



HOCHSCHULE TRIER

Trier University of Applied Sciences

Informatik - Computer Science

Wohin geht?

Where go?

Boos, Jeremias

Neugebauer, Manuel

Wissenschaftliches Arbeiten

Betreuer: Titel Vorname Name

7. Mai 2014, Trier Abgabedatum

Danksagung

Dank an ... ?Danke für diesen guten Morgen!?

Vorwort

Ein Vorwort ist nicht unbedingt nötig. Falls Sie ein Vorwort schreiben, so ist dies der Platz, um z.B. die Firma vorzustellen, in der diese Arbeit entstanden ist, oder einigen Leuten zu danken, die in irgendeiner Form positiv zur Entstehung dieser Arbeit beigetragen haben. Auf keinen Fall sollten Sie im Vorwort die Aufgabenstellung näher erläutern oder vertieft auf technische Sachverhalte eingehen.

Kurzfassung

A* ist der meistbenutzte Algorithmus in Sachen Wegfindung. Es werden die Funktionen und Berechnungen dahinter erklärt. Auch wird auf die Grundlage und Weiterentwicklung dessen eingegangen.

The same in english (option).

Inhaltsverzeichnis

1	Einleitung	1
2	Problemstellung	2
3	Aufgabenstellung und Zielsetzung	3
4	Übrige Abschnitte (Kapitel und Absätze)	4
5	Latex-Bausteine	5
5.1	Abschnitt	5
5.1.1	Unterabschnitt	5
5.2	Abbildungen und Tabellen	6
5.3	Mathematische Formel	6
5.4	Sätze, Lemmas und Definitionen	7
5.5	Fußnoten	8
5.6	Literaturverweise	8
6	Beispiel-Kapitel	9
6.1	Warum existieren unterschiedliche Konsistenzmodelle?	9
6.2	Klassifizierung eines Konsistenzmodells	10
6.3	Linearisierbarkeit (atomic consistency)	10
	Literaturverzeichnis	12
	Glossar	13
	Erklärung der Kandidatin / des Kandidaten	14

Abbildungsverzeichnis

5.1	Bezeichnung der Abbildung	6
-----	---------------------------------	---

Tabellenverzeichnis

5.1	Bezeichnung der Tabelle	7
6.1	Linearisierbarkeit ist erfüllt	11
6.2	Linearisierbarkeit ist verletzt, sequentielle Konsistenz ist erfüllt.	11
6.3	Linearisierbarkeit und sequentielle Konsistenz sind verletzt.	11

Einleitung

In vielen Computerspielen ist eine Funktion von Nöten um Einheiten und Charaktere ans Ziel zu führen. Egal in welchen Genre, wenn sich der Computer in einer simulierten Umgebung effizient zurecht finden soll, wird ein Wegfindalgorithmus benötigt. Um Umwege und unnatürlich wirkendes Verhalten zu vermeiden, bietet sich ein heuristischer, Graphen basierter Algorithmus an. Dieser hat sich als effizienteste Lösung für Wegfindprobleme erwiesen.

Nicht nur in Civilization, wo die Funktionsweise dem Spieler als ein Zentrales Spielelement bewusst wird, finden solche Algorithmen Verwendung. Auch Genres scheinbar abseits taktischer Tiefe, wie Beispielsweise MMORPGS oder Shootern wären ohne dieselben nicht realisierbar.

Wegen seiner einfachen Implementierung und seines hohen Effizienzgrades hat sich A*(A-Stern) als Geeignetster seiner Art bewährt. Daher wird heute kaum noch auf andere Lösungen zurückgegriffen um entsprechende Probleme zu bewältigen. In so fern wird sich diese Arbeit mit den Eigenschaften und der Umsetzung in Programmen von A* befassen.

