UNIVERSITÄT ZU LÜBECK

Modelle und Algorithmen zur Navigationserkennung innerhalb von Domains

Susan Mielke, 2. April 2011

Diese Bachelorarbeit wurde betreut von
Prof. Dr. math. K. Rüdiger Reischuk

Institut für Theoretische Informatik
Universität zu Lübeck.

Diese Arbeit widme ich meiner Familie.

Kurzfassung

Diese Arbeit befasst sich mit der Erkennung von Navigationen innerhalb von Webdokumenten. Zunächst werden eine eigene Charakterisierung von Navigationen, typische lokale und webdokumentübergreifende Navigationseigenschaften sowie mögliche Anwendungsgebiete der Navigationserkennung vorgestellt. Basierend auf der Charakterisierung der Navigationen und Navigationssysteme wird eine Struktur (*Gruppen*) für die Erkennung von einheitlich dargestellten Hyperlinks innerhalb eines Webdokumentes formal eingeführt und algorithmisch beschrieben. Im Anschluss werden verschiedene Ansätze für die Feststellung von Ähnlichkeiten zwischen Hyperlinks und zwischen Gruppen sowie Modelle und Algorithmen zur Überprüfung der Verlinkungsstruktur dargestellt. Abschließend werden die vorgestellten Modelle und Algorithmen in einem Algorithmus zur Navigationserkennung zusammengefasst und analysiert.

Schlüsselwörter

Navigation, Navigationserkennung, Hyperlink, Linkbasiertes Webseitenranking, HITS

Abstract

This thesis deals with the problem of detecting navigational hyperlinks in webdocuments of the World Wide Web. First of all a characterization of webnavigations will be formulated and typical local and non-local properties and application areas will be discussed. Based on the characterization of webnavigations new structures (*Gruppen*, engl. *groups*) and algorithms for the local detection of uniformly displayed hyperlinks inside the same webdocument will be demonstrated. Subsequent, different approaches for comparing hyperlinks and comparing groups are going to be introduced. In the end all of the discussed algorithms will be combined in one algorithm for detecting navigational hyperlinks.

Keywords

navigation, navigation detection, hyperlink, link analysis, webranking, hits

Inhaltsverzeichnis

1 Einleitung **4**

2 Einführung **6**
2.1 Begriffe und Definitionen 6
2.2 Eigenschaften zur Navigationserkennung 13
2.2.1 Navigationseigenschaften 14
2.2.2 Systemeigenschaften 15
2.3 Anwendung der Navigationserkennung 16
2.3.1 Optimierung des HITS-Algorithmus 16
2.3.2 Automatische Erstellung von Sitemaps für Domains 21

3 Elemente der Navigationserkennung **23**
3.1 Gruppen 23
3.1.1 Formale Beschreibung 23
3.1.2 Beobachtungen 24
3.1.3 Algorithmische Bestimmung von Gruppen 26
3.1.4 Erweiterung 31
3.1.5 Beispiel 33
3.2 Harmonierende Gruppen 36
3.2.1 Ähnlichkeiten von <a href>-Pattern 36
3.2.2 Ähnlichkeiten von Gruppen 40
3.2.3 Algorithmische Überprüfung von x-Harmonien 42
3.3 Verlinkungsstruktur in Navigationssystemen 43
3.3.1 Formale Beschreibung 43
3.3.2 Algorithmische Überprüfung der Verlinkungsstruktur 46

4 Navigationserkennung **50**
4.1 Nachbardokumente und Gruppennachbarschaft 50
4.2 Harmoniegraph und harmonische Kombinationen 52
4.3 Navigation oder Außenseiter 56
4.4 Überblick über die Schritte der Navigationserkennung 57

5 Zusammenfassung und Ausblick **61**

A Literaturverzeichnis **i**

B Abbildungsverzeichnis **iii**

1 Einleitung

Im 21. Jahrhundert spielt das Internet für das gesellschaftliche Leben eine immer größere Rolle. Im Jahr 2009 verfügten bereits 73% der deutschen Haushalte über einen Internetanschluss[13]. Heute ist ein noch höherer Anteil zu erwarten. In der JIM-Studie 2010[14] wird angegeben, dass in allen Familien der 1.208 befragten 12 bis 19-jährigen Jugendlichen ein Internetanschluss zur Verfügung steht[1]. Auch die Internetnutzung steigt kontinuierlich, insbesondere bei Jugendlichen. Mit einer steigenden Anzahl von Nutzern ist auch ein erhebliches Wachstum der Inhalte des World Wide Web[2], einem Informationsnetzwerk aus Webdokumenten und Hyperlinks, zu erwarten – einerseits durch immer einfachere Möglichkeiten der Gestaltung von Inhalten über kleine Nachrichten von wenigen Zeilen (beispielsweise mithilfe von Twitter[3]), Social Communitys, die die Erstellung eigener kleiner Informationsseiten unterstützen, umfangreichen Blogs und vielem mehr und andererseits, weil insbesondere im kommerziellen Bereich das Internet aufgrund der zahlreichen Nutzer zu einem lukrativen Markt heranwächst.

Dabei ist das World Wide Web ein sich selbst organisierendes Netzwerk ohne von Außen vorgegebene Struktur. Dies macht eine präzise und schnelle Suche nach bestimmten Inhalten zu einem großen Problem, das insbesondere von der Firma Google schon seit 1998 aufgegriffen wurde und Google heute zur wertvollsten Marke der Welt macht[15]. Zur Bewältigung dieser Aufgabe existieren heute Suchmaschinen, die in der Regel auf Eingabe eines Suchbegriffs oder Themas eine Reihe von Webdokumenten vorschlagen, die in möglichst engem Bezug zum Suchbegriff stehen. Grundlegend für diese Suchmaschinen sind Algorithmen, die Webdokumente finden und bewerten können. Einer der bekanntesten Ansätze der Bewertung von Webdokumenten ist der 1998 von Sergey Brin und Lawrence Page vorgestellte PageRank[3], auf dem auch die Suchmaschine von Google noch heute aufgebaut sein soll. Die heutigen Algorithmen stellen aber das vermutlich größte Betriebsgeheimnis der Firma dar. Innerhalb des PageRank-Algorithmus werden Webdokumente basierend auf den zwischen ihnen existierenden Hyperlinks bewertet. Diesen Ansatz verfolgt auch Jon M. Kleinberg in dem von ihm etwa zeitgleich vorgestellten HITS-Algorithmus[8]. Beide Algorithmen werden als linkbasierte Webseitenranking-Verfahren bezeichnet, da die grundlegende Annahme sowohl für den PageRank als auch für HITS ist, dass ein Hyperlink von einem Webdokument auf ein anderes eine menschliche Wertung enthält und damit Rückschlüsse auf die Qualität von Webdokumenten ermöglicht. Zahlreiche Hyperlinks innerhalb des World Wide Web sind jedoch beispielsweise navigations- oder werbebedingt und stellen damit Störquellen dar. Aus diesem Grund ist die Erkennung navigationsbedingter Hyperlinks das Thema dieser Arbeit. Neben dieser existieren für linkbasierte Rankingverfahren zahlreiche weitere Arbeiten mit anderen Optimierungsvorschlägen, wie in [2] und [9] für den HITS-Algorithmus und [4] für HITS und PageRank.

[1] Zitat: „Die Grundausstattung der Haushalte, in denen Jugendliche leben, ist sehr hoch. Fernseher, Handy, Computer und Internetanschluss sind in allen Familien vorhanden."

[2] Dieses sollte vom Internet unterschieden werden, das unter anderem aus einem einheitlichen Namensraum, der vom Verzeichnisdienst DNS (*Domain Name System*) verwaltet wird, Protokollen wie HTTP oder HTTPS, die der Übertragung von Dateien dienen, sowie Dokumenten, beispielsweise HTML- oder XHTML-Dokumente, aber auch Videos, Bilder oder Songs und Diensten für e-Mails und Ähnlichem besteht.

[3] http://www.twitter.com

Problembeschreibung

Das Ziel dieser Arbeit ist die Entwicklung von Modellen und Algorithmen zur Erkennung von Navigationen innerhalb von Webdokumenten. Dabei betrachte ich den Fall, dass ein einziges Webdokument als Eingabe gegeben ist und für dieses alle Hyperlinks bestimmt werden sollen, die Navigationen angehören. Da für Navigationen und Navigationssysteme im World Wide Web keine verbindlichen Standards existieren und Webdesigner jederzeit neue Navigationsarten implementieren und online verfügbar machen können, ist es unmöglich, Algorithmen zu entwerfen, die jede Art von Navigationen erkennen. In dieser Arbeit werde ich aufbauend auf einer eigenen Charakterisierung von Navigationen und Navigationssystemen Modelle und einzelne Algorithmen für ihre Erkennung vorstelllen.

