Vorbereitung für die Vertiefungsprüfung Künstliche Intelligenz, Logikprogrammierung und Wissensrepräsentation

Dominique Ziegelmayer

dominique.ziegelmayer@rwth-aachen.de

Inhaltsverzeichnis

Kapitel 1

Einleitung

Liebe Kommilitonen, in diesem Dokument habe ich die häufigsten Prüfungsfragen für die Diplomprüfung in den Gebieten KI, KR und Logikprogrammierung zusammengefasst und mich bemüht zu fast allen Fragen ein paar Worte zu schreiben. Ich möchte hier ausdrücklich darauf hinweisen, dass ich keinerlei Garantie auf Vollständigkeit und Richtigkeit gebe. Dieses Dokument soll eine Hilfe für all diejenigen sein, die kurz vor Ihrer Prüfung stehen und selbst ihr Wissen überprüfen möchten. Nicht empfehlenswert ist ausschließlich aus diesem Dokument zu lernen und keinerlei Hintergründe zu kennen. Verbesserungen und Anmerkungen könnt ihr mir gerne schicken, allerdings muss ich natürlich sehen, wieviel Zeit ich habe, diese zu berücksichtigen. Ansonsten allen viel Erfolg, Dominique Ziegelmayer.

Kapitel 2

Künstliche Intelligenz

2.1 Agenten

2.1.1 Welche Agententypen gibt es?

Es gibt Table-Lookup Agenten, Reflexive Agenten, Agenten mit internem Weltmodell, Agenten mit expliziten Zielen und Nützlichkeitsbasierte Agenten

2.1.2 Was unterscheidet die Agenten mit expliziten Zielen von den Nützlichkeitsbasierten Agenten?

Während der Agent mit expliziten Zielen nur jeweils die Aktion wählt, die am nächsten zu Ziel führt, betrachtet der Nützlichkeitsbasierte Agent auch die Frage „Wie komme ich am effizientesten ans Ziel“.

2.1.3 Nennen Sie mögliche Eigenschaften der Umgebungen un erklären Sie diese

Accessible vs. non-accessible: Werden alle relevanten Aspekte der Umwelt über die Sensoren erfasst?
Deterministic vs. non-deterministic: Hängt der Folgezustand nur vom aktuellen Zustand und der gewählten Aktion ab?
Static vs. Dynamic: Kann sich die Welt verändern, während der Agent „Nachdenkt“
Episodic vs. non-episodic: Hängt die Wahl der nächsten Aktion nur vom aktuellen Zustand ab?
Discrete vs. Continous: Ist die Welt mit einer endlichen Zustandsmenge beschreibbar?

2.2 Suche

2.2.1 Welche Problemklassen gibt es? Erklären sie diese

Es gibt Single-State Probleme (Vollständiges Weltwissen, vollständiges Wissen über den Effekt der Aktionen), Multiple State Probleme (Unvollständiges Weltwissen, vollständiges Wissen über den Effekt der Aktionen), Contingency Problems (Vollständiges Weltwissen, unvollständiges Wissen über den Effekt der Aktionen) und Exploration Problems (Unvollständiges Weltwissen, unvollständiges Wissen über den Effekt der Aktionen).

2.2.2 Was ist ein Zistand beim Suchen?

Ein Zustand ist abstrakt gesehen ein Knoten im Suchbaum.

2.2.3 Was sind die Kanten?

Die Kanten sind Aktionen, die Einen Zustand in einen anderen transformieren.

2.2.4 Was ist dann Suche?

Suche ist ein Verfahren, welches ausgehend von einem Intitialzustand eine Sequenz von Aktionen sucht, die den Initialzustand in einen Zustand aus der Menge der Zielzustände transformiert. Dafür expandiert Suche abhängig von der verwendeten Strategie Knoten (D.h. sie erzeugt alle seine Nachfolger) ausgehend vom Wurzelknoten.

2.2.5 Was ist dann ein Ziel?

Ein Zustand aus der Menge der Zielzustände.

2.2.6 Welche uninformierten Suchverfahren gibt es?

Breitensuche, Uniform cost Search, Tiefensuche, Depth-Limited Search, Iterative Deepening und Bidirectional Search.

2.2.7 Wann sind Breitensuche und Tiefensuche Vollständig, wann Optimal?

Breitensuche ist Vollständig, wenn der Verzweigungsfaktor endlich ist und optimal wenn die Lösung Tiefenoptimal ist, d.h. wenn die flachste Lösung auch die beste ist. Tiefensuche ist Vollständig, wenn der Baum keine unendlichen Pfade hat und nie Optimal.