Problemstellung

Hier wird i.d.R. zunächst das generell vorliegende Problem diskutiert: Was ist zu lösen - was gibt es bisher an Lösungsansätzen (prinzipiell) und warum ist es wichtig, dass man dieses Problem löst. Letzteres ergibt sich oftmals aus der vorliegenden Anwendungssituation: Man braucht die Lösung, um eine bestimmte Aufgabe zu erledigen, ein System aufzubauen etc. Der Bezug auf vorhandene oder auch bisher fehlende Lösungen begründet auch die Intension und Bedeutung dieser Arbeit. Dies können allgemeine Gesichtspunkte sein - man liefert einen Beitrag für ein generell erkanntes oder zu erkennendes Problem - oder man hat eben eine spezielle Systemumgebung oder Produkt (z.B. in einer Firma u.s.w.), woraus sich dieses noch zu lösende Problem ergibt.

Die genaue Problematik und Randbedingungen werden dann in Kapitel Kapitel 3 dargestellt.

Aufgabenstellung und Zielsetzung

Hier wird nun die Aufgabenstellung konkret dargestellt: Was ist spezifisch zu lösen? Welche Randbedingungen sind prinzipiell gegeben und was ist die Zielsetzung? Letztere soll das beschreiben, was man mit dieser Arbeit (mindestens) erreichen möchte.

Übrige Abschnitte (Kapitel und Absätze)

Die Gliederung hängt natürlich vom Thema und von der Lösungsstrategie ab. Als nützliche Anhaltspunkte können die Entwicklungsstufen oder -schritte z.B. der Softwareentwicklung betrachtet werden. Nützliche Gesichtspunkte erhält und erkennt man, wenn man sich

- in die Rolle des Lesers oder
- in die Rolle des Entwicklers, der die Arbeit z.B. fortsetzen, ergänzen oder pflegen soll,

versetzt. In der Regel wird vorausgesetzt, dass die Leser einen fachlichen Hintergrund haben - z.B. Informatik studiert haben. D.h. nur in besonderen, abgesprochenen Fällen schreibt man in populärer Sprache, so dass auch Nicht-Fachleute die Ausarbeitung prinzipiell lesen und verstehen können.

Die äußere Gestaltung der Ausarbeitung hinsichtlich Abschnittformate, Abbildungen, mathematische Formeln usw. wird in Kapitel 5 kurz dargestellt.

Latex-Bausteine

Der Text wird in bis zu drei Ebenen gegliedert:

1. Kapitel (`\chapter{Kapitel}`),
2. Unterkapitel (`\section{Abschnitt}`) und
3. Unterunterkapitel (`\subsection{Unterabschnitte}`).

5.1 Abschnitt

Text der Gliederungsebene 2.

5.1.1 Unterabschnitt

Text der Gliederungsebene 3. Text Text Text Text Text Text Text Text Text Text
Text Text Text Text Text Beispiel für Quelltext

```
1      Prozess 1:
2
3      Acquire();
4          a := 1;
5      Release();
6      ...
7      Acquire();
8      if (b == 0)
9      {
10         c := 3;
11         d := a;
12     }
13     Release();
```

```

1   Prozess 2:
2
3   Acquire();
4   b := 1;
5   Release();
6   ...
7   Acquire();
8   if (a == 0)
9   {
10      c := 5;
11      d := b;
12  }
13  Release();

```

Größere Code-Fragmente sollten im Anhang eingefügt werden.

5.2 Abbildungen und Tabellen

Abbildung und Tabellen werden zentriert eingefügt. Grundsätzlich sollen sie erst dann erscheinen, nach dem sie im Text angesprochen wurden (siehe Abb. 5.1). Abbildungen und Tabellen (siehe Tabelle 5.1) können im (fließenden) Text (**here**), am Seitenanfang (**top**), am Seitenende (**bottom**) oder auch gesammelt auf einer nachfolgenden Seite (**page**) oder auch ganz am Ende der Ausarbeitung erscheinen. Letzteres sollte man nur dann wählen, wenn die Bilder günstig zusammen zu betrachten sind und die Ausarbeitung nicht zu lang ist (< 20 Seiten).

