Prüfungsfragen zur Vorlesung P3

- **Die Fragen dieser Sammlung sind nur Beispiele, es können durchaus andere Fragen gestellt werden.**
- **Antworten müssen nicht exakt in der angegebenen Form erfolgen und können natürlich auch in anderer Form richtig sein.**
- **Es werden keine Fragen zu Teilgebieten gestellt, die in der Vorlesung nicht behandelt wurden.**

Hinweis:

In der Powerpoint-Version dieser Datei lassen sich die Antworten animiert einblenden.

Was versteht man unter dynamischen Datenstrukturen? Nennen Sie Beispiele.

Datenstrukturen, deren Struktur während der Laufzeit verändert werden können

Beispiele:
Listen
Keller (Stack, Stapel)
Schlangen (Queue)
Mengen
Bäume
gerichtete Graphen

Wie kann man eine Liste implementieren? Nach welchen Gesichtspunkten würden Sie eine Implementierungsart wählen?

1. Einfach verkettete Liste
   - unterstützt nur einen Teil der Operationen effizient
   - (nicht: previous)
2. Doppelt verkettete Liste
   - unterstützt alle Operationen effizient
3. Sequentielle Darstellung im Array
   - keine Zeiger erforderlich
   - Umkopieren
   - Verschwenden von Speicherplatz
   - Positionsinformation nicht stabil
4. Einfach oder doppelt verkettete Liste im Array
   - prinzipiell wie 1) und 2)
   - Speicherverwaltung im Array erforderlich

Was ist die Signatur des ADT Schlange?

ADT queue1			
TYPEN	queue, elem		
OPERATIONEN	empty	:	=> queue
	front	: queue	=> elem
	enqueue	: queue x elem	=> queue
	dequeue	: queue	=> queue
	isempty	: queue	=> bool

Entwerfen Sie eine Java-Klasse "Schlange" unter Verwendung eines Arrays. Wie realisieren Sie die Methode enqueue?

Was bedeutet es, wenn ein Algorithmus "von quadratischer Ordnung" ist?

Aufwandsmaß für Algorithmus (z.B. Schrittzahl) wächst quadratisch mit einem Parameter n, der die Problemgröße kennzeichnet.

Definition:

Seien f und g Funktionen natürlicher Zahlen. f ist höchstens von der Ordnung g, geschrieben

$$f = O(g)$$

wenn es positive Konstanten c und n_0 gibt, so daß gilt:

$$\bigwedge_{n>n0} f(n) \leq c \cdot g(n)$$

Wozu dienen Hash-Verfahren? Welche Eigenschaften sollte eine Hash-Funktion haben?

Grundidee von Hashverfahren:

Aus dem *Schlüssel* eines einzufügenden Elementes seine *Speicheradresse* berechnen

Erstrebenswerte Eigenschaften einer Hashfunktion h:

- h ist surjektiv, d.h. erfaßt alle Behälter
- h verteilt gleichmäßig über alle Behälter
- h kann effektiv berechnet werden

Was versteht man unter einer "Kollision" beim Hashing? Was macht man, wenn es dazu kommt?

Wenn mehrere Schlüssel in einen Behälter abgebildet werden, spricht man von *Kollision*.

Geschlossenes Hashing: Behälter kann nur konstante Zahl von Schlüsseln aufnehmen

Bei Kollision => Daten in mehr Behälter umlagern, neue Hash-Funktion

Offenes Hashing: Behälter kann beliebig viele Schlüssel aufnehmen

Was versteht man unter einem stabilen Sortierverfahren? Unter Sortierverfahren mit natürlichem Verhalten? Unter Sortieren in situ?

Ein Sortierverfahren ...

- **ist *stabil*, wenn zwei Einträge mit gleichrangiger Ordnungsrelation ihre Anordnung behalten**
- **hat *natürliches Verhalten*, wenn es bei vorsortierten Daten nicht langsamer ist als bei unsortierten Daten**
- **sortiert *in situ (in place)*, wenn es zum Sortieren nur konstanten zusätzlichen Platz braucht**

Welches Verarbeitungsprinzip steckt hinter einem Divide-and-Conquer-Verfahren? Welche Vorteile können dabei entstehen?

Prinzipieller Divide-and-Conquer-Algorithmus:

Falls die Objektmenge klein genug ist, löse das Problem direkt

Andernfalls

Divide:	Zerlege die Menge in mehrere Teilmengen (wenn möglich gleicher Größe)
Conquer:	Löse das Problem rekursiv für jede Teilmenge
Merge:	Berechne die Gesamtlösung durch Zusammenfügen der Lösungen für die Teilmengen

Beschreiben Sie ein Sortierverfahren, das nach dem Divide-and-Conquer-Prinzip arbeitet.

Algorithmus MergeSort (L) :

Falls |L| = 1 return L

Andernfalls

Divide:

Conquer: L1´:= MergeSort (L1); L2´:= MergeSort (L2)

Merge: return Merge (L1´, L2´)

Kann man eine Liste mit linearem Aufwand sortieren? Wenn ja, wie?

Sortieren mit BucketSort:

- Lege für jeden Schlüssel eine Liste an
- Mache die Listen zu Elementen eines Arrays, der mit dem Schlüssel indiziert werden kann
- Gehe die zu sortierenden Elemente einzeln durch und hänge jedes an die zu seinem Schlüssel gehörige Liste an
- Verbinde die Listen zu einer Ergebnisliste

Eine Bank führt eine Liste über die von ihr vergebenen Kontonummern. Mit welcher Datenstruktur würden Sie die Liste implementieren, um schnell prüfen zu können, ob eine vorgelegte Kontonummer in der Liste enthalten ist?

Implementation als binärer Suchbaum.

Definition:

B ist binärer Suchbaum, falls gilt:

B ist leer oder

- der linke und rechte Unterbaum von B sind binäre Suchbäume,

- ist w die Ordnungszahl der Wurzel, so sind alle Elemente im *linken* Unterbaum *kleiner* als w, alle Elemente im *rechten* Unterbaum *größer* als w.

Bei einem balancierten Binärbaum ist der Suchaufwand O ($\log_2$ N)

Wie können Sie den Ausdruck (a+b)/(c+d) mithilfe eines Binärbaumes in die Präfix- oder Postfix-Notation überführen?

Ablegen in Binärbaum, so daß jeder Operator ein Knoten ist und die zugehörigen Argumente Kinder des Knotens sind.

Rekursive Definitionen:

inorder(left, x, right) = inorder(left) ° [x] ° inorder(right)

preorder(left, x, right) = [x] ° preorder(left) ° preorder(right)

postorder(left, x, right) = postorder(left) ° postorder(right) ° [x]

° = Konkatenationsoperator
[ ] = Listenkonstruktor

Was ist eine Prioritätswarteschlange? Wie kann man sie implementieren?

Prioritätswarteschlangen (priority queues) sind Warteschlangen, deren Elemente (partiell) gemäß einer Priorität eingeordnet werden.

Besondere Operationen:

- **Entnahme des Elementes mit der höchsten Priorität (üblich: mit niedrigstem Zahlenwert)**
- **Einfügen eines neuen Elementes mit beliebiger Priorität**

Implementierung mit einem Heap:

Definition:
Ein Heap ist ein knotenmarkierter Binärbaum, in dem für jeden Teilbaum gilt:
Die Wurzel eines Teilbaums ist das Minimum des Teilbaums.

Heap-Operationen insert und deletemin sind O(log N)

Wie kann man Graphen implementieren?

1. Mithilfe von Adjazenzlisten

- **Jeder Knoten verfügt über eine Liste seiner Nachfolgerknoten**
- **Alle Knoten sind über einen Array direkt zugreifbar**

2. Mithilfe einer Adjazenzmatrix

A_{ij} = true falls Kante von i nach j

Was versteht man unter der Expansion eines Graphen in einen Baum? Wozu ist sie nützlich?