2.2.8 Welche uninformierte Suche würden Sie verwenden?

Bei einer unbekannten Suchtiefe würde ich iterative Deepening verwenden, da das Verfahren Vollständig und Tiefenoptimal ist jedoch die Zeit und Platzkomplexität in der selben Komplexitätsklasse liegt wie Tiefensuche (Zeit: $O(b^d)$, Platz: $O(bd)$)

2.2.9 Was ist eine Heuristik bei der Suche und was macht man damit?

Eine Heuristik bei der Suche ist eine Funktion (denotiert als $h(n)$), welche die Entfernung vom aktuellen Knoten zum Zielknoten approximiert. Diese Funktion wird von den Informierten Suchen verwendet um die Suche „anzuleiten“und sie dadurch effizienter zu machen.

2.2.10 Welche Anforderungen werden an eine Heuristik gestellt?

Eine Heuristik sollte leicht zu berechnen sein und die Entfernung zum Ziel möglichst exakt abschätzen.

2.2.11 Wie erhält man eine solche Heuristik?

Indem man das Problem abstrahiert, also möglichst viele Detail unbetrachtet lässt.

2.2.12 Welche informierten Suchverfahren kennen Sie?

Greedy Search, A*, IDA*, SMA*, Hill Climbing, Simulated Annealing

2.2.13 Unter welchen Bedingungen ist A* Vollständig, unter welchen Optimal?

A* ist immer Vollständig. Optimal ist A* jedoch nur, wenn die verwendete Heuristik zulässig ist, d.h. $\forall n \in V$ muss gelten: $h(n) \leq h*(n)$, wobei $h*(n)$ die tatsächlichen Kosten von n zum Ziel bezeichnet.

2.2.14 Welche Komplexität hat der A*-Algoritmus?

A* basiert auf Uniform Cost Search und ist somit für eine beliebige (zulässige) Heuristik nicht besser als Zeit: $O(b^d)$, Platz: $O(b^d)$ abschätzbar. Im Average Case ist A* jedoch recht gut.

2.2.15 Erklären Sie IDA*

IDA* ist grundsätzlich iterative deepening, jedoch nicht mit einem Tiefenlimit, sondern mit einem $f(n)$-Kostenlimit. In jedem Iterationsschritt werden dann die Söhne der soeben expandierten Knoten betrachtet und der günstigste verwendet um das neue $f(n)$-Kostenlimit festzusetzen. IDA* hat den Nachteil, dass es in der Regel mehr Knoten als A* expandiert, dafür jedoch nur die Platz-Komplexität der Tiefensuche (bzw. Iterative deepening) besitzt.

2.2.16 Erklären Sie SMA*

SMA* ist eine Variante der A* Suche, jedoch mit einem Speicherlimit. Heirbei ist der Vorteil, dass immer der gesamte zur Verfügung stehende Speicher verwendet wird. Ein gravierender Nachteil dieser Methode ist jedoch, das Probleme, die (theoretisch) mit dem A* Algorithmus lösbar sind durch die Speicherbegrenzung zu einer exponentiellen Laufzeit führen.

2.3 Spiele

2.3.1 Wie funktioniert Minimax?

Betrachtet wird der vollständige Spielbaum zwischen zwei Spielern MIN und MAX. Zunächst werden die Terminalknoten durch die Utilityfunktion bewertet (Sie Ordnet beispielsweise Loss=-1, Win=1 und Tie=0 zu). Im Anschluss werden diese Werte wie folgt durch den Baum „hochpropagiert“: Ist ein Knoten auf MAX- Ebene (also ist MAX am Zug) erhält er den Wert des Maximums seiner Söhne, ist der Knoten auf MIN-Ebene, erhält er das Minimum seiner Söhne. Am Wurzelknoten steht dann die Punktzahl die MAX erhält, wenn MIN rational spielt.

2.3.2 Was ist wenn MIN nicht rational, d.h. nicht optimal spielt mit dem Wert an der Wurzel

Spielt MIN nicht rational, ist die Punktzahl am Wurzelknoten nicht der Wert den MAX erhält, sondern das Minimum, das er erreichen kann.