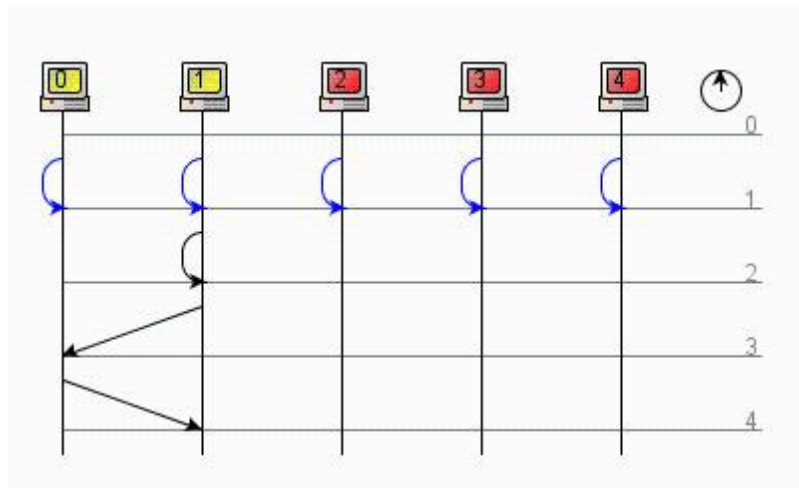


Abb. 5.1. Bezeichnung der Abbildung

5.3 Mathematische Formel

Mathematische Formeln bzw. Formulierungen können sowohl im laufenden Text (z.B. $y = x^2$) oder abgesetzt und zentriert im Text erscheinen. Gleichungen sollten für Referenzierungen nummeriert werden (siehe Formel 5.1).

Prozesse	Zeit \rightarrow
P_1	$W(x)1$
P_2	$W(x)2$
P_3	$R(x)2 \quad R(x)1$
P_4	$R(x)2 \quad R(x)1$

Tabelle 5.1. Bezeichnung der Tabelle

$$e_i = \sum_{i=1}^n w_i x_i \quad (5.1)$$

Entscheidungsformel:

$$\psi(t) = \begin{cases} 1 & 0 \leq t < \frac{1}{2} \\ -1 & \frac{1}{2} \leq t < 1 \\ 0 & \text{sonst} \end{cases} \quad (5.2)$$

Matrix:

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix} \quad (5.3)$$

Vektor:

$$\bar{a} = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix} \quad (5.4)$$

5.4 Sätze, Lemmas und Definitionen

Sätze, Lemmas, Definitionen, Beweise, Beispiele können in speziell dafür vorgesehenen Umgebungen erstellt werden.

Definition 5.1. (*Optimierungsproblem*)

Ein Optimierungsproblem \mathcal{P} ist festgelegt durch ein Tupel $(I_{\mathcal{P}}, sol_{\mathcal{P}}, m_{\mathcal{P}}, goal)$ wobei gilt

1. $I_{\mathcal{P}}$ ist die Menge der Instanzen,
2. $sol_{\mathcal{P}} : I_{\mathcal{P}} \mapsto \mathbb{P}(S_{\mathcal{P}})$ ist eine Funktion, die jeder Instanz $x \in I_{\mathcal{P}}$ eine Menge zulässiger Lösungen zuweist,
3. $m_{\mathcal{P}} : I_{\mathcal{P}} \times S_{\mathcal{P}} \mapsto \mathbb{N}$ ist eine Funktion, die jedem Paar $(x, y(x))$ mit $x \in I_{\mathcal{P}}$ und $y(x) \in sol_{\mathcal{P}}(x)$ eine Zahl $m_{\mathcal{P}}(x, y(x)) \in \mathbb{N}$ zuordnet (= Maß für die Lösung $y(x)$ der Instanz x), und
4. $goal \in \{min, max\}$.

Beispiel 5.2. MINIMUM TRAVELING SALESMAN (MIN-TSP)

- $I_{\text{MIN-TSP}} =_{\text{def}} \text{s.o.}$, ebenso $S_{\text{MIN-TSP}}$
- $\text{sol}_{\text{MIN-TSP}}(m, D) =_{\text{def}} S_{\text{MIN-TSP}} \cap \mathbb{N}^m$
- $m_{\text{MIN-TSP}}((m, D), (c_1, \dots, c_m)) =_{\text{def}} \sum_{i=1}^{m-1} D(c_i, c_{i+1}) + D(c_m, c_1)$
- $\text{goal}_{\text{MIN-TSP}} =_{\text{def}} \min$

□

Satz 5.3. Sei \mathcal{P} ein **NP**-hartes Optimierungsproblem. Wenn $\mathcal{P} \in \mathbf{PO}$, dann ist $\mathbf{P} = \mathbf{NP}$.