Definition

Die Expansion X(r) eines Graphen in einem Knoten r ist ein Baum mit folgenden Eigenschaften:

- **Falls r keine Nachfolger hat, besteht X(r) nur aus r.**
- **Falls r_1 ... r_k die Nachfolger von r sind, so ist X(r) der Baum (r, X(r_1), ... , X(r_k))**

Durch Expansion eines Graphen in einen Baum kann man Baumbearbeitungsverfahren zur Anwendung bringen (mit kleinen Modifikationen).

Beschreiben Sie die prinzipielle Arbeitsweise des A*-Algorithmus.

Mit dem A*-Algorithmus [Hart et al. 68] expandiert (entwickelt) man einen impliziten Graphen in der Weise, daß ein *kostenoptimaler* Pfad von einem Startknoten zu einem Zielknoten gefunden wird.

Kostenfunktion c bildet Kanten E in nichtnegative reelle Zahlen ab.

Grundidee:

- **Expandiere den Graphen schrittweise**
- **Untersuche zuerst den Nachfolger u mit der günstigsten Bewertung f(u)**
- **Bei Duplizierungen verfolge nur den günstigeren Weg**

Welche Bedingungen muß eine Kostenschätzung erfüllen, damit der A*-Algorithmus den kostengünstigsten Pfad findet?

Satz:

Der A*-Algorithmus terminiert mit dem günstigsten Pfad, wenn die Bewertungsfunktion f(u) = g(u) + h(u) verwendet wird und

- **g(u) = tatsächliche Kosten von s bis u**
- **h(u) ≤ h*(u)**

für alle u gilt. Schätzfunktionen h ≤ h* heißen zulässig.

Nach welchem Prinzip arbeitet der Kürzeste-Pfad-Algorithmus von Dijkstra?

Optimalitätsprinzip: Ist $p = (u_0, u_1, \ldots, u_k)$ kürzester Pfad von Knoten u_0 nach Knoten u_k, so ist $p' = (u_i, \ldots, u_j)$, $0 \le i < j \le k$, ein kürzester Pfad von u_i nach u_j.

Um einen kürzesten Pfad zu finden, kann man also von kürzesten Teilpfaden ausgehen.

Dijkstra´s Idee (1959): Vom Startknoten aus "äquidistante Welle" aussenden, die sukzessiv weitere Knoten erfaßt, bis Zielknoten erreicht ist. Dabei werden 3 Knotenmengen unterschieden:

- für "gewählte Knoten" G ist kürzester Weg vom Startknoten s bekannt

- für "Randknoten" R ist ein Weg von s bekannt

- für "unerreichte Knoten" U kennt man noch keinen Weg

Unter welchen Bedingungen liefert der A*-Algorithmus dasselbe Ergebnis wie der Kürzeste-Pfad-Algorithmus von Dijkstra?

Die Kostenfunktion g(u) entspricht der minimalen Pfadlänge von s bis u, für die Kostenschätzung h(u) von u bis zum Ziel gilt h(u) = 0.

Was ist ein Spannender Baum? Ein minimaler spannender Baum?

1. **Ein zusammenhängender, zyklenfreier Teilgraph eines Graphen, der alle Knoten des Graphen enthält**

2. **... für den zusätzlich die Summe der Kantenbewertungen minimal ist**

Nach welchen Regeln kann man sich beim Berechnen eines Minimalen Spannenden baumes leiten lassen?

Beispiele von Regeln für Kantenauswahl in einem MST-Algorithmus:

Regel 1: Wähle Schnitt, der keine gewählte Kante kreuzt. Wähle *kürzeste* unter den unentschiedenen Kanten, die den Schnitt kreuzen

Regel 2: Wähle einfachen Zyklus, der keine verworfene Kante enthält. Verwirf *längste* unter den unentschiedenen Kanten im Zyklus.

Kann man das Problem des Handelsreisenden (TSP) mit dem A*-Algorithmus lösen?

Im Prinzip ja.

Aber bei N Orten gibt es maximal N! Routen. Suchaufwand des A*-Algorithmus ist quadratisch in der Zahl der Möglichkeiten, also $(N!)^2$

Optimale Lösung des TSP ist NP-vollständig, gilt als nicht traktabel.

Was versteht man unter einem "gierigen" Algorithmus?

Ein Algorithmus, der eine einmal getroffene Entscheidung über die Lösung eines Problems nicht wieder zurücknimmt.

Welche Gesichtspunkte sind beim Entwurf von Datenbanken zu berücksichtigen?

Zu berücksichtigen sind:

- **große Datenbestände** ... im Terabytebereich (10^{12} Bytes)
- **Mehrbenutzerbetrieb** ... verteilter, paralleler Zugriff
- **heterogene Benutzer** ... verschiedenen Sichten
- **Redundanz** ... Modellierungsaufgabe
- **Integritätsverletzungen** ... Integritätsbedingungen überwachen
- **Sicherheit gegen Datenverlust** ... Schutzmechanismen
- **Entwicklungskosten** ... Standards, kommerzielle Systeme

Welche Aufgaben hat ein DBMS?

Das Datenverwaltungssystem (Database Management System, DBMS) vermittelt zwischen einer Benutzersicht der Daten und der internen Datenorganisation:

- **klare Strukturierung der Daten nach außen**
- **spezifische Datensichten für verschiedene Benutzer**
- **Maßnahmen zur Wahrung der Datenintegrität**
- **Flexibilität**
- **Trennung der internen Organisation von der äußeren Struktur**

Beschreiben Sie das Entity-Relationship-Modell für die Ergebnisse der P3-Übungen

Modellierung eines Realweltausschnittes durch Identifikation von

- **Gegenständen, Objekten (entities)**
- **Beziehungen (relationships)**

Objektmengen: **Übungsgruppenleiter, Studierende, Übungstermine, Aufgabenblätter, Lösungsblätter, Punktzahlen, ...**

Beziehungen: **"betreut", "findet-statt-wann", "hat-Punktzahl", ...**

Welche grundsätzlichen Beziehungstypen werden im Entity-Relationsship-Modell unterschieden? Nennen Sie Beispiele.

1:1	Ehepartner-Beziehung bei Monogamie
1:n	Mutter-Kinder-Beziehung
n:1	Beziehung Mensch : Geschlecht (ohne Zwitter)
n:m	Nachfolgerbeziehung zwischen Knoten eines gerichteten Graphen

Erläutern Sie, warum das Aufdecken funktionaler Abhängigkeiten für den DB-Entwurf wichtig ist.

Für eine redundanzfreie Gestaltung von Datenbanksystemen ist es wichtig, funktionale Abhängigkeiten zwischen Attributen (oder Gruppen von Attributen) zu identifizieren.

Mehrstellige Beziehungen

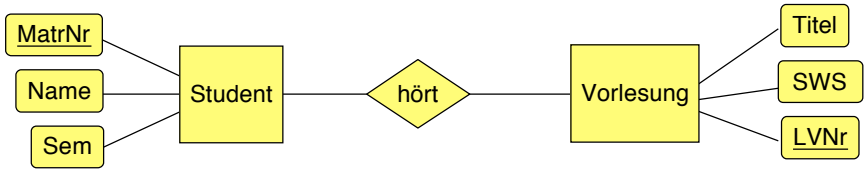
$R \subseteq EM_1 \times EM_2 \times ... \times EM_n$

Funktionalitätsangaben:

$R: EM_1 \times ... \times EM_{k-1} \times EM_{k+1} \times ... \times EM_n \Rightarrow EM_k$

Was versteht man beim Datenbankentwurf unter einem "Schlüssel"?

Ein Attribut oder eine Kombination von Attributen heißt *Schlüssel (Identifikationsschlüssel)*, wenn damit jede Entität einer Entitätsmenge eindeutig identifiziert werden kann.



Was bedeutet die Beziehung "is-a" zwischen Entitätsmengen?

Die Beziehung is-a beschreibt eine *Teilmengenbeziehung* zwischen zwei Entitätsmengen.

Konsequenz:

B is-a A:

Entitäten aus B erben alle Attribute und Beziehungen der Entitäten aus A.

B kann durch zusätzliche Attribute und Beziehungen beschrieben werden.

A ist Generalisierung von B.

B ist Spezialisierung von A.

Was bedeutet die Beziehung "part-of" zwischen Entitätsmengen?