Aufbau der Arbeit

In Kapitel 2 werde ich grundlegende Begriffe und Definitionen, meine eigene Charakterisierung für Navigationen und Navigationssysteme, typische Navigations- und Systemeigenschaften und mögliche Anwendungsgebiete der Navigationserkennung vorstellen. Innerhalb des Kapitels 3 stelle ich eigene Modelle und Algorithmen vor, die für die Beschreibung der Navigationserkennung notwendig sind. Abschließend beschreibe ich in Kapitel 4, wie mit Hilfe der in Kapitel 3 vorgestellten Elemente eine Navigationserkennung möglich ist. In Kapitel 5 folgen Zusammenfassung und Ausblick der Arbeit.

2 Einführung

Das Grundlagenkapitel dient als Einführung in die in dieser Arbeit verwendeten Begriffe sowie als Einstieg in die Problematik und mögliche Anwendungsgebiete der Navigationserkennung.

2.1 Begriffe und Definitionen

In dieser Arbeit werde ich zahlreiche neue Begriffe einführen, aber auch eine Reihe von Begriffen verwenden, die Internetnutzern täglich begegnen können. Beispiele für solche vermutlich bekannten Begriffe sind *World Wide Web, Webadresse, Webdokument* oder *Domain*. In diesem Abschnitt werde ich beschreiben, was ich in dieser Arbeit unter den eigentlich gängigen Begriffen genau verstehe.

World Wide Web

Als World Wide Web werde ich im Folgenden ausschließlich HTML- und XHTML-Dokumente (kurz: (X)HTML-Dokumente) und zwischen ihnen existierende Hyperlinks betrachten. (X)HTML-Dokumente werde ich als Webdokumente bezeichnen. Diese werden eindeutig durch eine Webadresse bestimmt.

Definition 1 (Webadresse)
Eine ***Webadresse*** *ist ein String der Form prot :// res, wobei prot das verwendete Übertragungsprotokoll angibt und res die Adresse der gewünschten Ressource.*

Die Definition der Webadresse ist gegenüber dem eigentlichen Schema einer Webadresse (URI) stark vereinfacht. Im Wesentlichen besteht eine URI aus fünf Komponenten: dem verwendeten Protokoll (z. B. *http* oder *https*), dem Namen der Domain, dem Verzeichnispfad (der durch ein Slash eingeleitet wird) zum entsprechenden Webdokument, Parametern (die durch ein Fragezeichen eingeleitet werden) sowie einer Fragmentangabe (die durch eine Raute eingeleitet wird). Der Pfad, die Parameter sowie die Fragmentangabe können auch leer sein. Das Protokoll und die Domain werden durch die Symbole „://“ voneinander abgegrenzt. Die Domain ist ein String, der vom Domain Name System in eine IP-Adresse umgewandelt wird, die die Adresse des Rechners angibt, auf dem die gewünschten Daten hinterlegt sind. Sind diese Daten in einer Verzeichnisstruktur hinterlegt, kann mit Hilfe des Verzeichnispfades auf einzelne Dokumente zugegriffen werden. In einigen Fällen werden ergänzend zum Verzeichnispfad auch Parameter für die Adressierung eines Webdokumentes genutzt. In *res* (siehe Definition 1) sind der Name der Domain, der Verzeichnispfad, die Parameter sowie die Fragmentangabe bereits enthalten.

Wird von einem Browser, also einem Programm, das der Darstellung von Webdokumenten dient, mit Hilfe eines Protokolls wie HTTP (*Hypertext Transfer Protocol*) eine Webadresse angefordert, wird ein entsprechender *Quellcode* geladen, vom Browser interpretiert und damit dem Benutzer angezeigt.

Definition 2 (Quellcode)
Ein Quellcode $C = c_0c_1 \ldots c_n$ ist ein String, der über das Protokoll HTTP oder HTTPS nach Aufruf einer Webadresse übertragen wird. Der String C folgt dabei den Regeln der Auszeichnungssprache HTML oder XHTML.

Mithilfe des Document Object Model (DOM), einem vom World Wide Web Consortium (W3C) entwickelten, plattformunabhängigen Interface, sind ein einheitlicher Zugriff sowie dynamische Änderungen auf HTML- und XML-Quellcodes möglich. DOM bildet Quellcode dabei in einer Baumstruktur ab, in einem sogenannten DOM-Baum. Möglich ist dies durch die baumähnliche Struktur der Quellcodes. Diese Struktur wird durch HTML-Elemente, die Substrings des Quellcodes sind, ermöglicht. Sie bestehen i.d.R. aus einem Start-Tag mit oder ohne Attributen und einem oder keinem End-Tag. Existieren Start- und End-Tag, so kann ein Element Attribute und Kindelemente und reinen Text zwischen Start- und End-Tag besitzen. Existiert ausschließlich ein Start-Tag, kann das Element ausschließlich Attribute besitzen. Die einzelnen Elemente, ihr jeweiliger Aufbau sowie erlaubte Attribute für jede HTML-Version und XHTML sind ebenso wie das DOM-Interface vom W3C auf *http://www.w3.org/* genau spezifiziert.

Welche Bezeichnungen für Elemente des Quellcodes eines Webdokumentes in dieser Arbeit einheitlich verwendet werden, ist in Tabelle 1 aufgeführt. Ist ein HTML-Element mit einem bestimmten Elementnamen gemeint, wird es in dieser Arbeit auch als <*Elementname*>-Element bezeichnet.

Bezeichnung	**Beispiel**
HTML-Element *(auch <table>-Element)*	<table class="middle"><tr><td>Hier steht etwas.</td></tr></table>
Anfangstag	<table class="middle">
Endtag	</table>
Einschluss	<tr><td>Hier steht etwas.</td></tr>
Elementname	table
Attribut	class
Attributwert	middle

Tabelle 1: Übersicht über Bezeichnungen von Elementen eines Quellcodes

Definition 3 (Webdokument)
Ein ***Webdokument*** *ist ein Tupel $d = (w, C_w)$, wobei der String w eine Webadresse und C_w den entsprechenden Quellcode darstellt.*

Definition 4 (Domainname, Domain)
Sei w eine Webadresse. Dann ist der ***Domainname*** *von w der Substring, der nach der Angabe des Protokolls und den Symbolen "://" beginnt und vor dem nächsten Symbol "/" endet. Ist kein weiteres Symbol "/" vorhanden, ist der Domainname der Suffix von w, der nach "://" beginnt. Seien $d_1 = (w_1, C_1)$ und $d_2 = (w_2, C_2)$ Webdokumente. Dann* ***gehören*** *d_1 und d_2 genau dann* ***derselben Domain an****, wenn die Domainnamen von w_1 und w_2 identisch sind.*

Das nach Aufruf der Webadresse vom Browser angezeigte Webdokument nenne ich *Interpretation*. Der Quellcode eines Webdokumentes kann beliebig viele HTML-Elemente mit dem Elementnamen A bzw. a und einem Attribut *href* enthalten. Diese nenne ich in diesem Dokument *<a href>-Pattern*.

Definition 5 (<a href>-Pattern, Verweis, Hyperlink)
Seien $d_\alpha = (\alpha, C_\alpha)$ und $d_\beta = (\beta, C_\beta)$ zwei Webdokumente. Dann ist ein ***<a href>-Pattern*** *p ein Substring innerhalb des Quellcodes C_α der Form*

$$p = \texttt{<a } \gamma_1 \texttt{ href="}\beta\texttt{" } \gamma_2\texttt{>}\delta\texttt{} \text{ oder } \texttt{<A } \gamma_1 \texttt{ href="}\beta\texttt{" } \gamma_2\texttt{>}\delta\texttt{</A>},$$

wobei γ_1 und γ_2 Strings sind, die weitere in <a>-Tags erlaubte Attribute sowie deren Zusweisungen enthalten können, und β eine Webadresse darstellt. Den String δ nennen wir ***Bezeichner*** *von p. Dieser darf weitere HTML-Elemente beinhalten, allerdings kein weiteres <a>- bzw. <A>-Element. Das Webdokument d_α nennen wir* ***Startdokument*** *und das Webdokument d_β* ***Zieldokument*** *von p. Das <a href>-Pattern p heißt auch* ***Verweis*** *auf β bzw. Verweis auf d_β. Ein* ***Hyperlink*** *von d_α nach d_β wird beschrieben durch das Tupel $h = (d_\alpha, d_\beta)$.*

Ein Beispiel für ein <a href>-Pattern ist der String

'<A href="http://www.w3.org/" charset= "ISO-8859-1">W3C Web site</A>'.

Ein weiteres Beispiel für ein <a href>-Pattern ist der String

'',

der ein Bild beinhaltet. <a href>-Pattern, die selbst wieder ein oder mehrere <a href>-Pattern enthalten, sind in HTML und XHTML nicht erlaubt.