2.3.3 Wie funktioniert $\alpha - \beta$ Suche?

An den Knoten der Bäume werden „Gültigkeitsintervalle“($\alpha - \beta$ Intervalle) betrachtet, so das für jeden Knoten getestet werden kann, ob sein Teilbaum Betrachtet werden muss oder nicht. Der Effekt dieses Verfahrens ist, dass Teilbäume, die auf keinen Fall gewählt werden egal wie sie ausgewertet werden, verworfen werden.

2.3.4 Wie ist die Komplexität von $\alpha - \beta$ Suche?

$\alpha - \beta$ Suche basiert auf der Tiefensuche und ist daher genau wie diese abzuschätzen.

2.3.5 Was ist Expectiminimax?

Dies ist ein Minimax-Algorithmus für Spiele mit Zufallselement. Anstatt nur die Werte der Knoten zu betrachten, betrachtet Expectiminimax die über die Erwartungswerte gewichteten Werte der Knoten.

2.4 Planung

2.4.1 Was sind die Unterschiede zwischen Planung und Suche?

Suche betrachtet Zustände als Black-Boxen, Planung interessiert sich für dessen Komponenten. Während Suche immer alle Knoten expandiert, werden bei Planung nur einige Knoten Expandiert. Zustände müssen für die Planung nicht immer vollständig spezifiziert werden. Während Suche nach einer total geordneten Sequenz von Aktionen sucht, die den Startzustand in den Endzustand transformiert, sucht Planung nach einer (möglicherweise Partiell geordneten) Beschreibung eines Plans.

2.4.2 Was ist ein STRIPS-Operator?

Ein STRIPS-Operator ist ein 3-Tupel (Name, Vorbedingungen, Effekte), wobei die Vorbedingungen ausschließlich negative Literale und die Effekte positive und negative Literale enthalten darf.

2.4.3 Was kann man daran vereinfachen?

Man kann die Effekte in ADD und DELETE Liste unterteilen und hat es dann nur noch mit positiven Literalen zu tun.

2.4.4 Was ist der Suchraum?

Der Suchraum bei STRIPS ist die Menge der STRIPS-Operatoren, da in jedem Schritt ein Operator gesucht wird, der mindestens eine unerfüllte Vorbedingung erfüllt.

2.4.5 Was ist ein Plan?

Ein Plan ist eine Menge von Planschritten (STRIPS-Operatoren) zusammen mit einer Partiellen Ordnung. Die Partielle Ordnung $S_i \preceq S_j$ sagt dabei aus, dass der Planschritt S_i vor dem Planschritt S_j ausgeführt wird. Desweiteren besteht ein Plan aus einer Menge von Substitutionen [x/t] und einer Kausalen Relation ($\rightarrow^c$), wobei $S_i \rightarrow^c S_j$ bedeutet, dass S_i die Vorbedingung c von S_j erfüllt.

2.4.6 Wann ist ein Plan eine Lösung?

Ein Plan ist eine Lösung wenn er konsistent und vollständig ist. Ein Plan ist konsistent, wenn die Unique Name assumption gilt und zusätzlich gilt wenn $S_i \preceq S_j$ dann nicht $S_j \preceq S_i$. Ein Plan ist vollständig, wenn alle Vorbedingungen aller Planschritte erfüllt sind.

2.4.7 Wie sieht der Anfang beim POP-Algoritmus aus?

Man hat zwei STRIPS-Operatoren, Start und Finish, wobei Start ausschließlich aus Effekten und Finish ausschließlich aus Vorbedingungen besteht. Zusätzlich muss eine Partielle Ordnung Start $\preceq$ Finish existieren. Zu beachten ist, dass der Startzustand ausschließlich aus funktionsfreien Grundliteralen besteht.

2.4.8 Was ist ein Threat bei der STRIPS-Planung?

Ein Threat ist ein Planschritt, der potentiell die kausale Verbindung zweier Planschritte ($S_i \rightarrow^c S_j$) zerstören kann. Ein solcher Planschritt muss dann entweder nach $S_i \rightarrow^c S_j$ (Promotion) oder vor $S_i \rightarrow^c S_j$ (Demotion) eingefügt werden.

2.5 Unsicheres Wissen

2.5.1 Wie lautet die Bayes Formel?

$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$

2.5.2 Wofür wird die Bayes Formel in der Praxis verwendet?

Die Bayes Formel kann beispielsweise in der Diagnose verwendet werden (Schließen von Effekten auf die Ursachen).

