine entsprechende Erkennung zyklischer Situationen vorgesehen werden, was sehr aufwendig ist.

Sussman Anomalie

Oben beschriebener Algorithmus terminiert nicht immer. Im Falle der Terminierung ist Minimalität des Plans in dem Sinn nicht sichergestellt, dass er keine überflüssigen Operatoren enthält. Bei Rückwertsuche lässt sich dieses Problem vermeiden.

Regressives Planen

Ausgehend vom „goal“ wird in Richtung der anfänglichen state description S ein Plan P in umgekehrter Reihenfolge produziert. Dh: es werden die Effekte von Operatoren der Reihe nach rückgängig gemacht, bis die Erfüllbarkeit des veränderten Goals in S gegeben ist.

Wann soll man regressiv Planen:

- Prinzipiell ist der regressive Ansatz zur Lösung praktische Probleme besser geeignet, da das Goal in der Regel klein ist und für die Erfüllung eines Subgoals wenige Operatoren vorhanden sind
- Der progressive Ansatz ist im Fall mit Variablen wesentlich einfacher. Die Behandlung von Variablen, insbesondere die Beschränkung der zulässigen Instanzierungen ist im Falle von regressiver Planung kompliziert und aufwendig.