Im Folgenden nenne ich die Menge aller Webdokumente d_i, die auf ein Webdokument d_j verweisen, *direkte Vorgängermenge* von d_j. Jedes Webdokument d_i dieser Menge heißt dann *direkter Vorgänger*. Die Menge aller Webdokumente d_k, auf die ein Webdokument d_j verweist, nenne ich *direkte Nachfolgermenge* von d_j und jedes Webdokument daraus *direkten Nachfolger*.

<a href>-Pattern, die von einem Webdokument einer Domain A auf ein Webdokument derselben Domain verweisen, heißen *Intrinsics*, alle anderen *Transverse*. Diese Bezeichnung orientiert sich an [8].

Mithilfe der <a href>-Pattern können Hyperlinks von einem Webdokument zu einem anderen erstellt werden. Neben <a href>-Pattern ist dies aber auch mit Hilfe von <area>-Elementen oder eingebetteten Scripten (z.B. JavaScript) möglich. Diese Möglichkeiten werde ich in dieser Arbeit aber nicht betrachten. Die Algorithmen für die Navigationserkennung werden sich auf die Informationen der Quellcodes der Webdokumente stützen. Eine Analyse eingebetteter Scripte würde den algorithmischen Aufwand unverhältnismäßig erhöhen. Aus diesem Grund werden

Scripte vernachlässigt, was dazu führt, dass durch eingebettete Scripte umgesetzte Navigationen i.d.R. von dem vorgestellten Verfahren nicht erkannt werden können. Der Grund für das Ignorieren der <area>-Elemente ist, dass diese Elemente mithilfe von Koordinaten eine beliebige Platzierung der Hyperlinks ermöglichen. Zueinander gehörende Hyperlinks kann man nur erkennen, indem man diese Koordinaten überprüft und vergleicht. Eine solche Analyse kann man in das vorgestellte Verfahren einbinden, sie wird aber in dieser Arbeit nicht weiter thematisiert.

Navigationsbegriff

Schlägt man in einem Fremdwörterbuch nach, so erfährt man, dass *Navigation* bzw. *navigieren* ihren Ursprung im lateinischen *navigare* haben, das mit *segeln* bzw. *mit dem Schiff fahren* übersetzt werden kann. Heute steht der Begriff allgemein für die Führung von Fortbewegungsmitteln wie Schiffen, Autos und Flugzeugen. Ziel des Navigierens ist dabei immer, ein Objekt möglichst effizient von einer Position ***A*** zu einer anderen Position ***B*** zu führen. In Bezug auf das World Wide Web wird häufig der Begriff Webnavigation verwendet. James Kalbach[7] fasst die drei gängigsten Definitionen für den Begriff Webnavigation folgendermaßen zusammen:

1. *Die Theorie und Praxis, wie sich Menschen im Web von Seite zu Seite bewegen.*
2. *Der Prozess des Browsens im Web, um zielgerichtet nach verlinkten Informationen zu suchen.*
3. *Alle Links, Menübeschriftungen und anderen Elementen, die den Zugriff auf Webseiten ermöglichen und dem Benutzer bei der Orientierung innerhalb einer Website helfen.*[1]

Dabei merkt Kalbach in einer Fußnote zu Punkt 3 an:

Auch die Verlinkung zwischen separaten Websites ist selbstverständlich wichtig, aber der Schwerpunkt dieses Buchs liegt auf der Navigation innerhalb von Websites.

Der für diese Arbeit zugrunde liegenden Beschreibung einer Navigation kommt dabei Punkt 3 am nächsten. Ich werde eine Navigation allerdings etwas anders betrachten. Zunächst nutze ich anstelle der Bezeichnung *Webseite* den bereits eingeführten Begriff *Webdokument* und betrachte anstelle von *Websites Domains*, da eine Domain technisch klarer definiert ist als eine Website. Aus Kalbachs Definition wird außerdem nicht ganz klar, welche *anderen Elemente* er neben Links und Menübeschriftungen meint. Ich werde aus diesem Grund <a href>-Pattern als Elemente von Navigationen betrachten. Kalbach betrachtet alle Elemente, die in irgendeiner Form der Orientierung innerhalb einer Website dienen, als eine einzige Navigation. Das führt dazu, dass eine Website auch entweder genau eine oder gar keine Navigation besitzt.

Betrachte ich jedoch beispielsweise das Webdokument aus Abbildung 1, kann ich mehrere Bereiche ausmachen, die ich als eigene Navigationen bezeichnen würde. Dabei bezeichne ich im Folgenden das, was auf einem einzigen Webdokument wie in Abbildung 1 zu sehen ist, als Navigation und alle Navigationen, die sich auf unterschiedlichen Webdokumenten befinden, miteinander zu interagieren scheinen (Unterpunkte öffnen sich usw.) und eine Einheit zu bilden scheinen, als Navigationssystem. In dem dargestellten Webdokument sind eine horizontal dargestellte Navigation mit den Bezeichnern *Universität, Studium, Forschung, Technologietransfer, Partner* und

Abbildung 1: Ausschnitt des Webdokumentes[19] aus der Rubrik Informatik der Website der Universität zu Lübeck

Aktuelles, eine direkt darunter mit *Startseite* → *Studium* → *Studiengänge* → *Informatik* → *Für Studierende* → *Vor, nach, im Semester* und eine vertikal dargestellte Navigation links mit mehreren Ebenen sichtbar.

Diese schon optisch ganz unterschiedlich dargestellten Elemente möchte ich im Folgenden auch unterschiedlichen Navigationssystemen zuordnen können. Allerdings werde ich die zweite genannte Navigation aus meinen Betrachtungen ausschließen. Es handelt sich um eine sogenannte Breadcrumb-Navigation, die anzeigt, auf welchem logischen Pfad man sich innerhalb der Domain befindet. Sie unterscheidet sich in ihrem Aufbau sehr von den beiden anderen Navigationen – aus diesem Grund werde ich nicht weiter auf sie eingehen. Auch die drittgenannte Navigation bringt eine Besonderheit mit sich. Durch die Einrückungen kann man bereits erkennen, dass sie einem Navigationssystem mit mehreren Ebenen angehört. Tatsächlich wird nach Anwendung der von mir in dieser Arbeit vorgestellten Modelle und Algorithmen jede dieser Ebenen als ein eigenes Navigationssystem erkannt werden. Der Grund ist, dass ich eine einheitliche Darstellung von <a href>-Pattern fordern werde, um beispielsweise das erstgenannte Navigationssystem vom drittgenannten unterscheiden zu können. Diese einheitliche Darstellung werde ich aber so formulieren, dass auch das drittgenannte Navigationssystem in unterschiedliche Navigationssysteme aufgeteilt wird. Dies ist ein unerwünschter Effekt. Ein nächster Schritt, den ich in dieser Arbeit aufgrund des begrenzten Umfangs nicht mehr beleuchtet werde, wäre die Zuordnung verschiedener Navigationen zueinander und die Extraktion der Hierarchie.

Charakterisierung von Navigationen und Navigationssystemen

Im Folgenden beschreibt ein **Navigationssystem** S eine Menge von **Navigationen** N_i, wobei eine Navigation N_i eine Menge von <a href>-Pattern eines einzigen Webdokumentes d_i ist. Ein Webdokument d_i, das eine Navigation aus S enthält, nenne ich im Folgenden *Navigationsdokument* von S. Sei D_S die Menge, die alle Navigationsdokumente von S, enthält, dann verwende ich auch die Formulierung „S *erstreckt sich über* D_S“. Für Navigationen und Navigationssysteme gelten dann folgende Eigenschaften.

1. Die <a href>-Pattern innerhalb einer Navigation N_i werden innerhalb des Quellcodes des Webdokumentes d_i, das N_i enthält, in einer bestimmten Struktur dargestellt. Diese Struktur heißt *Gruppe* und wird in Definition 6 beschrieben.

2. Die <a href>-Pattern einer Navigation N_i besitzen eine feste Ordnung. Diese ergibt sich aus der Reihenfolge, in der die <a href>-Pattern im Quellcode von d_i, das N_i enthält, aufgeführt werden. Die Ordnungen der <a href>-Pattern aller Paare $(N_i,N_j) \in S \times S$, die in N_i und N_j verweisäquivalent sind (siehe Definition 7 *Verweisäquivalenz*) sollte identisch sein (siehe Definition 12 *harmonierende Gruppen*).

3. Jedes <a href>-Pattern einer Navigation, das einen Intrinsic darstellt, zeigt auf ein Navigationsdokument.

4. Zwei verschiedene Navigationen $N_i,N_j \in S$ gehören verschiedenen Webdokumenten an.

5. Ein Navigationssystem S erstreckt sich über eine Webdokumentenmenge D_S, die ausschließlich Webdokumente derselben Domain enthält und für die $|D_S| \geq 2$ gilt.