2.5.3 Wie lautet die Formel für das Bayes Update und wofür benötigt man sie in der KI?

Die Formel für das Bayes Update lautet: $P(A|B,C) = P(A|B) \cdot \frac{P(C|A)}{P(C|B)}$. In der KI wird sie zum Beispiel bei der Diagnose (zufügen von beobachteten Symptomen) oder in der Lokalisierung (zufügen von Sensorreadings) verwendet.

2.5.4 Wann kann man das Bayes Update einsetzen?

Das Bayes Update ist nur dann anwendbar, wenn die „Evidences“von einander stochastisch unabhängig sind, d.h. $P(B|A,C) = P(B|A)$, also B ist von C unabhängig wenn A gegeben ist.

2.5.5 Und wie handhabt man das in der Praxis?

Das Bayes Update liefert in der Praxis auch bei Verletzung der stochastischen unabhängigkeit sehr gute Ergebnisse.

2.5.6 Was ist eine gemeinsame Verteilung?

Eine gemeinsame Verteilung ist die Zuweisung von Wahrscheinlichkeiten zu jeder Kombination der Zufallsvariablen aus der Verteilung. Die Verteilung ist Normiert, d.h. alle Wahrscheinlichkeiten addieren sich zu 1 und atomare Ereignisse schließen sich aus. Eine gemeinsame Verteilung kann man als Tabelle notieren. Das Problem hierbei ist jedoch, dass die Tabelle exponentiell in der Anzahl der Zufallsvariablen wächst.

2.5.7 Wie kann man das Vereinfachen?

Anstatt alle Wahrscheinlichkeiten zu repräsentieren, beschränkt man sich auf die kausalen Relationen und die Wahrscheinlichkeiten der Zufallsvariablen, die Einfluß auf einander haben. Eine solche Darstellung (als gerichteter azyklischer Graph) ist ein Believe Network.

2.5.8 Erklären Sie mal wie ein Believe-Network aufgebaut ist

Ein Believe Network ist ein gerichteter azyklischer Graph, dessen Knoten Zufallsvariabeln und dessen Kanten Kausale Verbindungen sind. Zusätzlich enthält jeder Sohnknoten eine CPT (Condianal probability table), in der alle Wahrscheinlichkeiten der Form $P(X|Parents(X))$ eingetragen sind.

2.5.9 Wann ist ein Believe Network die korrekte Repräsentation einer gemeinsamen Verteilung?

Ein Believe Network ist dann eine korrekte Darstellung einer gemeinsamen Verteilung, wenn für alle Knoten X_i gilt: $P(X_i|X_{i-1}, ..., X_1) = P(X_i|Parents(X_i))$ und $Parents(X_i) \in \{X_1, ..., X_{i-1}\}$.

2.5.10 Was ist d-Seperation?

d-Separation ist ein Verfahren um die stochastische Unabhängigkeit von Mengen von Zufallsvariablen festzustellen. Zwei Mengen von Zufallsvariablen X und Y werden von E d-Separiert $\Leftrightarrow$ Jeder ungerichtete Pfad zwischen X und Y wird von E blockiert. Ein Pfad wird dann von E blockiert, wenn ein Knoten Z existiert mit:

1. $Z \in E$ und beide Kanten des Pfades gehen aus Z heraus
2. $Z \in E$ und eine Kante des Pfades geht in Z herein, die andere heraus
3. $Z \notin E$ und keiner seiner Söhne ist in E und beide Kanten gehen in Z herein.

2.5.11 Und wie verwendet man dann d-seraration um stochastische Unabhängigkeit zu zeigen?

Nach dem Theorem von Judas Pearl gilt, das wenn zwei Mengen X und Y durch E d-separiert werden, dass sie unter der Bedingung, dass E gegeben ist auch stochastisch unabhängig sind. Das Verfahren ist übrigens nicht vollständig und in Polynomzeit berechenbar.

2.5.12 Was muss für das Believe Network gelten, damit ein Inferenz algorithmus effizient arebiten kann?

Believe Networks in denen der Inferenz-Algorithmus effizient ist müssen singly-connected sein. Das bedeutet, das zwischen je zwei Knoten des Graph höchstens ein ungerichteter Pfad existiert. In Graphen die dieser Eigenschaft nicht genügen benötigt der Inferenzalgoritmus statt linearer Zeit exponentielle Zeit.

2.5.13 Wie konstruiert man ein Believe Network?

Zunächst wählt man alle relevanten Zufallsvariablen aus und ordnet sie. Sei ZV diese geordnete Menge von Zufalssvariablen $X_1, ...$ Dann nimmt man jeweils den ersten Knoten aus der Menge ZV, berechnet für Ihn die minimale Menge Parents(X_i) und fügt die Kanten entsprechend ein. Dieses Verfahren iteriert man, bis die Menge ZV leer ist.