Die Beziehung Teil-von verbindet ein Aggregat mit seinen Teilen.

Konsequenz:

B Teil-von A:

Teil-von (part-of) ist häufig informell anwendbar, hat aber i.A. keine präzise Bedeutung.

Die Attribute und Beziehungen eines Teils hängen nicht zwingend mit denen des zugehörigen Aggregates zusammen.

Nennen Sie wichtige Datenbankmodelle

- **Hierarchisches Datenbankmodell** *historisch*
- **Netzwerk-Datenbankmodell**
- **Relationales Datenbankmodell**
- **Objektorientiertes Datenbankmodell**
- **Deduktives Datenbankmodell**

Beschreiben Sie die grundsätzliche Struktur einer Relationalen Datenbank

Relation:

$R \subseteq D_1 \times D_2 \times \ldots \times D_n$

Benutzer hat logische Sicht in Form von Tabellen

Für jede Relation muß ein *Identifikationsschlüssel* existieren.

Ein Identifikationsschlüssel ist ein Attribut oder eine minimale Attributkombination,

- **die jedes Tupel einer Relation eindeutig identifiziert und**
- **deren Wert sich während der Existenz des Tupels nicht ändert.**

Jedes Attribut einer Relation ist vom Identifikationsschlüssel funktional abhängig.

Kein Attribut aus dem Identifikationsschlüssel ist von den übrigen Attributen des Identifikationsschlüssels funktional abhängig.

Wann ist eine Relation in der 1. Normalform?

Eine Relation befindet sich in 1. Normalform, wenn die Attribute nur einfache (skalare, nicht zusammengesetzte) Attributwerte aufweisen.

Wann befindet sich eine Relation in der 2. Normalform? Wozu dient eine entsprechende Normalisierung?

Eine Relation befindet sich in 2. Normalform, wenn sie

- **in 1. Normalform ist und**
- **jedes nicht zum Identifikationsschlüssel gehörige Attribut funktional vom Identifikationsschlüssel aber nicht von einem anderen Attribut abhängig ist.**

Die 2. Normalform erzwingt Gruppierung in Sachgebiete und eliminiert Redundanzen

Wann befindet sich eine Relation in der 2. Normalform? Wozu dient eine entsprechende Normalisierung?

Eine Relation befindet sich in 3. Normalform, wenn sie

- **in 2. Normalform ist und**
- **kein nicht zum Identifikationsschlüssel gehöriges Attribut *transitiv* (d.h. über andere Attribute) vom Identifikationsschlüssel abhängt.**

Die 3. Normalform beseitigt weitere Redundanzen

Welche Ziele werden bei einer "globalen Normalisierung" einer DB verfolgt?

Eine Datenbasis muß in 3. Normalform sein und nur Lokal- oder Global-Attribute enthalten.

Ein Attribut heißt *global*, wenn es mindestens in einer Relation als Schlüssel vorkommt.

Ein Attribut heißt *lokal*, wenn es nur in einer Relation und dort nicht als Schlüssel vorkommt.

Was sind die Operationen der Relationenalgebra?

Übersicht über die Operationen auf Relationen

Mengenoperationen angewandt auf Relationen:
Vereinigung, Differenz, Durchschnitt, symmetrische Differenz

Erweiterungen für "Relationenalgebra":
Projektion, Theta-Verbund, natürlicher Verbund,, Restriktion, Division

Was versteht man unter der "Projektion" einer Relation? Nennen Sie ein Beispiel.

Die Projektion von R auf eine Attributmenge $A' \subseteq A$ ist die Relation, die durch Weglassen aller Attribute $A_i \notin A'$ entsteht.

Von gleichen Tupeln wird nur ein Exemplar aufbewahrt.

Projektion von	**{Ehefrau, ID1, Ehemann, ID2, Kind, ID3}**
auf	**{Ehefrau, ID1, Ehemann, ID2}**

Was versteht man unter dem "Natürlichen Verbund" zweier Relationen? Nennen Sie ein Beispiel.

Es seien $A = \{A_1, \ldots, A_k\}$, $B = \{B_1, \ldots, B_m\}$, $C = \{C_1, \ldots, C_n\}$ Attributmengen, $A \cup B$ Attribute einer Relation R_1, $A \cup C$ Attribute einer Relation R_2.

Der *natürliche Verbund* der Relationen R1 und R2 ist die Relation R mit den Attributen $A \cup B \cup C$ und den Tupeln aus R1 und R2, die gleiche Werte in A haben.

Verbund von
{Raum, Tag, Zeit, Lehrperson} mit {Raum, Tag, Zeit, LV-Nr} ergibt
{Raum, Tag, Zeit, Lehrperson, LV-Nr}

Was versteht man unter dem "Theta-Verbund" zweier Relationen? Nennen Sie ein Beispiel.

Der Theta-Verbund ist eine Verallgemeinerung des natürlichen Verbundes:

Statt Übereinstimmung wird Erfüllung eines Vergleichsprädikates gefordert.

{MatrNr, Sem, Fach} {Fach, RegelSem}

Vergleichsprädikat: Semesterzahl größer als Regelstudienzeit

{MatrNr, Sem, Fach, RegelSem}

Was versteht man unter einer Restriktion von einer Relation? Nennen Sie ein Beispiel.

Die Restriktion einer Relation R bezüglich zweier vergleichbarer Attribute A_i und A_k enthält diejenigen Tupel von R, für die die Werte der Attribute A_i und A_k eine durch einen Vergleichsoperator beschriebene Bedingung erfüllen.

<u>Beispiel</u>:

Studenten, deren Semesterzahl größer als die Regelstudienzeit ihres Studienfaches ist.

Vorgabe der Bedingung:

Sem > RegelSem

Wie ist eine SQL-Anfrage aufgebaut?

```
select   <Attributliste>
from     <Relation(en)>
where    <Bedingungen>
```

Welche generellen Vorteile bieten Objektorientierte Datenbanksysteme?

Relationale Datenbanken zwingen den DB-Programmierer dazu, komplexe Datenobjekte in "flachen" Tabellen abzulegen.

Objektorientierte Datenbanksysteme (OODBS) ermöglichen

- **durchgängig objektorientierten Entwurf und Implementierung,**
- **Wiederverwendung von DB-Konstrukten,**
- **Modellierung komplex strukturierter Daten.**

Nennen Sie einige Unterschiede zwischen Relationalen und Objektorientierten Datenbanken

Regel 1: Komplexe Objekte
Attribute von strukturierten Objekten können wiederum strukturierte Objekte sein
Regel 2: Objektidentität
Jedes Objekt trägt eindeutige Identität unabhängig von Attributen
Regel 3: Datenkapselung
Verbergen von Implementierungsdetails, Zugriff über Methoden
Regel 4: Typen und Klassen
Bereitstellen von Struktur- und Verhaltensbeschreibungen
Regel 5: Vererbung
Unterstützung der Wiederverwendbarkeit
Regel 6: Polymorphismus
Anwendung derselben Methodennamen auf Objekte unterschiedlicher Klassen

Mit welchen internen Organisationsformen kann ein Datenbankzugriff effektiv gemacht werden?

Grundidee:

Schneller Zugriff auf Blockadressen über Schlüssel des Datums

- **Indexsequentielle Organisation**
- **Balancierte Bäume**
- **Hash-Organisation (Gestreute Speicherung)**
- **Invertierte Dateien**
- **Interne Primärschlüssel**

Was versteht man unter einem Datenbankzugriff mit invertierten Dateien?

Meist wird ein einziger Schlüssel, der *Primärschlüssel*, beim Zugriff auf Datensätze bevorzugt behandelt.

Um weitere Schlüssel, *Sekundärschlüssel*, auch effektiv bedienen zu können, können Hilfsdateien eingerichtet werden, die Sekundärschlüssel auf Primärschlüssel abbilden.

Beispiel: Zugriff auf Namen über Telefonnummer

Mit welchen Mitteln erhöht die RAID-Technologie Sicherheit und Effektivitävon Datenbanksystemen?

RAID Levels 0 - 6 bieten Sicherheits- und Effizienzverbesserungen durch Verteilung der Daten und Sicherheitsinformationen (auch redundante Kopien) in verschiedenen Portionen auf verschiedene Laufwerke.

