Protokoll - Dimplomprüfung - Praktische Informatik - Prof. Lakemeyer

Datum: 21.07.2003
Fächer: Betriebssysteme, Datenbanken, Künstliche Intelligenz
Note: 1,3
Kommentar: - Keine Panik -
Da ich dieses Protokoll aus dem Kopf geschrieben habe, kann ich keinerlei Garantie für Vollständigkeit und Korrektheit (insbesondere bei meinen Antworten) geben. Ich habe hier versucht auch etwas den Prüfungsdialog hervor zuheben. Die Prüfung ist nicht einfach ein Frage-, Antwortspiel. Wenn man mal länger überlegt oder auf dem falschen Dampfer ist, gibt Prof. Lakemeyer auch Tipps bzw. Denkanstöße. Er geht gerne auf Stichworte ein, die man selbst geliefert hat. Lässt man aber absichtlich ein Stichwort aus um von unangenehmen Fragen verschont zu bleiben, so fragt er oft nach, was es denn sonst noch für Möglichkeiten gibt. Um Verwirrung zu vermeiden: Q(uestion) ist Prof. Lakemeyers Gesprächsteil. A(ntwort) ist mein Teil. Die Antworten sind recht knapp gehalten, da ich nicht mehr genau weiß, was ich alles erzählt habe.

Betriebssysteme
Q: Semaphoren?
A: Synchronisation. IPC.

Q: Klassisches IPC-Problem: Dining Philosophers. Wie löst man das?
A: Grob erklärt. Mutex um Zugriff auf Array mit Status-Variablen exklusiv zu halten. Eine Semaphore für jeden Philosoph.

Q: Wozu braucht man Mutexe in Multiprozessorsystemen?
A: ... denk ... Bus sperren.

Q: Was gibt es denn so für Multiprozessorsysteme?
A: UMA - Zugriff auf alle Speicherwörter dauert gleich lang, NUMA - langsamerer Zugriff auf remote Speicher als auf lokalen Speicher, CC-NUMA versteckt unterschiedliche Zugriffszeiten, NC-NUMA nicht.

Q: Verschaltung in Multiprozessorsystemen. Wie sieht Crossbar Switch aus?
A: Grob erläutert. Kreuzungspunkte können individuell geschaltet werden. Es kann immer auf jedes verfügbare Speichermodul zugegriffen werden. Aber aufwendig und teuer, da sehr viele switches.

Q: Was gibt es sonst noch für Verschaltungsmöglichkeiten?
A: Sogenanntes Omega Netzwerk. Hierbei deutlich weniger Switches. Allerdings können sich Prozesse bei Zugriff auf Speichermodule in die Quere kommen. Also, Tradeoff zw. Anzahl Switches und Parallelität.

... hier kam eventuell noch was, habe ich aber leider schon wieder vergessen ... Es war irgendetwas mit Prozessen und Threads. Ach ja und Kernel- vs. User-level-Threads.

Q: Dateisysteme: FAT-16/32, Unix Dateisystem. Wo ist denn da der Unterschied?
A: FAT: File Allocation Table. Muss komplett im Speicher sein. Kann ziemlich groß werden. Bei Unix: i-nodes. Müssen nur im Speicher sein, wenn entsprechende Datei benutzt wird.

Q: Was ist denn ein großes Problem mit FAT-16?
A: ???

Q: Man denke z.B. daran, dass man aufgrund des Problems viele Partitionen einrichten muss.
A: FAT-16 reicht nicht aus um große Platte komplett zu adressieren. Außerdem werden bei maximaler Partitionsgröße (was nicht sehr viel ist) große Blöcke verwendet, wodurch Speicherplatz verschwendet wird.

Q: Ein ziemlich neues Dateisystem ist NTFS. Was gibt es denn da als nützliches feature, das es bei anderen Dateisystemen nicht gibt.
A: - raten - Erfüllt Sicherheitsstandard des US Verteidigungsministeriums.

Q (grinst): Das ist bestimmt nützlich. Denken sie mal an den Multimedia-Bereich.
A: ... ? ...