2.5.14 Was passiert, wenn die Ordnung schlecht gewählt wurde?

Wenn die Ordnung der ZV's schlecht gewählt wurde kann das Netz sehr groß werden.

2.5.15 Welche Arten von Inferenz gibt es in einem Believe Network?

Es gibt Diagnostisch (Von Effekten zu Ursachen), Kausal (Von Ursachen zu Effekten), Interkausal (Zwischen Ursachen eines gemeinsamen Effekts) und Mixed (Kombination von 2 oder mehr der vorher genannten Verfahren).

2.5.16 Was versteht man unter Marginalisierung?

Bei der Marginalisierung versteht man die Berechnung einer Wahrscheinlichkeit P(A) durch Addition aller Wahrscheinlichkeiten der gemeinsamen Verteilung in denen A vorkommt.

2.5.17 Was versteht man unter Conditionalisierung?

Unter Conditioning versteht man das Aufspalten einer Wahrscheinlichkeit auf seine Disjunkte. Zum Beispiel: $P(A) = P(A|B) \cdot P(B) + P(A|\neg B) \cdot P(\neg B)$.

2.6 Lernen

2.6.1 Erklären Sie die Information-Theory

In der Informationstheorie wird der Informationsgehalt einer Aussage in Bits berechnet. So seien also $v_1, ..., v_n$ Anwtorten und $P(v_1), ..., P(v_n)$ ihre Auftretens-Wahrscheinlichkeiten. Dann ist ihr Informationsgehalt $I(v_1, ..., v_n) = \sum -P(v_i) \cdot log_2(P(v_i))$.

2.6.2 Wozu kann man die Informationstheorie verwenden?

Man kann die Informationstheorie verwenden um das beste Teilungsattribut beim DT-Learnin Algorithm zu bestimmen.

2.6.3 Welche Arten des Feedbacks gibt es beim Lernen?

Man unterscheidet beim Lernen zwischen Supervised Learning, Reinforcement Learning und Unsupervised Learning. Während der Agent beim Supervised Learning die Optimale Ausgabe (das optimale verhalten) kennt, bekommt er beim Reinforcement Learning sein Feedback nur in Form von Belohnung und Bestrafung. Beim Unsupervised Learning stehen dem Agenten dagegen keinerlei Informationen über die Güte seiner Handlungen zur Verfügung.

2.6.4 Was ist induktives Lernen?

Induktives Lernen ist Lernen durch Tupel der Form (Eingabe, Ausgabe). Hierbei versucht man die optimale Funktion f durch die gegebenen Wertepaare möglichst exakt anzunähern.

2.6.5 Welche Lernverfahren kennen Sie?

In der Vorlesung wurden folgende Lernverfahren vorgestellt: Decision Trees, Decision Lists und Neuronale Netzwerke.

2.6.6 Für welche Probleme eignen sich die einzelnen Lernverfahren?

Decision Trees und Decision Lists eignen sich beide für attributbasierte Probleme, bei denen die Attribute eine möglichst gute Trennung der Examples erlauben. Bei attributbasierten Problemen, bei denen jedes Attribut einen ähnlich schlechten Informationsgehalt hat (z.B.Mehrheitsfunktion) sind Neuronale Netzwerke vorzuziehen.

2.6.7 Was sind die Vor- bzw. Nachteile von Neuronalen Nezwerken?

Vorteile von Neuronalen Netzwerken: Können gut mit verrauschten Daten umgehen, eignen sich herrvorragend für Linear separierbare Funktionen.
Nachteile von Neuronalen Netzwerken: Im Allgemeinen kann Konvergenz und Effizienz beim Lernen nicht garantiert werden, Es gibt keine Heuristiken für die Wahl des richtigen Netzes (Großes Netz = keine Generalisierung, kleines Netz = Darstellbarkeit).

2.6.8 Was sind die Vorteile von Decision Trees, was die Vorteile von Decision Lists?

Decision Trees sind sehr transparent und übersichtlich, während k-DL eine echte Einschränkung darstellen und sowohl eine kompaktere Darstellung erlauben als auch weniger Examples benötigen als ein unbeschränkter DT um eine bestimmte Funktion zu lernen.