RAID 0: Striping der Blöcke

Durch blockweise Verteilung auf mehrere Laufwerke werden höhere Datenraten erzielt.

RAID 1: Spiegelung der Blöcke (Mirroring)

Spiegelkopien auf verschiedenen Laufwerken

RAID 2: Bitlevel-Striping und Paritätsbit auf extra Platte

Durch welche Datenbankdienste können Datenbanksysteme "intelligenter" gemacht werden?

- **Wahrung komplexer Konsistenzbedingungen**

  Beispiel: Widersprüchliche Attribute - Mückenfreies Ferienhaus mit Angelgelegenheit

- **Deduktive Fähigkeiten**

  Beispiel: Implizite Eigenschaften explizit machen - z.B. Rentenberechtigung

- **Induktive Fähigkeiten**

  Beispiel: Data Mining - Fehlerursachen finden (Montagsproduktion)

Wie arbeitet ein regelbasiertes System?

Vorwärtsverkettung:

Wiederhole, bis Ziel abgeleitet ist:

- Bestimme Regeln, deren Bedingungen sich mit vorhandenen *Fakten* erfüllen lassen
- Wähle daraus eine Regel aus
- Wende Regel an, etabliere neue Fakten

Rückwärtsverkettung:

Wiederhole, bis Ziel abgeleitet ist:

- Bestimme Regeln, mit denen *Ziele* abgeleitet werden können
- Wähle daraus eine Regel aus
- Wende Regel an, etabliere neue Ziele für nicht erfüllte Bedingungen

Wie ist eine Regel aufgebaut?

Allgemein: **<Bedingungsteil> → <Konsequenzteil>**

Der Bedingungsteil bezieht sich auf Fakten und und ist an ihre jeweilige Repräsentationsform angepaßt. Er enthält Datenmuster mit Konstanten, Variablen und logischen Beziehungen, auf deren Vorkommen in der Datenbasis geprüft wird.

Beispiel (OPS5):

Personen mit gleichem Wohnort finden

```
[P   Finde-Familie [Person  ^Name   <x1>     ^Wohnort    <y>]
                   [Person  ^Name   <x2>     ^Wohnort    <y>]      --> ... ]
```

Was geschieht, wenn in einem Regelbasierten System mehrere Regeln feuern können?

•	alte Fakten (Ziele) bevorzugen	*Breitensuche*
•	neue Fakten (Ziele) bevorzugen	*Tiefensuche*
•	spezielle Regel vor allgemeiner	*spezieller = mehr Bedingungen*
•	Priorisierung von Regeln	*z.B. durch Speicherfolge (PROLOG)*
•	Entscheidung durch Metaregeln	*Regeln für die Regelauswahl*

Welche Vor- und Nachteile bietet die Verwendung von Regeln für "intelligente" Datenbanken?

Vorteile:

- **Regeln passen zu objektorierter Datensicht**
- **Regeln bieten intuitive Formulierungsmöglichkeit**
- **Regeln können vielseitig eingesetzt werden**
- **keine Ablaufsteuerung erforderlich**
- **inkrementell zu entwickeln**

Nachteile:

- **Regeln können inkonsistent sein**
- **Regeln können inkonsistente Daten erzeugen**
- **keine klare Semantik**
- **große Regelmengen (> 1000) sind unübersichtlich**

Welche Erweiterungsmöglichkeiten bietet Datalog für eine Datenbank?

DATALOG ermöglicht deduktive Erweiterung einer relationen Datenbasis.

- **Repräsentation von Fakten und Regeln auf der Basis eines eingeschränkten Prädikatenkalküls**
- **ähnlich wie PROLOG**
- **präzise Definition von *extensionaler* und *intensionaler* DB**

Extensionale Datenbasis (EDB):

"Faktenbasis", explizit gespeicherte Tupel von Relationen (= Prädikate)

Deduktionskomponente

DATALOG-Programm, besteht im wesentlichen aus Ableitungsregeln

Intensionale Datenbasis (IDB):

Virtuelle Relationen, werden mit Deduktionskomponente abgeleitet

Gegeben seien die folgenden Fakten:
Otto ist Bruder von Karl. Karl ist Vater von Fritz.
Skizzieren Sie ein Datalog-Programm, mit dem automatisch abgeleitet wird, daß Otto Onkel von Fritz ist.

```
onkel(X, Y) :- bruder(X, Z), vater(Z, Y).

bruder(otto, karl).
vater(karl, fritz).
```

Welche Schlußregel wird bei einem Widerspruchsbeweis in Datalog ausgenutzt?

Das Resolutionsverfahren beruht auf der Schlußregel:

((A => B) ∧ (B => C)) => (A => C)

Prinzip: Verbinde zwei Klauseln, die dasselbe Literal, aber mit verschiedenem Vorzeichen enthalten

Beispiel: { ¬ancestor(X, anna) } *Klausel 1*
{ ancestor(X, Y), ¬parent(X, Y) } *Klausel 2*

{ ¬parent(X, anna) } *Resolvente*

Nennen Sie einige Inferenzdienste einer Beschreibungslogik

Inferenzdienste:

- Konsistenztest	***Ist eine Beschreibung erfüllbar?***
- Subsumptionstest	***Impliziert ein Konzept eine gegebene Beschreibung?***
- Klassifikation	***Einordnung einer gegebenen Beschreibung in eine Konzepthierarchie***
- Abstraktion	***Gemeinsamkeiten mehrerer Beschreibungen***

Was versteht man unter einer Transaktion? Warum ist dieses Konzept wichtig?

Eine Transaktion ist eine Folge von Aktionen (Anweisungen), die ununterbrechbar ausgeführt werden soll.

Erforderlich, um Probleme durch unerwünschte Verzahnung nebenläufiger Zugriffe (s. Beispiel Kontoführung) zu vermeiden.

Welche Eigenschaften sollen Transaktionen nach dem ACID-Modell haben?

ACID-Paradigma steht für 4 Eigenschaften:

Atomicity (Atomarität)

Eine Transaktion wird als unteilbare Einheit behandelt ("alles-oder-nichts").

Consistency (Konsistenz)

Eine Transaktion hinterläßt nach (erfolgreicher oder erfolgloser) Beendigung konsistente Datenbasis.

Isolation

Nebenläufig ausgeführte Transaktionen beeinflussen sich nicht gegenseitig.

Durability (Dauerhaftigkeit)

Eine erfolgreich abgeschlossene Transaktion hat dauerhafte Wirkung auf die Datenbank, auch bei Hardware- und Software-Fehlern.

Welche Synchronisationsprobleme entstehen bei nebenläufigem Datenbankzugriff?

Konfliktursache im DB-Kontext ist read und write von zwei Prozessen i und k auf dasselbe Datum A:

$read_i(A)$ $read_k(A)$	Reihenfolge irrelevant, kein Konflikt
$read_i(A)$ $write_k(A)$	Reihenfolge muß spezifiziert werden, Konflikt
$write_i(A)$ $read_k(A)$	analog
$write_i(A)$ $write_k(A)$	Reihenfolge muß spezifiziert werden, Konflikt

Was versteht man unter "Synchronisation" bei nebenläufigen Prozessen?

Partielle zeitliche Ordnung zwischen Aktionen nebenläufiger Prozesse festlegen.

Nicht jede partielle Ordnung ist serialisierbar:

Serialisierbarkeitstheorem:
Eine partiell geordnete Menge nebenläufiger Operationen ist genau dann serialisierbar, wenn der Serialisierungsgraph zyklenfrei ist.

Serialisierbarkeitsgraph:
Knoten = atomare Operationen (z.B. read, write)
Kanten = Ordnungsbeziehung (Operation i vor Operation k)

Beschreiben Sie Sperrsynchronisation bei Datenbankzugriffen

Sperranweisungen zur Erzeugung konfliktfreier Abläufe:

- **Sperrmodus S (shared, read lock, Lesesperre)**

Wenn Transaktion T_i eine S-Sperre für ein Datum A besitzt, kann T_i read(A) ausführen. Mehrere Transaktionen können gleichzeitig eine S-Sperre für dasselbe Objekt A besitzen.