6. Die Verlinkungen von einer Navigation $N \in S$ auf Navigationsdokumente von S bilden einen *Navigationsgraphen* (siehe Definition 14).

Dabei ist wichtig zu erkennen, dass sich ein Navigationssystem über eine Webdokumentenmenge D_S erstreckt, die nur Webdokumente einer einzigen Domain enthält, es aber dennoch zulässig ist, dass die enthaltenen Navigationen Verweise auf Webdokumente außerhalb der Domain, also Transverse, enthalten.

Außenseiter

Einige Domains enthalten eine Reihe von Webdokumenten, die keine Navigationsdokumente sind, aber scheinbar eine Navigation enthalten. Ein solches Beispiel stellt ein beliebiger Wikipedia-Artikel dar. Auf der linken Seite in Abbildung 2 ist scheinbar eine Navigation zu erkennen, aber das Webdokument, das gerade geöffnet ist, ist nicht Teil des Navigationssystems. D.h. über dieses Navigationssystem kann der Artikel nicht erreicht werden.

Aus diesem Grund ist das dargestellte Webdokument auch kein Navigationsdokument. Dennoch enthält es eine Menge von <a href>-Pattern, die <a href>-Pattern einer Navigation sehr ähneln. Solche Mengen von <a href>-Pattern nenne ich *Außenseiter*, da sie sinnbildlich von außen auf das Navigationssystem zeigen. Dabei müssen Außenseiter alle Eigenschaften erfüllen, die auch eine Navigation erfüllen muss. Die einzige Ausnahme ist, dass sie nicht in die Überprüfung der Verlinkungsstruktur mit einbezogen wird.

Abbildung 2: Ausschnitt des Webdokumentes [17] mit dem Wikipedia Artikel über die Universität zu Lübeck

Navigationslinks

<a href>-Pattern, die von einer Navigation oder einem Außenseiter ausgehen, bezeichne ich im Folgenden als *Navigationslinks*, während ich alle anderen <a href>-Pattern bzw. Hyperlinks grob als *Inhaltslinks* bezeichne (auch wenn es sich möglicherweise um Werbung o.ä. handelt). Dies führt zu der Einteilung von Hyperlinks in Typen, die in Abbildung 3 dargestellt ist.

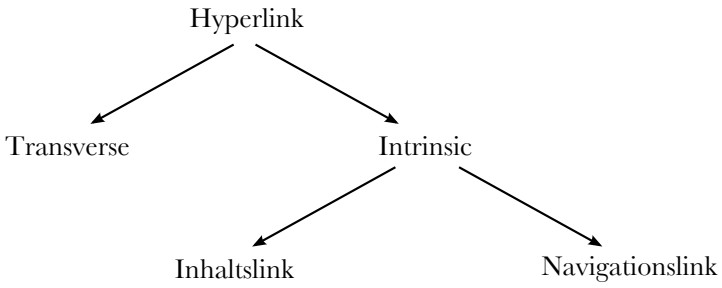


Abbildung 3: Linktypen

Die Unterteilung in Transverse und Intrinsics habe ich aus [8] übernommen. Um Hyperlinks realitätsnah abzubilden, wären sicherlich weitere Linktypen und eine zu Intrinsics äquivalente oder ähnliche Einteilung von Transversen sinnvoll. Eine solche Abbildung ist aber nicht das Ziel meiner Arbeit. Aus diesem Grund verzichte auf eine umfangreichere Einteilung.

2.2 Eigenschaften zur Navigationserkennung

In verschiedenen Handbüchern, die sich mit Webdesign auseinandersetzen, werden zahlreiche Vorschläge erbracht, wie Navigationen aussehen sollten, wo sie platziert werden sollten [1] [11], welche Farbe oder wie viele Elemente optimal seien, wie Verlinkungsstrukturen innerhalb von Domains aussehen können[7] und sollten usw. Einheitliche Standards oder Richtlinien gibt es aber nicht. Die Problematik der Definition einer effektiven, in jedem Fall geeigneten Navigation beschreiben Morvill und Rosenfeld [10] sehr treffend.

In the real world, the boundaries are fuzzy
and the lines get crossed every day.

Dies macht eine Navigationserkennung schwierig. Dennoch weisen zahlreiche Navigationen und lokale Navigationen verschiedener Websites eine Reihe gemeinsamer Eigenschaften auf.

Eigenschaften zur Navigationserkennung unterteile ich in Navigationseigenschaften und Systemeigenschaften. Als Navigationseigenschaften bezeichne ich Eigenschaften, mit denen es möglich ist, bei Betrachtung eines einzigen Webdokumentes zu beurteilen, ob ein <a href>-Pattern mit hoher Wahrscheinlichkeit einer Navigation angehört. Systemeigenschaften beschreiben in dieser Arbeit Eigenschaften, die die Verlinkungen zwischen Webdokumenten, aber auch beispielsweise eine ähnliche Darstellungsart von Hyperlinks unterschiedlicher Webdokumente betreffen und mit dessen Hilfe die Entscheidung, ob es sich bei einem <a href>-Pattern um einen Navigationslink handelt oder nicht, zuverlässiger als nur bei der Betrachtung von Navigationseigenschaften getroffen werden kann. Analysiert wird also webdokumentübergreifend das gesamte System.

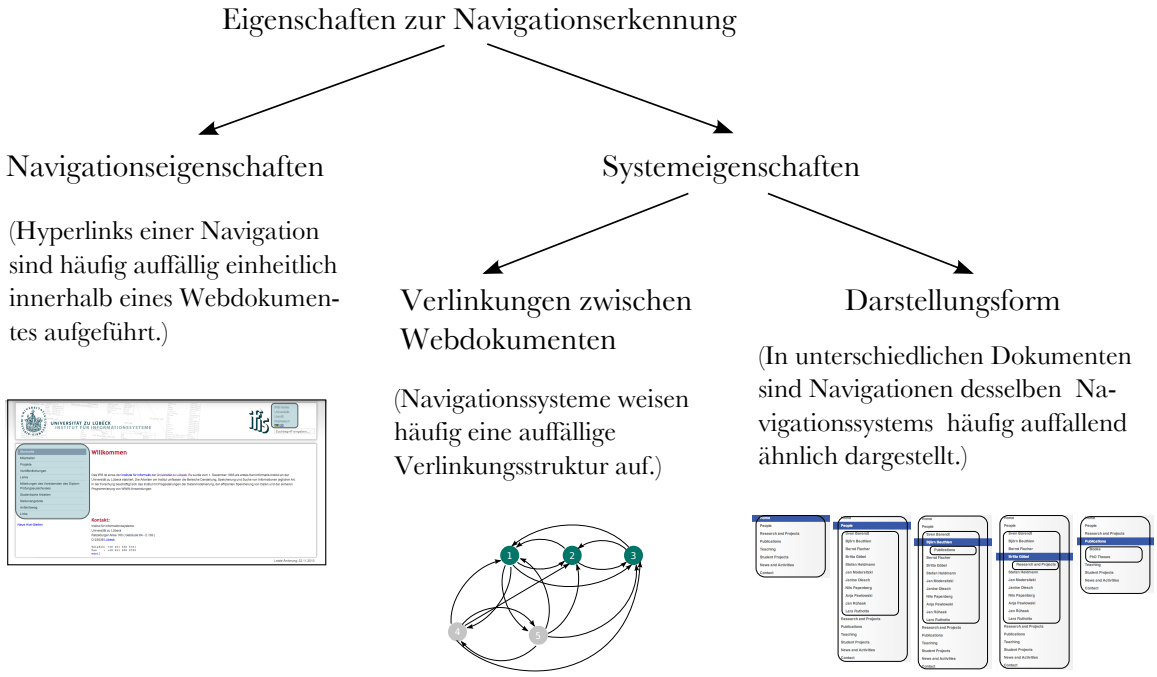


Abbildung 4: Einteilung von Navigationseigenschaften

2.2.1 Navigationseigenschaften

Cai et. al schlugen 2003 in [5] den Algorithmus **Vi**sionbased **P**age **S**egmentation (kurz: VIPS) vor, mit dem Webdokumente in verschiedene semantische Bereiche geteilt werden können. Das ist interessant für die Navigationserkennung, da eine Navigation auch einen eigenen semantischen Teil repräsentiert, also einfach ausgedrückt, eine andere Aufgabe besitzt, als andere Inhalte des Webdokumentes. Für diese Einteilung in Bereiche nutzen Cai et. al allerdings die Interpretation des Webdokumentes. Dabei handelt es sich bei HTML und XHTML um Beschreibungssprachen, die Informationen zur Darstellung von Webdokumenten enthalten, diese Darstellung aber nicht eindeutig festlegen. Die Interpretation des Quellcodes wird dem jeweiligen Browser überlassen. Aus diesem Grund halte ich es nicht für sinnvoll, die Interpretation eines Webdokumentes zu analysieren wie Cai et. al es vorschlagen. Ich orientiere mich stattdessen am Quellcode des jeweiligen Webdokumentes.