Beweis. Um zu zeigen, dass $\mathbf{P} = \mathbf{NP}$ gilt, genügt es wegen Satz A.30 zu zeigen, dass ein einziges **NP**-vollständiges Problem in \mathbf{P} liegt. Sei also \mathcal{P}' ein beliebiges **NP**-vollständiges Problem.

Weil \mathcal{P} nach Voraussetzung **NP**-hart ist, gilt insbesondere $\mathcal{P}' \leq_T \mathcal{P}$. Sei R der zugehörige Polynomialzeit-Algorithmus dieser Turing-Reduktion. Weiter ist $\mathcal{P} \in \mathbf{PO}$ vorausgesetzt, etwa vermöge eines Polynomialzeit-Algorithmus A . Aus den beiden Polynomialzeit-Algorithmen R und A erhält man nun leicht einen effizienten Algorithmus für \mathcal{P}' : Ersetzt man in R das Orakel durch A , ergibt dies insgesamt eine polynomielle Laufzeit. □

Lemma 5.4. Aus $\mathbf{PO} = \mathbf{NPO}$ folgt $\mathbf{P} = \mathbf{NP}$.

Beweis. Es genügt zu zeigen, dass unter der angegebenen Voraussetzung $\text{KNAPSACK} \in \mathbf{P}$ ist.

Nach Voraussetzung ist $\text{MAXIMUM KNAPSACK} \in \mathbf{PO}$, d.h. die Berechnung von $m^*(x)$ für jede Instanz x ist in Polynomialzeit möglich. Um KNAPSACK bei Eingabe (x, k) zu entscheiden, müssen wir nur noch $m^*(x) \geq k$ prüfen. Ist das der Fall, geben wir 1, sonst 0 aus. Dies bleibt insgesamt ein Polynomialzeit-Algorithmus.

□

5.5 Fußnoten

In einer Fußnote können ergänzende Informationen¹ angegeben werden. Außerdem kann eine Fußnote auch Links enthalten. Wird in der Arbeit eine Software (zum Beispiel Java-API²) eingesetzt, so kann die Quelle, die diese Software zur Verfügung stellt in der Fußnote angegeben werden.

5.6 Literaturverweise

Alle benutzte Literatur wird im Literaturverzeichnis angegeben³. Alle angegebene Literatur sollte mindestens einmal im Text referenziert werden[CDK02].

¹ Informationen die für die Arbeit zweitrangig sind, jedoch für den Leser interessant sein könnten.

² <http://java.sun.com/>

³ Dazu wird ein sogenannter bib-File, literatur.bib verwendet.

Beispiel-Kapitel

In diesem Kapitel wird beschrieben, warum es unterschiedliche Konsistenzmodelle gibt. Außerdem werden die Unterschiede zwischen strengen Konsistenzmodellen (Linearisierbarkeit, sequentielle Konsistenz) und schwachen Konsistenzmodellen (schwache Konsistenz, Freigabekonsistenz) erläutert. Es wird geklärt, was Strenge und Kosten (billig, teuer) in Zusammenhang mit Konsistenzmodellen bedeuten.

6.1 Warum existieren unterschiedliche Konsistenzmodelle?

Laut [Mal97] sind mit der Replikation von Daten immer zwei gegensätzliche Ziele verbunden: die Erhöhung der Verfügbarkeit und die Sicherung der Konsistenz der Daten. Die Form der Konsistenzsicherung bestimmt dabei, inwiefern das eine Kriterium erfüllt und das andere dementsprechend nicht erfüllt ist (Trade-off zwischen Verfügbarkeit und der Konsistenz der Daten). Stark konsistente Daten sind stabil, das heißt, falls mehrere Kopien der Daten existieren, dürfen keine Abweichungen auftreten. Die Verfügbarkeit der Daten ist hier jedoch stark eingeschränkt. Je schwächer die Konsistenz wird, desto mehr Abweichungen können zwischen verschiedenen Kopien einer Datei auftreten, wobei die Konsistenz nur an bestimmten Synchronisationspunkten gewährleistet wird. Dafür steigt aber die Verfügbarkeit der Daten, weil sie sich leichter replizieren lassen.