2.6.9 Welche Funktionen kann man durch ein Feed-Forward Neuronal-Network mit keinem, mit einem, mit zwei hidden Layern darstellen?

Mit keinem hidden Layer lassen sich die linear separierbaren, mit einem hidden layer die stetigen und mit zwei hidden Layern alle Funktionen darstellen.

2.6.10 Wie aufwendig ist das Lernen in Neuronal Networks?

Das Lernen in Perceptrons konvergiert immer, soweit die Funktion darstellbar ist. Ferner ist der Aufwand linear mit der Anzahl der Examples. Im Allgemeinen muss jedoch der Back-Propagation Algorithmus verwendet werden, der weder Konvergenz noch Effizienz garantiert.

2.6.11 Was kann passieren, wenn man das Netz zu groß, bzw. zu klein wählt?

Wenn man ein Neuronales Netzwerk zu groß wählt, wird die Noise mit gelernt, d.h. das Netz kann nur sehr schlecht generalisieren. Ist das Netz zu klein, kann es sein, dass die Funktion nicht mehr darstellbar ist.

2.6.12 Wie lautet die Update-Regel für Perceptrons?

Die Updateregel lautet: $W_{i,j} = W_{i,j} + \alpha \cdot I_j \cdot Error$, wobei Error=T-O (Richitger - Tatsächlicher Output) und α die Learning Rate ist.

2.6.13 Wie funktioniert das beim Back-Propagation Algorithmus?

Der Back Propagation Algorithmus funktioniert wie folgt: Zunächst wird er in einer Schleife ausgeführt, die so Lange iteriert wird, bis das Netz konvergiert. Dann werden für jedes Example die Gewichte wie folgt aktualisiert:

1. Auf dem Output Layer:
   $W_{i,j} = W_{i,j} + \alpha \cdot a_j \cdot Error_i \cdot g'(In_i)$
2. Für jedes folgende Layer:
   $\Delta_j = g'(In_j) \cdot \sum_i W_{i,j} \cdot Error_i \cdot g'(In_i)$
   $W_{k,j} = W_{k,j} + \alpha \cdot I_k \cdot \Delta_j$

Kurz erklärt aktualisiert der Algorithmus Ähnlich dem Algorithmus des Perceptrons die Gewichte auf Output-Level und propagiert dann diese Werte hoch.

2.6.14 Kennen Sie eine Heuristik um die Anzahl der Kanten in einem Neural Network anzugeben?

Da gibt es die Heuristik des optimal Brain Damage. Hier beginnt man mit einer maximalen Anzahl von Kanten und streicht dann Kanten die gemäß der Informationstheorie zu vernachlässigen sind.

2.6.15 Wie lernt man mit Decision Trees?

Man betrachtet eine Menge von Examples die man lernen möchte. Gibt es nur noch positive oder negative Antworten, antworte entsprechend. Gibt es noch sowohl positive als auch negative Antworten suche gemäß der Informationstheorie das beste Teilungsattribut, teile die Menge und beginne von vorne. Das beste Teilungsattribut findet man dabei wie folgt:
Zunächst berechnet man den Informationsgehalt I des Zielattributs, bzw. des letzten Teilungsattributs. Dieses sei mit I bezeichnet. Dann berechnet man für jedes Attribut X, Gain(X)=I-Remainder(X). Der Remainder ist dabei der Gewichtete

Informationsgehalt der einzelnen Werte die das Attribut annehmen kann oder vereinfacht gesagt, der Remainder ist der Informationsgehalt, der nach Teilung der Examples durch X erhalten bleibt.

2.6.16 Wie groß wird ein Decision Tree für eine Mehrheitsfunktion der Größe n?

Er bekommt die Größe 2^n (Vollständiger Baum).

2.6.17 Was ist PAC-Learning?

PAC Learning ist ein Verfahren, mit dem man die minimale Anzahl der Examples bestimmen kann, die notwendig ist um die optimale Funktion in einer ε Umgebung mit Wahrscheinlichkeit (1-δ) anzunähern.

2.6.18 Wie viele Examples benötigt man bei Decision Trees, wie viele bei eine k-Decision List?

1. Für einen DT: $N \geq \frac{1}{\varepsilon} \cdot (ln\frac{1}{\delta} + ln|H|)$, mit $|H|$=Größe des Hypothesenraums und N=Anzahl der Samples.
2. Für eine k-DL: $N \geq \frac{1}{\varepsilon} \cdot (ln\frac{1}{\delta} + O(n^k \cdot ln(n^k))$

2.7 Knowledge Bases

2.7.1 Warum ist Inferenz bei Knowledge Bases schwerer als bei Datenbanken?

Das liegt daran, dass Datenbanken annehmen vollständiges Wissen über die Welt und die Individuen zu besitzen.