Datenbanken

Q: Man kann mehrere Streams in einer Datei anlegen. OK. Um noch etwas in der Nähe der Betriebssysteme zu bleiben: Welche Datenstrukturen gibt es denn bei Datenbanken bei der physischen Datenorganisation?
A: ISAM, B-Bäume, B^+-Bäume, Hashing, R-Bäume.

Q: Was gibt es mit ISAM für Probleme?
A: Aufwendige Indexverschiebung, wenn ein komplett neuer Knoten in den Datenseiten benötigt wird.

Q: Wie aufwendig ist das denn?
A: ... Ist in linearer Zeit möglich.

Q: Was ist gut an ISAM?
A: Schnelles Lesen. Also für statische Daten durchaus einsetzbar.

Q: Hashing ist ja sehr schnell. Wieso braucht man denn überhaupt noch was anderes?
A: - falsche Fährte - Wenn viele Daten eingefügt werden, dann häufig Kollisionen.

Q: Ist eigentlich kein Problem, bei guter Hashfunktion.
A: - Klick - Range-Anfragen. Man muss für jeden Wert die Hashfunktion wieder aufs neue berechnen.

Q: Genau. Wie geht das denn besser?
A: B^+-Bäume. Absteigen bis Blatt und dann einfach Blätter durchlaufen, da diese in sortierter Reihenfolge vorliegen und über Zeiger verknüpft sind.

Q: Wie groß ist so ein Knoten im Baum?
A: Größe entspricht einer Seite.

Q: Wie groß sind denn so Seiten?
A: 4KB, 8KB.

Q: Gut. In Relationaler Entwurftheorie gibt es die Normalformen. Wie sieht denn z.B. die 4. NF aus?
A: Kurz MVDs angesprochen. Es muss eine der beiden folgenden Bedingungen gelten: MVD $\alpha \rightarrow \beta$ ist trivial. Oder α Superschlüssel. (Der $\rightarrow$ soll natürlich eigentlich einer der doppelten Sorte sein.)

Q: Armstrong-Axiome?
A: Reflexivität, Transitivität, Verstärkung.

Q: wie verwendet man diese um weitere FDs zu bestimmen.
A: Anschauen, welche Axiome auf die gegebenen FDs anwendbar sind um neue FDs herzuleiten.

Q: Gibts da nicht eine bestimmten Algorithmus?
A: ... Mir viel nur der AttrHülle-Algoirthmus ein und habe diesen kurz erklärt.

- Lakemeyer hat irgendwas erklärt, dass es noch einen anderen gibt, habe ich aber gar nicht so richtig verstanden. -

Q: Warum will man denn die Normalformen haben?
A: Ohne NF, schlechte Relationenschemata. Führen zu Einfüge-, Lösch-, Update-Anomlaien. Diese kurz erklärt.

... hier kam eventuell noch was, habe ich aber leider schon wieder vergessen ...

Q: Was sind bzw. macht man mit Sichten?
A: Man kann verschiedenen Benutzern, verschiedene Daten auf einfache Weise zugänglich machen.

Q: Wie erzeugt man so eine Sicht?
A: create view V as ... und dann andere SQL-Befehle.

Q: Problem: Update-fähige Sichten?
A: Führen zu Inkonsistenzen. (Weiß gar nicht, ob das stimmt.)

Q: Wie erhält man denn update-fähige Sichten?
A: ?

Q: Es sollten z.b: keine Aggregatfunktionen darin vorkommen. Wie sieht es denn mit den relationalen Anfragesprachen aus. Sind diese unterschiedlich mächtig.
A: Relationale Algebra, Tupel-, Domänen-Kalkül, auf sichere Ausdrücke eingeschränkt sind gleichmächtig. Werden als relational vollständig bezeichnet.

Q: Was sind den sichere Ausdrücke?
A: Ergebnisse von Anfragen bei Tupel-Kalkül müssen Teilmenge der Domäne sein. Bei Domänen-Kalkül noch 2 weitere Bedingungen.

Künstliche Intelligenz
Q: Unterschied zw. Datenbanken und Wissenbasen?
A: CWA. Domain Closure. Erklärungen was das ist, wie es funktioniert.