Nach [Mos93] kann die Performanzsteigerung der schwächeren Konsistenzmodelle wegen der Optimierung (Pufferung, Code-Scheduling, Pipelines) 10-40 Prozent betragen. Wenn man bedenkt, dass mit der Nutzung der vorhandenen Synchronisierungsmechanismen schwächere Konsistenzmodelle den Anforderungen der strengen Konsistenz genügen, stellt sich der höhere programmiertechnische Aufwand bei der Implementierung der schwächeren Konsistenzmodelle als ihr einziges Manko dar.

In [Che85] ist beschrieben, wie man sich Formen von DSM vorstellen könnte, für die ein beachtliches Maß an Inkonsistenz akzeptabel wäre. Beispielsweise könnte DSM verwendet werden, um die Auslastung von Computern in einem Netzwerk zu speichern, so dass Clients für die Ausführung ihrer Applikationen die am wenigsten ausgelasteten Computer auswählen können. Weil die Informationen dieser Art innerhalb kürzester Zeit ungenau werden können (und durch die Verwendung der

veralteten Daten keine großen Nachteile entstehen können), wäre es vergebliche Mühe, sie ständig für alle Computer im System konsistent zu halten [CDK02]. Die meisten Applikationen stellen jedoch strengere Konsistenzanforderungen.

6.2 Klassifizierung eines Konsistenzmodells

Die zentrale Frage, die für die Klassifizierung (streng oder schwach) eines Konsistenzmodells von Bedeutung ist [CDK02]: wenn ein Lesezugriff auf eine Speicherposition erfolgt, welche Werte von Schreibzugriffen auf diese Position sollen dann dem Lesevorgang bereitgestellt werden? Die Antwort für das schwächste Konsistenzmodell lautet: von jedem Schreibvorgang, der vor dem Lesen erfolgt ist, oder in der „nahen“ Zukunft, innerhalb des definierten Betrachtungsraums, erfolgten wird. Also irgendein Wert, der vor oder nach dem Lesen geschrieben wurde.

Für das strengste Konsistenzmodell, Linearisierbarkeit (atomic consistency), stehen alle geschriebenen Werte allen Prozessoren sofort zur Verfügung: eine Lese-Operation gibt den aktuellsten Wert zurück, der geschrieben wurde, bevor das Lesen stattfand. Diese Definition ist aber in zweierlei Hinsicht problematisch. Erstens treten weder Schreib- noch Lese-Operationen zu genau einem Zeitpunkt auf, deshalb ist die Bedeutung von „aktuellsten“ nicht immer klar. Zweitens ist es nicht immer möglich, genau festzustellen, ob ein Ereignis vor einem anderen stattgefunden hat, da es Begrenzungen dafür gibt, wie genau Uhren in einem verteilten System synchronisiert werden können.

Nachfolgend werden einige Konsistenzmodelle absteigend nach ihrer Strenge vorgestellt. Zuvor müssen wir allerdings klären, wie die Lese- und Schreib-Operationen in dieser Ausarbeitung dargestellt werden.

Sei x eine Speicherposition, dann können Instanzen dieser Operationen wie folgt ausgedrückt werden:

- $R(x)a$ - eine Lese-Operation, die den Wert a von der Position x liest.
- $W(x)b$ - eine Schreib-Operation, die den Wert b an der Position x speichert.

6.3 Linearisierbarkeit (atomic consistency)

Die Linearisierbarkeit im Zusammenhang mit DSM kann wie folgt definiert werden:

- Die verzahnte Operationsabfolge findet so statt: wenn $R(x)a$ in der Folge vorkommt, dann ist die letzte Schreib-Operation, die vor ihr in der verzahnten Abfolge auftritt, $W(x)a$, oder es tritt keine Schreib-Operation vor ihr auf und a ist der Anfangswert von x . Das bedeutet, dass eine Variable nur durch eine Schreib-Operation geändert werden kann.
- Die Reihenfolge der Operationen in der Verzahnung ist konsistent zu den Echtzeiten, zu denen die Operationen bei der tatsächlichen Ausführung aufgetreten sind.