- **Sperrmodus X (exclusive, write lock, Schreibsperre)**

Nur eine einzige Transaktion , die eine X-Sperre für A besitzt, darf write(A) ausführen.

Verträglichkeit der Sperren untereinander:
(NL = no lock, keine Sperrung)

	NL	S	X
S	ok	ok	-
X	ok	-	-

Beschreiben Sie das 2-Phasen-Sperrprotokoll

1. Jedes von einer Transaktion betroffene Objekt muß vorher entsprechend gesperrt werden.
2. Eine Transaktion fordert eine Sperre, die sie besitzt, nicht erneut an.
3. Eine Transaktion muß solange warten, bis es eine erforderliche Sperre entsprechend der Verträglichkeitstabelle erhalten kann.
4. Jede Transaktion durchläuft 2 Phasen:
   - in Wachstumsphase werden Sperren angefordert, aber nicht freigegeben
   - in Schrumpfungsphase werden Sperren freigegeben, aber nicht angefordert
5. Bei EOT (Transaktionsende) muß eine Transaktion alle ihre Sperren zurückgeben.

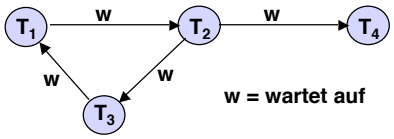
Können bei Durchführung der 2-Phasen-Sperrsynchronisation Verklemmungen auftreten?

Sperrbasierte Synchronisationsmethoden können (unvermeidbar) zu Verklemmungen führen:

Gegenseitiges Warten auf Freigabe von Sperren

Welche Maßnahmen kennen Sie, Verklemmungen zu vermeiden?

1. Zyklen in Wartegraph erkennen



Nach Erkennen eines Zyklus muß Verklemmung durch Zurücksetzen einer geeigneten Transaktion beseitigt werden.

2. **Preclaiming - Vorabforderung aller Sperren**
   Beginn einer Transaktion erst, nachdem die für diese Transaktion insgesamt erforderlichen Sperren erfolgt sind.
   Problem: Vorab die erforderlichen Sperren erkennen
3. **Zeitstempel**
   Transaktionen werden durch Zeitstempel priorisiert. Zurücksetzen statt Warten, wenn T_1 Sperre fordert, T_2 aber Sperre erst freigeben muß:
   - **Strategie Wound-wait: Abbruch von T_2, falls T_2 jünger als T_1, sonst warten**
   - **Strategie Wait-die: Abbruch von T_1, wenn T_1 jünger als T_2, sonst warten**

Was versteht man unter einer verteilten Datenbank? Welche besonderen Probleme sind bei einer verteilten DB zu lösen?

Eine verteilte Datenbank ist eine Sammlung von Informationseinheiten, die auf mehrere über ein Kommunikationsnetz miteinander verbundene Rechner verteilt sind.

1. Fragmentierung

Aufteilung der Informationen auf lokale DB

2. Transaktionskontrolle

Abwicklung einer Transaktion über mehrere lokale DB

Was versteht man unter Fragmentierung einer Relation?

Horizontale Fragmentierung:

Zerlegung einer Relation in disjunkte Tupelmengen

Vertikale Fragmentierung:

Zerlegung einer Relation durch Projektionen

Wie können Sie in einer verteilten Datenbank lokale Sichten unterstützen? Geht das ohne Mehraufwand durch Redundanz?

Zerlegung durch vertikale Fragmentierung:

Zerlegung einer Relation durch Projektionen, um Zugriffe auf Teilmengen der Attribute entsprechend verschiedener Anwendungen zu unterstützen.

Um *Rekonstruierbarkeit* zu gewährleisten, erhält jedes (vertikale) Fragment den Primärschlüssel der Originalrelation (oder ein Surrogat).

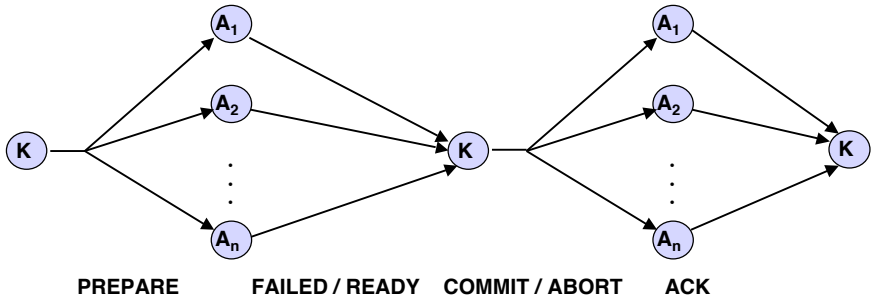
***Redundanz* (d.h. Überlappung von Fragmenten)**

- **ist hinsichtlich des Primärschlüssels notwendig,**
- **kann aus Anwendungssicht nützlich sein,**
- **führt zu Mehraufwand bei Änderungen.**

Wie reagiert eine verteilte Datenbank, wenn während einer Transaktion ein lokaler Rechner ausfällt?

Reaktion entsprechend Zweiphasen-Commit-Protokoll (2PC):

K	**Koordinator**
A_1 ... A_n	**Agenten (Stationen des verteilten Datenbanksystems)**



Was versteht man in der Informatik unter einem Prozess?

<u>Allgemein</u>:

Ein Prozess ist Folge von Vorgängen und Systemzuständen

<u>Informatik</u>:

Prozess	**Ablauf eines sequentiellen Programms**
Zustand eines Prozesses	**Werte expliziter und impliziter Variablen, qualitative Aussagen über Variable**
(atomare) Aktion	**Veränderung eines Zustands durch (unteilbare) Anweisungen**

Erläutern Sie den Unterschied zwischen nebenläufigen, parallelen und quasi-parallelen Aktivitäten.

"Aktivitäten sind *nebenläufig*":

- **Die Aktivitäten können von mehreren Prozessoren ausgeführt werden**
- **Die Aktivitäten können in beliebiger Folge sequentiell von einem Prozessor ausgeführt werden**

"Aktivitäten werden *parallel* ausgeführt":

- **Aktivitäten werden auf mehreren Prozessoren zeitüberlappend ausgeführt**
- **Parallelität ist Spezialfall von Nebenläufigkeit**

"Aktivitäten werden *quasi-parallel* ausgeführt":

- **Aktivitäten werden auf einem Prozessor sequentiell aber ohne vorgeschriebene Reihenfolge ausgeführt**

Zwei nebenläufige Prozesse greifen auf eine Variable X zu, Prozess A inkrementiert einen Zähler mit INCR(X), Prozess B liest den Zählerstand mit READ(X). Kann es zu einer Interferenz kommen?

Nein:

INCR(X) führt in jedem Fall ein korrektes Inkrement durch

READ(X) bekommt den jeweils aktuellen Zählerstand

Zwei nebenläufige Prozesse greifen jeder mit einer Dekrement-Anweisung auf eine Variable X zu. Kann es zu einer Interferenz kommen?

Ja:

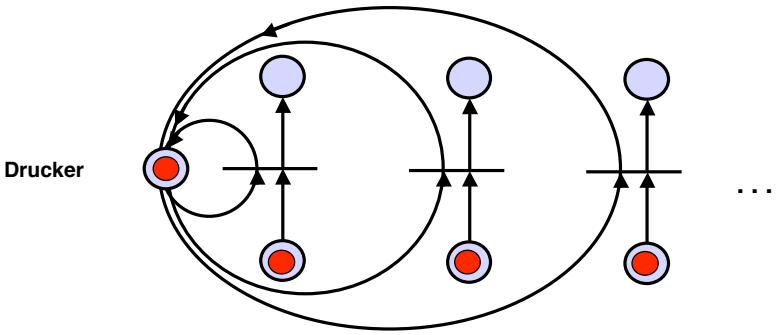
DECR(X) kann sich auf der Ebene von Maschinenbefehlen in FETCH - DECR - STORE zerlegen

unglückliche Verzahnung führt zu Interferenz

Was versteht man unter einem kritischen Abschnitt?