2.7.2 Nennen Sie die 3 Ebenen der KR und erklären Sie diese

Knowledge-Level: Abstraktes Wissen, Symbolic Level: Kodiertes Wissen in einer Logiksprache, Implementation Level: Implementierung, z.B. Bitmuster, ...

2.7.3 Definieren Sie die Closed World Assumption ($\models_c$)

Ein Satz α ist unter der CWA aus KB ableitbar $(KB \models_c \alpha) \Leftrightarrow KB \cup Negs \models_\alpha$, mit $Negs = \{\neg p | p \text{ ist atomar und } KB \not\models p\}$

2.7.4 Wie wird eine Anfrage unter der CWA ausgewertet?

Unter der CWA werden komplexe Anfragen auf atomare reduziert. Zum Beispiel: $KB \models_c (\alpha \wedge \beta) \Rightarrow KB \models_c \alpha \ \wedge \ KB \models_c \beta$

2.7.5 Was können Sie zur Konsistenz unter der CWA sagen?

2.7.6 Was ist die CWA und was bewirkt sie?

2.7.7 Was ist mit der Monotonie unter der CWA?

2.7.8 Was macht man mit Anfragen, die Quantoren enthalten?

2.8 Resolution

2.8.1 Wie schwer ist Resolution?

2.8.2 Wie führt man eine Anfrage KB$\models \alpha$ durch?

2.8.3 Wie schwer ist Unifikation?

2.8.4 Was ist am Algorithmus in der Vorlesung Exponentiell?

2.8.5 Was ist Answer Extraction?

2.8.6 Welche Strategien gibt es Resolution effizienter zu machen?

2.8.7 Was ist ein Herbrand-Universum?

2.8.8 Was ist die Herbrand-Basis zu S?

2.9 Robots

2.9.1 Erklären Sie den Algorithmus zur Pfadplanung

2.9.2 Was sind die Vorteile des Value-Iteration Algorithmus?

2.9.3 Wie funktioniert Selbstlokalisierung?

2.9.4 Wie werden die Bewegungen integriert?

2.9.5 Wie werden die Sensordaten integriert?

2.9.6 Erklären Sie die Markov-Assumption

2.9.7 Was passiert wenn man in einer dynamischen Welt von der Markov-Assumption ausgeht?

Kapitel 3

Wissensrepräsentation

3.1 First Order Logic

3.1.1 Was bedeutet die logische Implikation $KB \models \alpha$?

$KB \models \alpha$ bedeutet, dass jedes Modell von KB auch ein Modell von α ist.

3.1.2 Wie zeigt man eine solche Deduktion $KB \models \alpha$?

Eine solche Deduktion kann man zeigen, indem man die Folgerung äquivalent wie folgt umschreibt: $KB \wedge \neg\alpha$ unerfüllbar. Dann muss man diesen Ausdruck nur noch in KNF überführen (im FO-Fall zusätzlich Skolemisierung) und dann Resolvieren (im FO-Fall mit Unifikation). Ist die leere Klausel ableitbar, ist die Folgerung gültig.

3.1.3 Was ist der Unterschied zwischen Wissensbasen und Datenbanken?

Datenbank nehmen an über vollständiges Wissen zu verfügen. Deshalb sind Anfragen in Datenbank auch auf atomare Anfragen reduzierbar und so effizient lösbar.

3.1.4 Wofür benötigt man die Domain-Closure Formell?

Die DC wird benötigt um Anfragen mit Quantoren zu beantworten.

3.1.5 Wie stellt man Anfragen, wenn man FO-Logik ohne CWA und DC hat?

Per Resolution.

3.1.6 Wie schwer ist AL-Resolution?

AL-Resolution ist Exponentiell entscheidbar, d.h. NP Schwer.

3.1.7 Wie schwer ist FO-Resolution?

FO-Resolution ist Semi-Entscheidbar, d.h. ist die leere Klausel ableitbar, wird diese in exponentieller Zeit gefunden; Existiert diese nicht, muss das Verfahren nicht terminieren.

3.1.8 Woher kommt die Unentscheidbarkeit bei der FO-Resolution?

Die Unentscheidbarkeit rührt daher, dass Variablen in FO für alle ihre instanzen stehen und eine geschickte Substitution und Auswahl der Resolutionsklauseln notwendig ist.