Durch stichprobenartige Vergleiche zwischen Webdokumenten und den dazu gehörenden DOM-Bäumen ist mir aufgefallen, dass <a href>-Pattern, die aus meiner Sicht einer Navigation angehören, im DOM-Baum häufig einer auffälligen Struktur zu folgen scheinen. Diese Struktur werde ich im Abschnitt 3.1 beschreiben und einen Algorithmus angeben, der diese Strukturen erkennt und <a href>-Pattern, die einer solchen Struktur angehören, in Gruppen zusammenfasst. Ein Beispiel für Gruppen ist in Abbildung 5 dargestellt.

(a) DOM-Baum (b) Webdokument mit eingerahmten Gruppen

Abbildung 5: DOM-Baum und Interpretation mit eingerahmten Gruppen der Startseite [20] des Instituts für Informationssysteme der Universität zu Lübeck

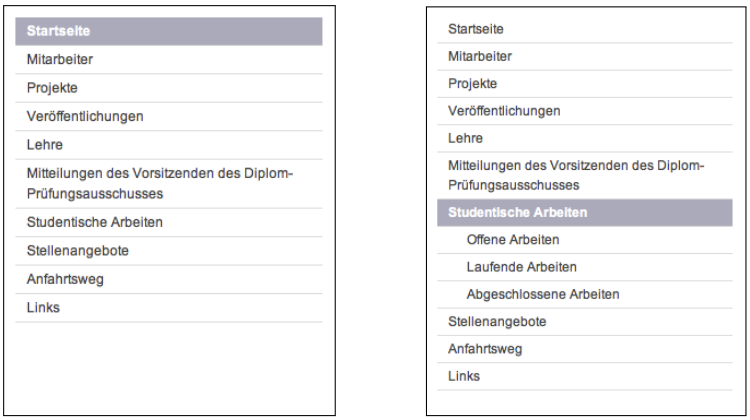
Einer solchen Struktur folgen nicht nur Navigationslinks, sondern beispielsweise auch Aufzählungen und <a href>-Pattern in einem Abschnitt eines Fließtextes. Aus diesem Grund kann eine solche Betrachtung nicht genügen. Navigationseigenschaften jedoch noch genauer festzulegen, führt aufgrund der großen Bandbreite unterschiedlicher Navigationsumsetzungen höchstwahrscheinlich dazu, dass Navigationslinks nicht mehr erkannt werden können. Häufig weisen Navigationen aber typische webdokumentübergreifende Eigenschaften auf.

2.2.2 Systemeigenschaften

Ein Navigationssystem besteht aus mehreren Navigationen, die ein Benutzer als ein einziges veränderliches Objekt wahrnehmen kann. Bildlich gesprochen heißt das, wenn man ausgehend von einem Webdokument auf alle Hyperlinks des Navigationssystems klickt, bleibt scheinbar immer dieselbe Navigation erhalten. Manchmal kommen möglicherweise Unterpunkte dazu und schließen sich bei der Wahl eines anderen Navigationslinks wieder, aber das grundlegende Aussehen - z.B. die Schriftfarbe der Hyperlinks, ihre Position im Webdokument, die Reihenfolge der angezeigten Hyperlinks usw. ändern sich kaum oder gar nicht. Diese Ähnlichkeiten zwischen verschiedenen Navigationen eines Navigationssystems stellen eine Systemeigenschaft dar und fallen unter den Punkt *Darstellungsform*.

Eine weitere Systemeigenschaft stellen die Verlinkungen untereinander dar. Häufig folgen Hyperlinks innerhalb eines Navigationssystems einer festen Struktur.

Hauptnavigation. Eine wesentliche Eigenschaft von Navigationssystemen ist beispielsweise, dass es meistens ein sogenanntes Hauptnavigationssystem gibt, d.h. es existiert in jeder Navigation eine Menge von Hyperlinks, die auf dieselben Webdokumente führen. Innerhalb der Domain *http://www.ifis.uni-luebeck.de* sind dies in der Navigation links beispielsweise die mit *Startseite, Mitarbeiter, Projekte, Veröffentlichungen, Lehre, Mitteilungen des Vorsitzenden des Diplom-Prüfungsausschusses, Studentische Arbeiten, Stellenangebote, Anfahrtsweg* und *Links* bezeichneten Hyperlinks. Egal welchem der Hyperlinks der Navigation man folgt, die im neu geöffneten Webdokument angezeigte Navigation enthält Hyperlinks mit denselben Bezeichnungen auf genau dieselben Webdokumente wie vorher.



(a) Startseite (b) Studentische Arbeiten

Abbildung 6: Navigationen aus Sicht der Startseite [20] sowie aus Sicht des Webdokumentes [21] nach Verfolgen des Hyperlinks mit dem Bezeichner *Studentische Arbeiten*.

Unterpunkte. Ebenso existieren innerhalb von Navigationen häufig Unterpunkte. Vergleicht man die Webdokumente *http://www.ifis.uni-luebeck.de* und *http://www.ifis.uni-luebeck.de/index.php?id=studentische-arbeiten*, kann man beobachten, dass im zweitgenannten Webdokument Hyperlinks auf Webdokumente vorhanden sind, die vorher nicht verlinkt wurden (*Offene Arbeiten,*

Laufende Arbeiten, Abgeschlossene Arbeiten). Diese stellen Unterpunkte des Hauptpunktes *Studentische Arbeiten* dar und werden nur angezeigt, wenn man sich auf dem Webdokument, das der Navigationspunkt *Studentische Arbeiten* verlinkt (also auf dem zweitgenannten Webdokument), einem der neu verlinkten Webdokumente oder aber auf einem Unterpunkt dieser Unterpunkte befindet, sofern sie existieren. Nicht erlaubt ist, dass die Unterpunkte von *Studentische Arbeiten* angezeigt werden, obwohl wir uns auf einem ganz anderen Webdokument der Hauptnavigation, z. B. der Startseite, befinden.

Eigenschaften dieser Art sollen durch die Überprüfung der Verlinkungsstruktur abgedeckt werden.

2.3 Anwendung der Navigationserkennung

Die Navigationserkennung kann verschiedenen Aufgaben dienen. In diesem Abschnitt stelle ich zwei Anwendungsgebiete der Navigationserkennung vor.

2.3.1 Optimierung des HITS-Algorithmus

Beispielhaft werde ich ein Rankingverfahren für Webdokumente beschreiben, für das die Navigationserkennung angewandt werden kann – den von Kleinberg 1998 in [8] vorgestellten HITS-Algorithmus (Hypertext Induced Topic Search). Dieser Algorithmus beschreibt ein linkbasiertes Rankingverfahren für Webdokumente. Das heißt, dass für die Bewertung der Webdokumente vor allem Hyperlinks betrachtet werden. Als Motivation für die Analyse von Hyperlinks als Mittel zur Erkennung qualitativ hochwertiger und zum Suchbegriff relevanter Webdokumente formuliert Kleinberg die zentrale Annahme

> *[...] the creation of a link on the www represents a concrete indication of the following type of judgement: the creator of a page p, by including a link to page q, has in some measure* conferred authority *on q.*

Nach Eingabe eines Suchbegriffes σ, der laut Kleinberg möglichst einem breiten Anfragethema entsprechen sollte, wird innerhalb des HITS-Algorithmus eine Menge von Webdokumenten, die Basismenge, ausgewählt, die dann auf Grundlage einer Analyse der zwischen diesen Webdokumenten bestehenden Hyperlinks bewertet werden. Vor der Bestimmung der zu analysierenden Basismenge wird innerhalb des HITS-Algorithmus zunächst eine sogenannte Rootmenge gebildet. Während für die Bestimmung der Rootmenge auf sehr einfaches Information Retrieval zurückgegriffen wird, indem nur Webdokumente aufgenommen werden, die den Suchbegriff σ enthalten, werden zur Erweiterung der Rootmenge zur Basismenge sowie den Bewertungen aller Webdokumente der Basismenge ausschließlich Hyperlinks verwendet. Die größte Besonderheit an Kleinbergs HITS-Algorithmus liegt jedoch in der Unterscheidung zwischen Authoritys und Hubs, wobei Authoritys Webdokumente repräsentieren sollen, die eine besonders hohe Qualität und Relevanz zur Eingabe aufweisen, während Hubs Webdokumente repräsentieren sollen, die auf gute Authoritys zeigen. Für die Bestimmung der Authoritys wird für jedes Webdokument jeweils ein Authoritywert bestimmt. Dieser ist abhängig von den Hyperlinks innerhalb der betrachteten Webdokumentenmenge und soll unter der Annahme, ein Urheber eines Hyperlinks „bescheinige“

dem Zieldokument Qualität und Relevanz, Aufschluss über seine Eignung als Authority geben. Eine Authority ist dann ein Webdokument mit einem hohen Authoritywert.