Q: Wie schwer ist den Anfrageberechnung bei KB?
A: Schwer. FOL-Resolution semi-entscheidbar. Exp. Laufzeit.

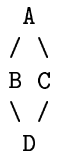
Q: Und wie sieht es bei AL-Resolution aus?
A: Entscheibar, denn man kann gesammte Resolutionsmenge berechnen. Laufzeit noch immer exp.

Q: Wie führt man eine Anfrage der Form $KB \models \alpha$ durch?
A: $KB \models \alpha \rightsquigarrow KB \wedge \neg\alpha \rightsquigarrow$ Skolemisierung $\rightsquigarrow$ in Klauseln umwandeln $\rightsquigarrow$ mit Resolution leere Klausel herleiten.

Q: Hier machen sie mal.
A: - Auf Papier skolemisiert, unifiziert und resolviert. -

Q: Unsicherheit. Baysian-Nets/Belief-Networks?
A: Knoten: Ereignisse. Kanten: kausale Zusammenhänge. CPTs. Kompakte Darstellung von Joint-Distributions. Voraussetzende Annahme stochastische Unabhängigkeit.

Q: Gegeben folgendes Netzwerk: (Pfeile nach unten.)



Wie berechnet man denn $P(A \wedge B \wedge \neg C \wedge \neg D)$? (weiß nicht mehr genau wie Formel aussah.)
A: $P(A)P(B|A)P(\neg C|A)P(\neg D|B, \neg C)$

Q: D-Separation?
A: Menge E separiert die Mengen X und Y, wenn die folgenden drei Bedingungen gelten: 3 Bed. erklärt. Man kann aber mit dem Algorithmus nicht alle Unabhängigkeiten erkennen, dafür schnell.

Q: Wir hatten da einen Algorithmus um in solchen Netzen eine Wahrscheinlichkeit, so wie eben am Beispiel, zu berechenen. Was setzte dieser Algorithmus voraus?
A: Singly connected networks, auch polytrees genannt.

Q: Induktives Lernen. Verfahren?
A: DTs, DLs, NNs.

Q: Was sind Vor- und Nachteile von DTs, NNs? (im folgenden eigentlich nur Multilayer Feed-Forward Netze)
A: NN: tolerant gegenüber Rauschen, keine Transparenz; DT: Man kann zu Antwort Herleitung angeben.

Q: Ist jedes NN für verrauschte Daten zu gebrauchen?
A: Nein. Sind zuviele Units im Netz wird nicht vernünftig generalisiert und das Rauschen wird mitgelernt.

Q: Was für eine Funktion lässt sich z.B. mit NNs deutlich besser lernen als mit DTs?
A: Mehrheitsfunktion.

Q: Warum ist das so?
A: Bei DTs findet man keine "guten" Attribute mit denen man die Trainingsmenge unterteilen kann. Sieht man durch Informationstheorie: Alle Attribute haben "gleich schlechten" Informationsgewinn.

Q: Wie aufwendig ist Lernen in einem NN, wie lange dauert es bis das Lernen konvergiert?
A: Exponentielle Anzahl Units. Exponentielle Laufzeit.

Q: Was ist an Laufzeit exponentiell?
A: Back-Propagation Algorithmus arbeitet in Epochen und Anzahl der Epochen kann exponetiell werden.

Q: Ist die Konvergenz dann garantiert?
A: - Fehler - Wenn man die Lernrate der Update-Regel richtig wählt, dann ja.

Q: Dies gilt nur für Perceptrons. Aber wie ist es mit der Backpropagation in Multilayer-Feedforward Netzen?
A: - Erleuchtung - Der Back-Prop.-Lernalgorithmus kann als Gradient Descent Search interpretiert werden. Man hat also das übliche Problem, dass man an lokalen Extrema hängenbleibt, ohne sagen zu können, ob dies ein globales Optimum ist.

Q: Wie kann man das verbessern?
A: Random-Restart. Simulated Annealing.

... hier kam eventuell noch was, habe ich aber leider schon wieder vergessen ...