Aktivitäten (Anweisungen), deren Ausführung einen gegenseitigen Ausschluss erfordern, heißen kritische Abschnitte.

Modellieren Sie die Vergabe eines unteilbaren Betriebsmittels (zB Drucker) an N Prozesse mithilfe eines ST-Netzes

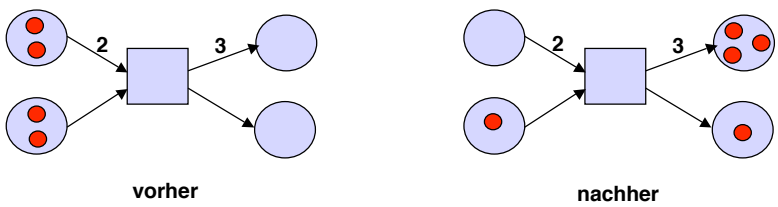


Was bedeuten Gewichtungen an den Kanten eines ST-Netzes?

Kanten können eine *Gewichtung* tragen:

- **Zahl der Marken, die beim Schalten der Transition von einer Eingangsstelle entfernt werden müssen**
- **Zahl der Marken, die beim Schalten der Transition einer Ausgangsstelle zugefügt werden müssen**

Wie sieht das folgende ST-Netz nach dem Schalten aus?



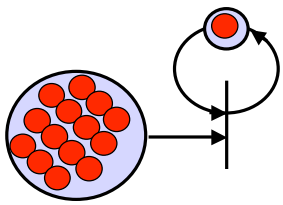
vorher nachher

Was versteht man unter der Kapazität einer Stelle in einem ST-Netz? Was kann man damit beispielsweise modellieren?

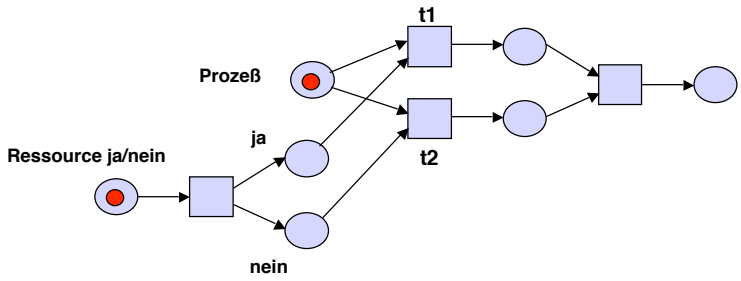
Die *Kapazität* einer Stelle ist die Zahl der maximal aufnehmbaren Marken dieser Stelle. Ohne Angabe ist die Kapazität ∞.

Man kann damit z.B. die Anzahl von Prozessen limitieren, die gleichzeitig einen bestimmten Zustand einnehmen (zB sich eine Ressource teilen).

Modellieren Sie den Verbrauch einer endlichen Ressource (z.B. Druckerpapier) durch ein ST-Netz

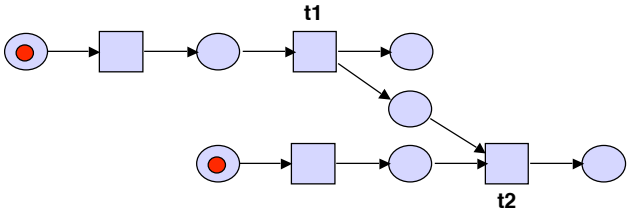


Ein Prozess soll abhängig vom Vorhandensein einer Ressource eine Auswahl treffen. Modellieren Sie dies in einem ST-Netz.



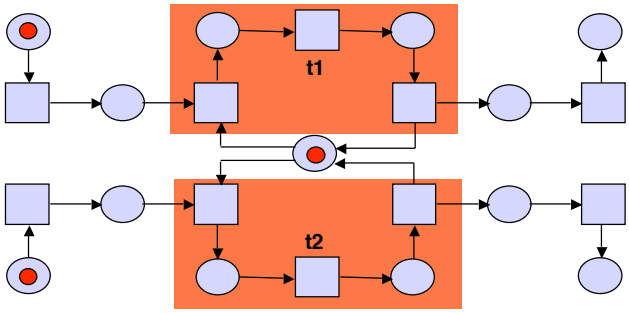
Wie kann man eine einseitige Synchronisierung mit einem ST-Netz modellieren?

Einseitige Synchronisierung t1 → t2



Wie kann man gegenseitigen Ausschluss mit einem ST-Netz modellieren?

Gegenseitiger Ausschluß t1 ↔ t2



kritische Abschnitte

Wie kann es zu einer Verklemmung kommen?

Beispiel: Zweibandproblem

Allgemein: Wenn mindestens eine der 4 notwendigen und hinreichenden Bedingungen verletzt ist

1. Betriebsmittel sind nur exclusiv nutzbar.
2. Betriebsmittel können nicht entzogen werden.
3. Prozesse belegen zugewiesene Betriebsmittel auch dann, wenn sie auf Zuweisung weiterer Betriebsmittel warten.
4. Es gibt zyklische Kette von Prozessen, von denen jeder ein Betriebsmittel besitzt, das der nächste Prozeß in der Kette benötigt.

Wie werden Prozesse durch Pfadausdrücke synchronisiert?

Pfadausdrücke geben einer internen Prozessverwaltung Ablaufmuster von Prozessen vor.

Typische Pfadoperatoren:

a → b	**a muß vor b ausgeführt werden**
a ll b	**a muß zu b nebenläufig ausgeführt werden**
a + b	**Es muß alternativ entweder a oder b ausgeführt werden**
a*	**a darf beliebig oft ausgeführt werden**
N: a	**a darf bis zu N mal nebenläufig ausgeführt werden**

Vergleiche reguläre Ausdrücke für endliche Automaten:
Pfadausdrücke sind mächtiger - warum?

Weil reguläre Ausdrücke keine Nebenläufigkeit zulassen.

Wie können Synchronisationsanforderungen programmiertechnisch befriedigt werden?

Zwei Klassen von Lösungen:

1. Schlossvariable => Aktives Warten (busy-wait)

2. Semaphore und höhere Abstraktionsformen => Prozessverwaltung (process control)

Nach welcher Idee regelt eine Schlossvariable den Zutritt zu kritischen Abschnitten?

Naive Idee: Schlossvariable *locked* ist Schlüssel für kritischen Abschnitt

locked = false	**Schlüssel vorhanden, kritischer Abschnitt offen**
locked = true	**Schlüssel fehlt, kritischer Abschnitt gesperrt**

Was kann schiefgehen, wenn man die Schlüsselbelegung in nebenläufigen Prozessen wie folgt realisiert:

```
...
while (commonLock.isLocked()) { };
commonLock.setLocked(true);
...
```

Lese- und Schreibvorgänge sind nicht ununterbrechbar => Interferenz

Wi kann man mit Schlossvariablen einen beiseitigen Ausschluss realisieren?

Idee:

- **Jeder Prozess hat eigene Schlossvariable, sichtbar auch für anderen Prozess**
- **Gemeinsame Prioritätsvariable löst Vorrangproblem**
- **Betreten des kritischen Abschnittes, wenn die Schlossvariable des anderen Prozesses dies zuläßt und die Prioritätsvariable den Prozess favorisiert**

Was ist ein Semaphor?

Semaphor ist Zähler mit Prozessverwaltungskompetenz: Statt aktivem Warten wird ein Prozess durch ein Semaphor ggf. blockiert und deblockiert.