Beschreibung des Algorithmus

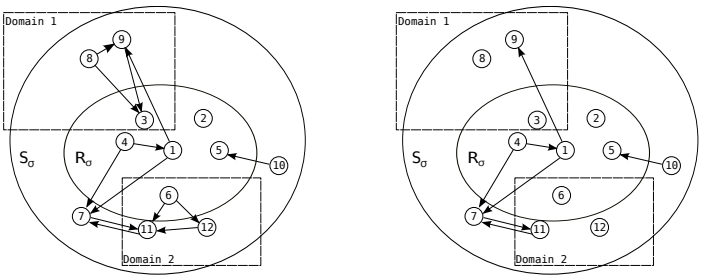
Der HITS-Algorithmus erfolgt in mehreren Schritten. Innerhalb der ersten vier Schritte wird der Graph G_σ bestimmt, der Webdokumente und Hyperlinks enthält, auf denen im fünften Schritt die Berechnungen der Authority- und Hubwerte stattfinden können.

1. Zunächst wird auf Eingabe eines Suchbegriffs die sogenannte Rootmenge R_σ bestimmt: In diese anfangs leere Menge wird eine begrenzte Anzahl von Webdokumenten eingefügt, die eine für den Algorithmus festgelegte textbasierte Suchmaschine auf Eingabe des Suchbegriffs σ ausgibt. Kleinberg schlägt für diese Suchmaschine AltaVista[4] vor.

2. Anschließend wird mit Hilfe der Rootmenge R_σ die Basismenge S_σ bestimmt. In diese werden alle Webdokumente $d_i \in R_\sigma$ und ihre direkten Nachfolgermengen sowie eine begrenzte Anzahl t beliebig gewählter direkter Vorgänger von $d_i \in R_\sigma$ in die Menge S_σ aufgenommen. Für die Wahl von t schlägt Kleinberg den Wert 50 vor, um die Basismenge klein genug für die Berechnungen zu halten, aber groß genug, damit auch wertvolle Seiten mit hoher Wahrscheinlichkeit in die Basismenge aufgenommen werden. Seit 1998 ist die Leistungsfähigkeit von Rechnern deutlich gestiegen. Entsprechen kann man diesen Wert heute deutlich größer wählen.

3. Die Menge S_σ wird in einem nächsten Schritt zu dem Graphen $G_{[S_\sigma]} = (S_\sigma, E)$ umgeformt, wobei die Webdokumente aus S_σ durch Knoten und Hyperlinks durch Kanten repräsentiert werden. Die Menge E enthält alle Hyperlinks, die zwischen Webdokumenten aus S_σ existieren.

4. Der Graph $G_{[S_\sigma]}$ wird anschließend zum Graphen $G_\sigma = (S_\sigma, E_\sigma)$, indem alle Kanten aus E, die Intrinsics repräsentieren, gelöscht werden. Auf diesem Graphen können die Berechnungen der Hub- und Authoritywerte stattfinden.



(a) $G_{[S_\sigma]}$ mit Rootmenge R_σ, Basismenge S_σ und Hyperlinks

(b) G_σ mit Rootmenge R_σ, Basismenge S_σ und ausschließlich Transversen

Abbildung 7: Darstellung eines Webgraphen $G_{[S_\sigma]}$ und G_σ.

[4]http://www.altavista.com, wobei dem Wall Street Journal Digital Network[16] in einem Bericht vom 17. Dezember 2010 zu entnehmen ist, dass Yahoo! plant, AltaVista einzustellen.

Den Schritt 4 nimmt Kleinberg vor, um den Einfluss von Hyperlinks einzudämmen, die einer Navigation angehören, da er der Ansicht ist, dass sehr viele Intrinsics innerhalb eines Webgraphen Navigationslinks repräsentieren und generell deutlich weniger Schlüsse als Transverse darüber zulassen würden, welchen qualitativen Wert oder Themenbezug ein verlinktes Webdokument besitzt.[5] Zusätzlich schlägt Kleinberg vor - ohne näher auf diesen Vorschlag einzugehen - die Menge der Hyperlinks von Webdokumenten einer Domain auf ein einzelnes Webdokument auf einen festen Wert m zu beschränken, also maximal m Kanten zu erlauben, die aus einer Domain auf dasselbe Webdokument zeigen. Dies soll den Einfluss automatisch generierter Links wie „This site is designed by ...“ eingrenzen. Für m schlägt Kleinberg einen Wert zwischen 4 und 8 vor. Dabei wird nicht klar, ob m Hyperlinks erhalten bleiben - und wenn ja, welche - oder ob alle Kanten mit dieser Eigenschaft gelöscht werden sollen.

5. Nachdem der Graph $G_\sigma = (S_\sigma, E_\sigma)$ bestimmt wurde, werden für jedes Webdokument $d \in S_\sigma$ Authority- und Hubwerte bestimmt. Zunächst wird jedem Webdokument der Authority- und Hubwert 1 zugewiesen, also

$$\forall d_i \in S_\sigma: \quad \text{Authoritywert } x^{\langle d_i \rangle} = \text{ Hubwert } y^{\langle d_i \rangle} = 1,$$

und dann der Authoritywert neu bestimmt, indem für jedes Webdokument $d_i \in S_\sigma$ die Hubwerte der Webdokumente $d_j \in S_\sigma$ summiert werden, für die eine Kante von d_j nach d_i existiert. Anschließend wird für jedes Webdokument d_i der Hubwert neu berechnet, indem die Authoritywerte der Webdokumente d_j, für die eine Kante von d_i nach d_j existiert, summiert werden. In mehreren Iterationen werden die Authority- und Hubwerte aktualisiert, indem in jedem Iterationsschritt erneut für jedes Webdokument $d_i \in S_\sigma$ die Hubwerte der Webdokumente $d_j \in S_\sigma$ für die Bestimmung des neuen Authoritywertes, summiert werden. Analog werden auch die Hubwerte aktualisiert. Die Berechnung der Authoritywerte erfolgt in jeder Iteration durch die Operation $\mathscr{I}$ und für die Berechnung der Hubwerte durch die Funktion $\mathscr{O}$.

$$\mathscr{I}: x^{\langle d_i \rangle} \leftarrow \sum_{d_j:(d_j,d_i)\in E} y^{\langle d_j \rangle} \quad \text{(Berechnung der Authoritywerte)}$$

$$\mathscr{O}: y^{\langle d_i \rangle} \leftarrow \sum_{d_j:(d_i,d_j)\in E} x^{\langle d_j \rangle} \quad \text{(Berechnung der Hubwerte)}$$

Um für Invarianz zu sorgen, werden beide Gewichte nach jeder Iteration normalisiert, so dass die Summe ihrer Quadrate 1 ergibt.

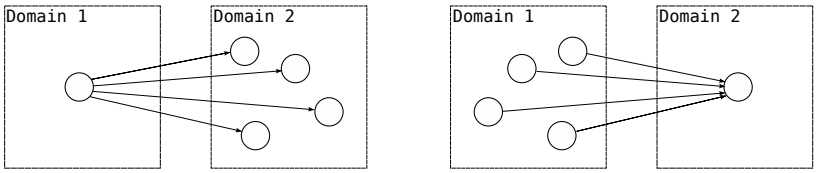
$$\sum_{d_i \in S_\sigma} (x^{\langle d_i \rangle})^2 = 1 \quad \text{und} \quad \sum_{d_i \in S_\sigma} (y^{\langle d_i \rangle})^2 = 1.$$

Auch die Anzahl der Iterationen muss festgelegt werden. Experimentiell hat Kleinberg herausgefunden, dass sich die Werte in der Regel nach etwa 20 Iterationen nicht mehr maßgeblich ändern. Nachdem die Iterationen abgeschlossen sind, werden die Webdokumente mit den höchsten Authoritywerten sowie die mit den höchsten Hubwerten als Authoritys und Hubs ausgegeben.