3.1.9 Was kann man bei der FO-Resolution vereinfachen?

Verwendung von Horn-Formeln, Resolutionsstrategien, User-Control.

3.2 Horn Logik

3.2.1 Was ist eine Horn-Formel?

Eine Horn-Formel ist eine Formel in KNF, deren Klauseln je höchstens ein pos. Literal enthalten.

3.2.2 Was ist SLD-Resolution?

3.2.3 Wie ist die Komplexität von SLD-Resolution?

3.2.4 Warum verwendet man SLD-Resolution, d.h. Hornformeln, obwohl diese eine Einschränkung darstellen?

3.2.5 Wie ist es mit der Effizienz im Aussagenlogischen Fall?

3.3 Procedural Control

3.3.1 Erklären Sie den CUT-Operator

3.3.2 Was ist Negation-as-Finite-Failure?

3.3.3 Wie wird das durch den CUT simuliert?

3.4 Production Systems

3.4.1 Was ist ein Production System?

3.4.2 Erklären Sie die verschieden Phasen eines Zyklus

3.4.3 Wie bestimmt man die Regeln die feuern (Conflict Resolution)?

3.5 Beschreibungslogiken

3.5.1 Was sind die Vorteile von Konzeptsprachen?

Description Logics sind Objektorientiert, d.h. sie vereinen alle relevanten Daten über ein Objekt in

3.5.2 Was ist eine Role?

3.5.3 Was für Operatoren gibt es?

3.5.4 Geben sie die Interpretation $\Im = (D, \Phi)$ formal an

3.5.5 Schreiben Sie AND, ALL, FILLS, AT-MOST und AT-LEAST formal auf

3.5.6 Welche Inferenzen interessieren uns bei Beschreibungslogiken?

3.5.7 Wie berechnet man die Subsumierung?

3.5.8 Wo arbeitet der Strukturmatching-Algorithmus rekursiv?

3.5.9 Erklären sie den Operator SOME von FL^-

3.5.10 Schreiben Sie (SOME r C) formal auf

3.5.11 Wird (SOME r C) von (ALL r C) subsumiert?

3.5.12 Wie sieht der RESTR-Operator von FL formal aus?

3.5.13 Was hat die Verwendung von RESTR für Auswirkungen?

3.6 Hierarchie und Vererbung

3.6.1 Welche Preemtion-Strtegien haben wir kennengelernt?

3.6.2 Was ist eine Credulous Extension?

3.7 Defaults

3.7.1 Wie ist das minimal Entailment $\models_m$ definiert?

3.7.2 Was sind default-logics?

3.7.3 Wann ist eine Menge E eine Erweiterung?

3.7.4 Warum verwendet man die autoepistemic logic?

3.7.5 Wie ist eine Stabile Extension definiert?

3.8 Action / Planning

3.8.1 Was ist das Situationskalkül?

Das Situationskalkül ist ein Dialekt von FOL, der zur Beschreibung von dynamischen Welten verwendet wird.

3.8.2 Was ist das Frame-Problem?

Das Frame Problem bezeichnet das Problem, dass die Anzahl der Frame-Axiome $2 * F * A$ beträgt, wobei F die Anzahl der Fluenten und A die Anzahl der Aktionen denotiert. Es gilt zur Lösung des Problems eine kompaktere Darstellung zu finden.

3.8.3 Geben Sie eine einfache Lösung des Frame-Problems an

3.8.4 Erklären Sie den Regressions-Operator

3.8.5 Erklären Sie Linear-Regresion-Planning

3.8.6 Warum ist Linear-Regression-Planning vollstänig und korrekt?

3.9 Abductive Reasoning

3.9.1 Was versteht man unter Abductive Reasoning?

3.9.2 Wir hatten eine Anwendung bei der Fehlererkennung in Schaltkreisen. Wie funktioniert dies?

3.9.3 Wie berechnet man ein minimales Fehlerszenario?

Man Berechnet die Menge der Erklärungen $\{(P - \{out_i\} | Pist Primimplikant und \{out_i\} \subseteq P\}$

3.9.4 Wie sieht es dabei mit der komplexität aus?

Die Berechnung von Primimplikaten geschieht durch Resolution (Resolution ist vollständig für nicht tautologische Primimplikanten). Da Resolution verwendet wird, kann das verfahren exponentiell werden.