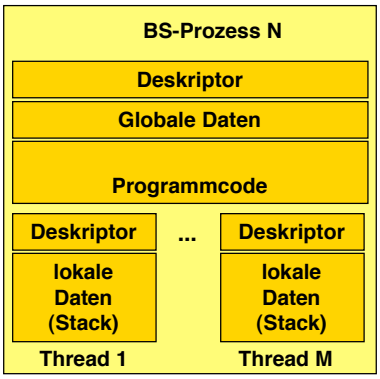
Traditionelle Operationen (Dijkstra 68):

P (passeeren, passieren)	bei Zähler = 0 Prozess blockieren, vor Passage dekrementieren
V (vrijgeven, freigeben, verlassen)	Zähler inkrementieren, wartenden Prozess deblockieren

```
s = new Semaphore(1)

class P1 extends thread {          class P2 extends thread {
...                                ...
s.P();                             s.P();
<kritischer Abschnitt>             <kritischer Abschnitt>
s.V();                             s.V();
...                                ...
}                                  }
```

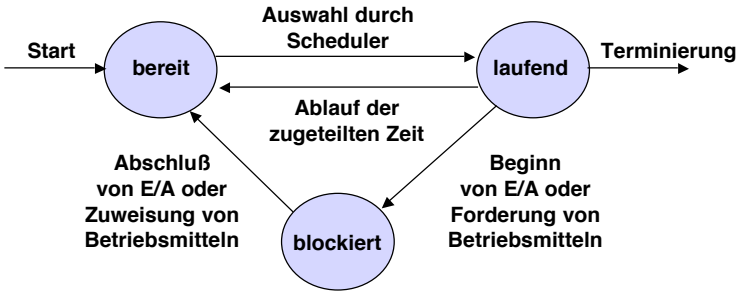
Was versteht man unter einem leichtgewichtigen Prozess?



Leichtgewichtiger Prozess besteht nur aus Deskriptor und lokalen Daten

Programmcode ist in übergeordnetem Prozess

Welche grundsätzlichen Zustände kann ein Prozess aus der Sicht einer Prozessverwaltung haben?

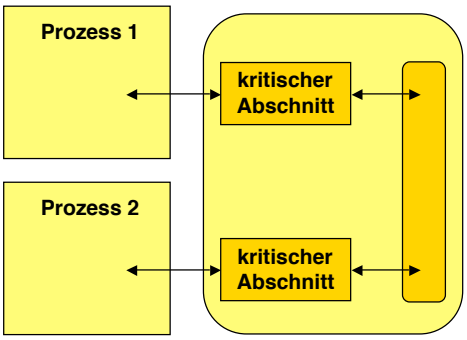


Werden Threads in Java durch den Benutzer oder durch eine interne Prozessverwaltung verwaltet?

In jedem Falle durch die interne Prozessverwaltung, zusätzlich ggf durch Benutzeranweisungen:

- start() *bewirkt Aufruf der Methode* run() *und nebenläufige Ausführung des Threads*
- *Thread terminiert, wenn* run() *terminiert oder* stop() *ausführt*
- *Prädikat* isAlive() *liefert true, wenn Thread gestartet und noch nicht terminiert ist*
- *Laufender (*running*) Thread kann Prozessor durch* yield() *aufgeben*
- *Thread kann durch* suspend() *blockiert (*non-runnable*) und durch* resume() *deblockiert (*runnable*) werden*
- *durch* sleep() *wird Thread auf bestimmte Dauer blockiert (*non-runnable*)*

Was versteht man unter einem Monitor im Kontext nebenläufiger Programme?



gekapselte Datenstruktur zur Realisierung kritischer Abschnitte

Auf ein Lager soll durch Lieferantenprozesse und Abholerprozesse nebenläufig zugegriffen werden. Welche Synchronisierungsaufgaben sind für Lagerobjekte zu realisieren?

(Produzenten-Konsumentenproblem mit anderen Bezeichnungen)

Klasse Lager muss bei Zugriff auf Ware durch Lieferanten und Abholer ...

1. gegenseitigen Ausschluss garantieren

2. zugreifende Prozesse blockieren und deblockieren

(3. über Warenbestand buchführen)

Vergleichen Sie die Aufgaben eines gepufferten Nachrichtenkanals mit denen eines Lagers, auf das Produzenten und Konsumenten zugreifen

Gemeinsamkeiten:

- **Füllen und Leeren des Puffers bei gegenseitigem Ausschluss**
- **Blockieren und Deblockieren bei vollen Puffer**

Unterschiede:

- **Reihenfolge bei Waren i.A. irrelevant**
- **Empfänger von Nachrichten muß bei leerem Puffer nicht notwendigerweise blockiert werden**

Was kennzeichnet einen Software-Agenten?

Ein Software-Agent ist ein Programm, das in einer vernetzten Umgebung selbständig Aufgaben durchführen kann.

Bei welchen Aufgaben kann Agententechnologie nützlich sein?

Komplexe Systeme

Beispiele: Roboter, Produktionsautomatisierung, Katastrophenplanung

- Dekomposition in Agenten bietet Modularisierung und Abstraktion
- Metapher einer Gesellschaft kooperierender Agenten

Offene Systeme

Beispiele: Internet, Reservierungssystem, Auskunftssystem, E-Commerce

- Struktur eines offenen Systems ändert sich dynamisch
- Systemkomponenten sind heterogen
- Kooperation durch Aushandeln (Negotiation) statt Auftragsvergabe

Partnerschaftliche Systeme

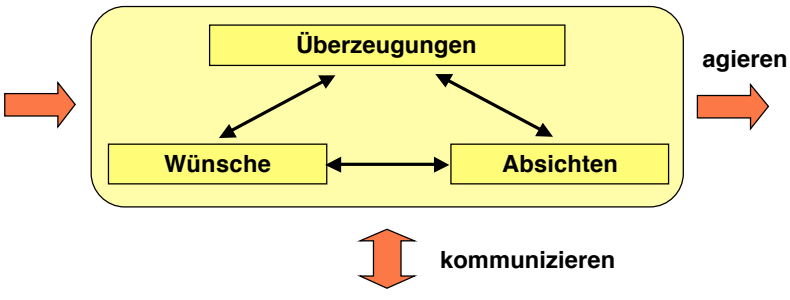
Beispiele: Beratungssysteme, Handelspartner, Manager

- vage, unpräzise Auftragsbeschreibung
- proaktives Verhalten, eigene Vorschläge
- paßt sich an veränderte Aufgaben und veränderte Umgebung an

Beschreiben Sie einen Agenten mit BDI-Architektur

Beliefs = Wissen, Überzeugungen
Desires = Wünsche, (langfristige) Ziele
Intentions = Absichten, Vorhaben

"deliberative agent"
(überlegender, abwägender Agent, BDI-Agent)



Was versteht man unter Planen?

Planen ist das Entwerfen einer Aktionsfolge, mit der eine Startsituation in eine gewünschte Zielsituation überführt werden kann

Ein Agent soll Daten aus Sprache A in eine Sprache B transformieren. Er kennt Transformationsprogramme A-C, A-D, C-E, D-F, E-A, ... , X-B. Wie kann er seine Aufgabe lösen?

Planungsproblem:

Suche Transformationsfolge, die Anfangszustand (Sprache A) in gewünschten Endzustand (Sprache B) überführt.

Lösung z.B. durch Suche im Zustandsraum:

Zustand = Sprache

Operator = Transformationsprogramm

Was versteht man unter dem Qualifikationsproblem beim Planen? Unter dem Rahmenproblem?

Qualifikationsproblem:
Unter welchen Vorbedingungen kann eine Aktion ausgeführt werden?

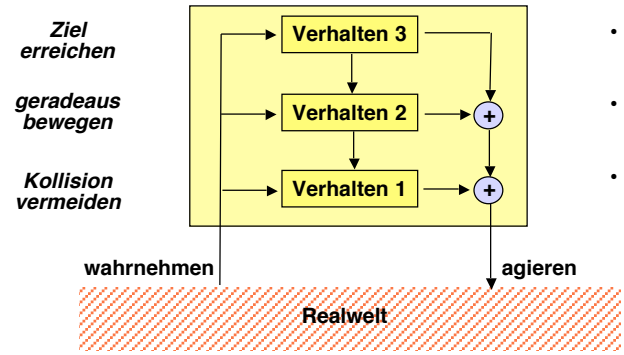
Rahmenproblem (frame problem):
Was sind die Effekte einer Aktion?

Welche Merkmale hat ein reaktives Agentenmodell?

- **Agenten reagieren durch Verhaltensmodule unmittelbar auf ihre Umgebung**
- **kein internes Weltmodell**

Bekanntestes Beispiel:
Subsumptionsarchitektur nach Brooks (1986)

Beispiel:



- **Jedes Verhalten ist (fast) eigenständiger Prozess**
- **Verhalten i "subsumiert" (umfasst) Verhalten i-1**
- **Realwelt übernimmt Rolle eines "Weltmodells"**

Wie können Agenten kooperieren?