[5] „Since intrisic links very often exist purely to allow for navigation of the infrastructure of a site, they convey much less information than transverse links about the authority of the pages they point to.“ [8]

Optimierungen

Für den HITS-Algorithmus wurden zahlreiche Optimierungen vorgeschlagen. So beschreiben beispielsweise Bharat und Henzinger in [2] verschiedene Optimierungsideen, insbesondere die Einführung von Gewichten für Kanten und Relevanzgewichten für Knoten. Die Kantengewichte sollen bewirken, dass alle Webseiten einer Domain den gleichen Einfluss auf ein Webdokument haben, wie ein einzelnes Webdokument hätte. Damit soll das *Mutually Reinforcing Relationship*-Problem gelöst werden. Bharat und Henzinger formulieren dieses Problem sinngemäß so: Zeigt ein einzelnes Webdokument einer Domain 1 auf eine Webdokumentenmenge einer anderen Domain 2, führt dies für das Webdokument der Domain 1 zu einem stetig steigenden Hubwert, während die Webdokumente der zweiten Domain einen stetig steigenden Authoritywert erhalten (Variante 1). Analog tritt das Problem dann auf, wenn mehrere Webdokumente einer Domain 1 auf ein einzelnes Webdokument einer Domain 2 verweisen (Variante 2). So hat, wie Bharat und Henzinger kritisieren, die Meinung eines einzigen Autors oder einer Organisation einen zu großen Einfluss auf die Bewertung der Webdokumente.



(a) Variante 1 (b) Variante 2

Abbildung 8: Mutually Reinforcing Relationship

Die Einführung der Relevanzgewichte für Knoten soll dagegen das *Topic Drift*-Problem dämpfen. Der Topic Drift ist ein Problem, das entsteht, wenn nach einem Begriff eines bestimmten Themas gesucht wird, die am besten bewertete Seite mit diesem Thema aber nichts mehr zu tun hat. Aus diesem Grund definieren Bharat und Henzinger eine neue Suchanfrage Q in Abhängigkeit von σ und S_σ. Zusätzlich soll diese Maßnahme den Einfluss automatisch generierter Hyperlinks dämpfen[2]. Das Relevanzgewicht eines Webdokumentes d entspricht dabei seiner Ähnlichkeit zur Suchanfrage. Anstelle der Suchanfrage σ werden die ersten 1000 Wörter aller Webseiten der Rootmenge R_σ konkateniert und als Suchanfrage Q definiert. Diese wird mit dem Webdokument d verglichen. Dazu dient ein Ähnlichkeitswert zwischen einem Webdokument d und der Suchanfrage Q, der unter anderem mit Hilfe einer im Vorfeld berechneten IDF (*inverse document frequency*), die eine Schätzung für die Häufigkeit eines Wort i im World Wide Web angibt, sowie der Frequenz des Wortes i im betrachteten Webdokument, bestimmt wird. Die genaue Berechnungsvorschrift der Relevanzgewichte ist in [2] angegeben. Nach Bharat und Henzinger können diese Gewichte verwendet werden, um Knoten aus der Basismenge zu eliminieren, die für das Anfragethema σ nicht relevant sind.

Kleinberg sowie Bharat und Henzinger stützen sich in [8], [2] auf die Annahme, der Urheber eines Webdokumentes d_i, das einen Hyperlink auf ein Webdokument d_j enthält, gewähre eine gewisse Maßgeblichkeit für d_j. Weiterhin gehen sie davon aus, dass Webdokumente, die einen gemeinsamen direkten Vorgänger d_i besitzen, mit einer hohen Wahrscheinlichkeit in Bezug zu

demselben Thema ständen. Wie Cai et al. in [4] anmerken, gelten diese Annahmen jedoch in vielen Fällen nicht, da sich innerhalb eines Webdokumentes häufig Bereiche unterschiedlicher Bedeutungen befinden, die im Allgemeinen eine unterschiedliche Qualität und Relevanz zum gewählten Suchstring aufweisen. Um dieses Problem zu beheben, schlagen Cai et al. vor, Webdokumente in mehrere Blöcke zu unterteilen und diese Blöcke als kleinste Einheit zu betrachten. Aufbauend auf dieser Idee entwickelten Cai et al. unter anderem den Block Level HITS-Algorithmus (BLHITS). Der BLHITS-Algorithmus ähnelt dem HITS-Algorithmus, arbeitet aber auf einem anders modellierten Graphen.

Eigener Optimierungsvorschlag mit Hilfe der Navigationserkennung

Kleinberg stellte fest, dass sehr viele Intrinsics Navigationslinks, also Hyperlinks, die lediglich der Navigation innerhalb der Infrastruktur einer Website dienen, repräsentieren und Intrinsics darüber hinaus deutlich weniger Schlüsse als Transverse darüber zulassen würden, welchen qualitativen Wert oder Themenbezug das verlinkte Webdokument besitzt. Aus diesem Grund löscht er alle Intrinsics, die nach der Erweiterung der Rootmenge in der Basismenge auftreten. Dieses Vorgehen halte ich aus verschiedenen Gründen nicht für sinnvoll und schlage folgende Änderungen mit Hilfe einer Navigationserkennung vor.

Nichtzulassung von Navigationsdokumenten zur Basismenge. Während Kleinberg für jedes Webdokument *d* aus der Rootmenge alle Zieldokumente der Hyperlinks innerhalb von *d* mit in die Basismenge aufnimmt, schlage ich vor, das Zieldokument nicht aufzunehmen, wenn es ausschließlich als Navigationsdokument Einzug in die Basismenge finden würde. Insbesondere wenn es sich um ein bekanntes und häufig verlinktes Webdokument handelt, könnte dies zu einem Topic Drift führen. Wird beispielsweise als mögliches Anfragethema „Twitter Lübeck kämpft“ angegeben und *www.google.de/search? &q=Lübeck+kämpft&tbs=mbl:1* als Übersicht über Ergebnisse zum Thema „Lübeck kämpft“ in Echtzeitdiensten (u.a. Twitter[6]) in die Rootmenge mit aufgenommen, so könnte nach Kleinbergs Vorgehensweise auch das Webdokument *www.google.de* Einzug in die Basismenge finden, da *www.google.de/search? &q=Lübeck+kämpft&tbs=mbl:1* einen Verweis auf *http://www.google.de* enthält – allerdings als <a href>-Pattern einer möglichen lokalen Navigation. Letztendlich kann *www.google.de* eine hohe Bewertung erhalten, wenn es auch von anderen Dokumenten, die in die Basismenge aufgenommen wurden, häufig verlinkt wird. Es kann dann als Authority ausgegeben werden, dem Nutzer aber im schlechtesten Fall nicht einmal klar sein, welchen Bezug seine Anfrage zur Suchmaschine Google besitzt.

Löschen von Navigationslinks. Anstelle von Kleinbergs Vorgehensweise, alle Intrinsics zu löschen, halte ich es für sinnvoll, ausschließlich Navigationslinks zu löschen, da Intrinsics, die keiner Navigation angehören, ebenfalls inhaltliche Relevanz aufweisen können. Dass die Navigationslinks aber auf jeden Fall entfernt werden sollten, hat folgenden Grund. Nahezu jedes Webdokument einer Domain mit einer Navigation zu versehen, hat sich für eine gute Benutzerführung bewährt. Dies führt zu dem Problem, dass Navigationslinks, je mehr Dokumente zu einer Domain gehören, auch häufiger auf immer dieselben Navigationsdokumente zeigen. Da-

[6] http://www.twitter.com

durch werden Navigationsdokumente automatisch besser bewertet als es der Fall wäre, wenn sie keine Navigationsdokumente darstellen würden.

Gewichtung nicht-navigationsbedingter Intrinsics. Unter den verbleibenden Kanten können sich nun auch Intrinsics befinden. Wie ich bereits angemerkt habe, stellte Kleinberg fest, dass sehr viele Intrinsics innerhalb eines Webgraphen Navigationslinks repräsentieren und generell deutlich weniger Schlüsse als Transverse darüber zulassen würden, welchen qualitativen Wert oder Themenbezug ein verlinktes Webdokument besitzt. Aus diesem Grund löscht Kleinberg innerhalb des HITS-Algorithmus alle Intrinsics, die nach der Erweiterung der Rootmenge in der Basismenge auftreten. Die Zieldokumente der Intrinsics verbleiben jedoch in der Basismenge. Erst dann finden die Berechnungen auf den Webdokumenten und noch verbleibenden Kanten statt. Diese Bewertung von Intrinsics nahm Kleinberg im Jahr 1998 vor, als Online-Communitys, Blogs, Foren, riesige Online-Enzyklopädien und andere kollaborative Inhalte innerhalb von Domains noch nicht so zahlreich wie heute waren. Immer häufiger vereinen einzelne Domains eine riesige Menge von Beiträgen verschiedener Autoren, wobei häufig innerhalb eines Beitrags auf einen anderen Beitrag eines anderen Autors verwiesen wird. Hyperlinks von einem Webdokument d zu einem anderen Webdokument d' existieren innerhalb großer Domains häufig domainintern, stellen also Intrinsics dar. Ein Beispiel für solche Domains sind Online-Enzyklopädien, die innerhalb eines Beitrags häufig Hyperlinks zu anderen anderen Beiträgen derselben Online-Enzyklopädie besitzen, die eine thematische Relevanz aufweisen. Ebenso werden Intrinsics häufig innerhalb von Nachrichten-Websites genutzt, um auf einzelne zum Thema passende Artikel zu verweisen. Es liegt nahe, dass zahlreiche Betreiber die Nutzung von Intrinsics bevorzugen, um Besucher auf ihrer Domain zu halten, als auf teils besser geeignete Dokumente außerhalb der Domain zu verweisen. Daraus kann man ableiten, dass Intrinsics weniger Aussagekraft als Transverse besitzen. Aus diesem Grund sollten sie auch mit weniger Einfluss in die Berechnung der Hubs und Authoritys eingehen als Transverse. Sie aber völlig aus den Betrachtungen auszuschließen, halte ich für einen Informationsverlust, da domaininterne Zieldokumente, die keiner Navigation angehören, auch thematisch motiviert sein und eine Aussage über die Qualität des Zieldokumentes enthalten können.