Prozesse	Zeit \rightarrow
P_1	$W(x)1$ $W(y)2$
P_2	$R(x)1$ $R(y)2$

Tabelle 6.1. Linearisierbarkeit ist erfüllt

Die Bedeutung dieser Definition kann an folgendem Beispiel (Tabelle 6.1) nachvollzogen werden. Es sei angenommen, dass alle Werte mit 0 vorinitialisiert sind.

Hier sind beide Bedingungen erfüllt, da die Lese-Operationen den zuletzt geschriebenen Wert zurückliefern. Interessanter ist es, zu sehen, wann die Linearisierbarkeit verletzt ist.

Prozesse	Zeit \rightarrow
P_1	$W(x)1$ $W(x)2$
P_2	$R(x)0$ $R(x)2$

Tabelle 6.2. Linearisierbarkeit ist verletzt, sequentielle Konsistenz ist erfüllt.

In diesem Beispiel (Tabelle 6.2) ist die Echtzeit-Anforderung verletzt, da der Prozess P_2 immer noch den alten Wert liest, obwohl er von Prozess P_1 bereits geändert wurde. Diese Ausführung wäre aber sequentiell konsistent (siehe kom-mender Abschnitt), da es eine Verzahnung der Operationen gibt, die diese Werte liefern könnte ($R(x)0$, $W(x)1$, $W(x)2$, $R(y)2$). Würde man beide Lese-Operationen des 2. Prozesses vertauschen, wie in der Tabelle 6.3 dargestellt, so wäre keine sinn-volle Verzahnung mehr möglich.

Prozesse	Zeit \rightarrow
P_1	$W(x)1$ $W(x)2$
P_2	$R(x)2$ $R(x)0$

Tabelle 6.3. Linearisierbarkeit und sequentielle Konsistenz sind verletzt.

In diesem Beispiel sind beide Bedingungen verletzt. Selbst wenn die Echtzeit, zu der die Operationen stattgefunden haben, ignoriert wird, gibt es keine Verzahnung einzelner Operationen, die der Definition entsprechen würde.

Literaturverzeichnis

- CDK02. COULOURIS, GEORGE, JEAN DOLLIMORE und TIM KINDBERG: *Verteilte Systeme: Konzepte und Design*. Addison-Wesley-Verlag, 2002.
- Che85. CHERITON, DAVID R.: *Preliminary Thoughts on Problem-oriented Shared Memory: A Decentralized Approach to Distributed Systems*, 1985.
- Mal97. MALTE, PETER: *Replikation in Mobil Computing*. Seminar No 31/1997, Institut für Telematik der Universität Karlsruhe, Karlsruhe, 1997.
<http://www.ubka.uni-karlsruhe.de/cgi-bin/psview?document=/ira/1997/31>.
- Mos93. MOSBERGER, DAVID: *Memory Consistency Models*. Technical Report 93/11, University of Arizona, November 1993.

A

Glossar

DisASter	DisASter (Distributed Algorithms Simulation Terrain) A platform for the Implementation of Distributed Algorithms
DSM	Distributed Shared Memory
AC	Linearisierbarkeit (atomic consistency)
SC	Sequentielle Konsistenz (sequential consistency)
WC	Schwache Konsistenz (weak consistency)
RC	Freigabekonsistenz (release consistency)

B

Erklärung der Kandidatin / des Kandidaten

☐ Die Arbeit habe ich selbständig verfasst und keine anderen als die angegebenen Quellen- und Hilfsmittel verwendet.

☐ Die Arbeit wurde als Gruppenarbeit angefertigt. Meine eigene Leistung ist ...
...

Diesen Teil habe ich selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet.

Namen der Mitverfasser: ...

Datum

Unterschrift der Kandidatin / des Kandidaten