1. Agenten kooperieren durch komplementäre Aufgabenverteilung

Gemeinsames Ziel wird implizit erreicht, wenn jeder Agent seine Aufgabe ausführt.

Beispiel: Blackboard-System

2. Agenten verwenden Kooperationsmethoden

Kooperationsmethoden erlauben es, Multi-Agentenpläne zu schaffen und durchzuführen.

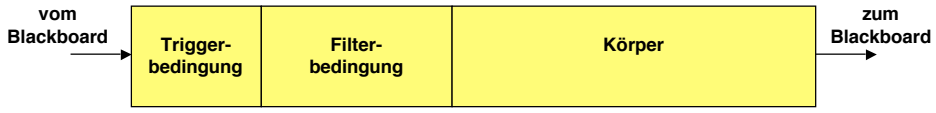
Kooperationsmethoden sind abstrakte, für eine breite Klasse von Anwendungen geeignete Schemata.

Beispiel: Kontraktnetz

Welche Struktur hat ein Blackboard-System?

- **Kommunikation und Koordination über eine Kommunikationstafel (Blackboard)**
- **Agent liest Eingangsdaten vom Blackboard und schreibt Ergebnisse auf das Blackboard**
- **Agent wird tätig, wenn Eingangsdaten bereitstehen**

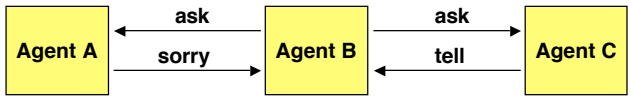
Struktur eines Blackboard-Agenten:



Wozu dient ein Kommunikationsprotokoll in einem Multiagentensystem?

Ein Kommunikationsprotokoll legt Regeln fest, nach denen Sprechakte in einem Dialog aufeinander folgen dürfen.

Beispiel:

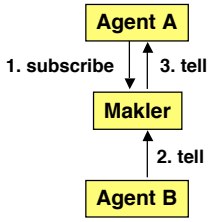


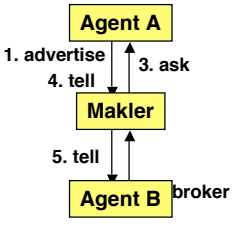
"Sprechakt" **Akt des Nachrichtenaustausches**

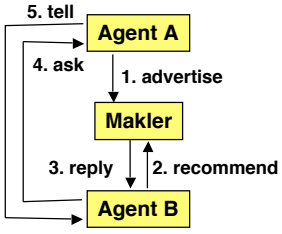
"Sprechakttyp" **Unterscheidung von Sprechakten nach Art des Ziels (z.B. fragen, antworten, anweisen, informieren, ...)**

Beschreiben Sie das Kommunikationsprotokoll für einen Nachrichtenaustausch über einen Makler

3 alternative Formen:

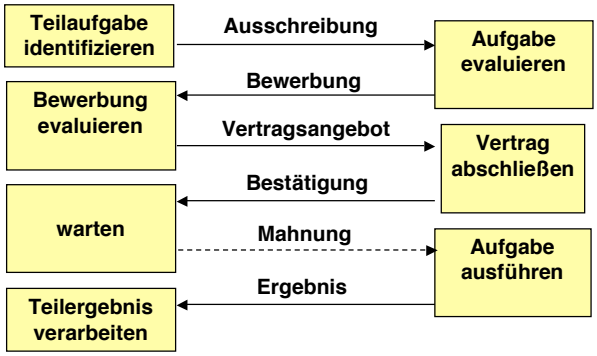






Beschreiben Sie das Kontraktnetz-Kooperationsprotokoll

Schematisierter Verhandlungsablauf zwischen Manager und Agenten zur Vergabe von Teilaufgaben:



Jeder Agent kann seinerseits Teilaufgaben als Manager vergeben

Wie kann man einen Agenten lernen lassen, einen Benutzer beim Umgang mit seiner Email zu unterstützen?

<u>Grundidee</u>:

- **Email-Aktionen des Benutzers anhand von Situations-Aktions-Beschreibungen speichern**
- **Situationen durch zahlreiche Merkmale beschreiben (Sender, Empfänger, Kurzüberschrift, Länge, Schlüsselworte, ...)**
- **Relevanz von Merkmalen lernen und gewichten**
- **Aktionen für neue Situationen durch Vergleich mit gespeicherten Situations-Aktions-Paaren ermitteln (→ Fallbasiertes Schließen)**

Was versteht man unter Fallbasiertem Schließen?

Fallbasiertes Schließen:

Aktion desjenigen gespeicherten Falles F´ vorschlagen, dessen Merkmale mit dem aktuellen Fall F am besten übereinstimmen

Wie ist eine einfache Suchmaschine für Internet-Informationen aufgebaut?

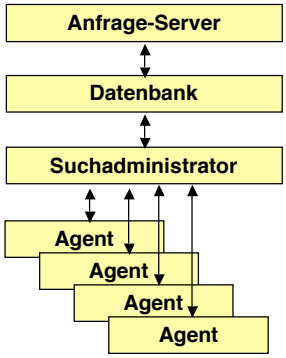
Arbeitsweise

1. Erfassen unbekannter Dokumente durch rekursives Verfolgen von bereits referenzierten Dokumenten

2. Indizieren der Informationen und Speicherung in einer Datenbank

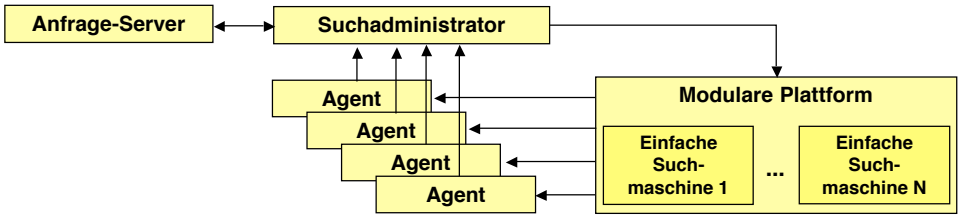
3. Bedienen von Suchanfragen (s. heutige Suchmaschinen)

Architektur



Was versteht man unter einer Meta-Suchmaschine?

Lieferung qualitativ hochwertiger Ergebnisse durch parallele Abfrage der Datenbanken von einfachen Suchmaschinen und Aufbereitung der Einzelergebnisse.



Was versteht man unter einem Mobilen Agenten? Nennen Sie eine denkbare Anwendung.

<u>Stationäre Agenten</u>:

- **Programm läuft auf einem Rechner**
- **Informationsaustausch durch Netzkommunikation**

<u>Mobile Agenten</u>:

- **Laufendes Programm kann Rechner wechseln**
- **Direkte Interaktion mit entfernten Agenten und Informationen**

Anwendungsbeispiel:

Aussuchen von Bildmaterial aus einem entfernten Archiv:

- **Stationärer Agent muss (ggf. viele) Bilder aus Archiv zu lokalem Rechner übertragen, um aussuchen zu können**
- **Mobiler Agent überträgt sich zum Archiv, sucht dort aus und überträgt sich dann mit dem Bild zurück zum lokalen Rechner**

Welche Rolle spielt die Agententechnologie beim E-Commerce?

Software-Agenten übernehmen die Rolle von Anbietern und Nachfragern

- ***Einfache Kaufagenten*** **verschaffen Produktinformationen**
  - **Suche**
  - **Preisvergleich**
- ***Komplexe Kaufagenten*** **unterstützen gesamten Kaufvorgang**
  - **Suche**
  - **Preisvergleich**
  - **Zahlung**
  - **Lieferung**
- ***Agentenbasierte Marktplätze*** **umfassen Kauf- und Verkaufsagenten, Kreditagenten, Zahlungsagenten, Werbeagenten etc.**
  - **Suche**
  - **Werbung**
  - **Preisvergleich**
  - **Verhandlung**
  - **Kreditvergabe**
  - **Zahlung**
  - **Lieferung**