Neben der Anwendung der Navigationserkennung im HITS-Algorithmus ist sie auch in Kombination mit verschiedenen Optimierungsvorschlägen und anderen Rankingverfahren für Webdokumente denkbar. Exemplarisch soll in dieser Arbeit jedoch der HITS-Algorithmus genügen.

2.3.2 Automatische Erstellung von Sitemaps für Domains

Ein weiteres Anwendungsgebiet kann die automatische Erstellung von Sitemaps für Domains darstellen. Eine Sitemap beschreibt i. A. eine Übersicht über die hierarchische Struktur einer Domain. Können mithilfe der Navigationserkennung alle Navigationssysteme innerhalb einer Domain vollständig nachgebildet werden, kann dies die automatische Erstellung einer Sitemap unabhängig von der Ordnerstruktur auf dem Server, der die Webdokumente der Domain bereitstellt, ermöglichen bzw. unterstützen. Mit den Algorithmen und Modellen, die im Folgenden vorgestellt werden, ist solch eine Struktur noch nicht rekonstruierbar. Sie stellt aber einen ersten Schritt dar, da einzelne Navigationsysteme in vielen Fällen erkannt, aber noch nicht untereinander in

Beziehung gesetzt werden können. In einem weiteren Schritt könnte ausgehend von der Startseite einer Domain, betrachtet werden, welche Navigationslinks eines Navigationssystems verfolgt werden müssen, um zu anderen Navigationssystemen der Domain zu gelangen. Auf diese Weise könnte ein Baum von Navigationssystemen entstehen, aus dem die logische Gesamtstruktur einer Domain hergeleitet werden könnte.

3 Elemente der Navigationserkennung

In diesem Kapitel formalisiere ich Elemente der Navigationserkennung, die ich bereits in der Charakterisierung der Navigationen und Navigationssysteme angesprochen habe und stelle Algorithmen für ihre Überprüfung vor. Diese Elemente sind Gruppen, harmonierende Gruppen sowie die Verlinkungsstruktur innerhalb eines Navigationssystems. Erst im Kapitel 4 werde ich diese Elemente in einem Algorithmus zur Navigationserkennung anwenden.

3.1 Gruppen

Bei stichprobenartigen Vergleichen zwischen Webdokumenten und deren DOM-Bäumen, habe ich beobachtet, dass in vielen Fällen die <a href>-Pattern einer Navigation einer festen Struktur folgen. Diese Strukturen habe ich *Gruppen* genannt. In diesem Unterkapitel werde ich Gruppen formal beschreiben und Algorithmen vorstellen, mit denen Gruppen innerhalb eines Webdokumentes bestimmt werden können.

3.1.1 Formale Beschreibung

Im Folgenden bezeichnet $d = (w, C_w)$ ein Webdokument mit dem Quellcode C_w, der Baum $T = (V, E)$ den DOM-Baum des Quellcodes C_w und $P \subseteq V$ die Bezugsmenge, die alle <a href>-Pattern aus C enthält.

In diesem Kapitel werden folgende Notationen verwendet. Für $i > 0$ bezeichnet $parent_0(v)$ den Knoten v, $parent(v) = parent_1(v)$ den Vaterknoten von $v \in V$ und $parent_i(v) = parent(parent_{i-1}(v))$ den i-ten Vorfahren von v. Die Menge aller Knoten v des Teilbaumes, der durch den Knoten k aufgespannt wird, für die $parent_i(v) == k$ gilt, heißt i-te Kindgeneration von k. Die Tiefe $depth(v)$ beschreibt die Anzahl der Kanten auf dem Pfad vom Wurzelknoten von T zum Knoten v. Die Baumtiefe $depth(T)$ ist die maximale Tiefe eines Knotens innerhalb eines Baumes T.

Da ein <a href>-Pattern kein weiteres <a href>-Pattern enthalten darf, gilt für die Bezugsmenge P, dass kein Knoten $k_1 \in P$ i-ter Vorfahre eines anderen Knotens $k_2 \in P$ ist.

Im Folgenden sollen die in P enthaltenen <a href>-Pattern, wenn möglich, gruppiert werden. Dazu werde ich die Bezugsmenge in verschiedene Gruppen und eine Menge der ungruppierbaren Knoten partitionieren.

Definition 6 (Gruppe, Gruppierung, Menge der ungruppierbaren Knoten)
Sei $V = \{v_1, \ldots, v_{|V|}\}$ die Knotenmenge von T und $Z = \{frei, unfrei\}$ eine Menge von Zuständen, die Knoten aus V annehmen können. Dann ist ein ***Baumzustand*** *von T zu einem* ***Zeitpunkt*** *$0 \leq i \leq depth(T)$, $i \in \mathbb{N}$ ein Tupel $z_i^{T,P}(T) = (z_i^{T,P}(v_1), \ldots, z_i^{T,P}(v_{|V|})) \in Z^{|V|}$. Der* ***Startzustand*** *des Baumes T ist der Baumzustand $z_0^{T,P}(T)$ mit*

$$z_0^{T,P}(v_k) = \begin{cases} frei, & \text{wenn } v_k \in P \\ unfrei, & \text{sonst.} \end{cases} \tag{1}$$

Für $p \in V$ sei

$$F_{i,p}^{T,P} = \{v_k \in V : z_i^{T,P}(v_k) = frei \ \wedge \ parent_i(v_k) = p\} \tag{2}$$

Dann ist der Baumzustand für $i > 0$

$$z_i^{T,P}(v_k) = \begin{cases} unfrei, & \text{wenn } |F^{T,P}_{i-1,parent_{i-1}(v_k)}| \geq 2 \\ z_{i-1}^{T,P}(v_k) & sonst. \end{cases} \tag{3}$$

Der ***Endzustand*** *von* T *ist der Zustand des Baumes zum Zeitpunkt* $depth(T)$*. Jede Menge* $F_{i,p}^{T,P}$ *mit* $|F_{i,p}^{T,P}| \geq 2$ *heißt* ***Gruppe*** *von* T*, die* ***Gruppierung*** *von* T *ist die Menge* $\Omega^{T,P} = \{F_{i,p}^{T,P} : |F_{i,p}^{T,P}| \geq 2\}$*.* U *stellt die Menge aller Knoten dar, die im Endzustand frei sind und wird als* Menge der ungruppierbaren Knoten *bezeichnet.*

In einer Gruppe befinden sich dann ausschließlich Knoten aus der Bezugsmenge, die in derselben Tiefe liegen. Alle Knoten derselben Gruppe weisen außerdem die Eigenschaft auf, dass sie mit keinem Knoten derselben Tiefe außerhalb der Gruppe einen gemeinsamen Vorfahren besitzen, der weiter oder genauso weit von der Wurzel entfernt ist wie der gemeinsame Vorfahre der Knoten innerhalb der Gruppe.

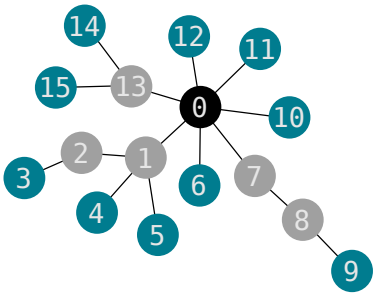


Abbildung 9: Beispiel-DOM-Baum T

Für den in Abbildung 9 dargestellten Beispiel-DOM-Baum T eines Webdokumentes d, dessen Menge aller blauen Knoten die Bezugsmenge P und dessen schwarzer Knoten mit der Nummer 0 die Wurzel von T ist, ist die Gruppierung

$$\Omega^{T,P} = \{F_{1,0}^{T,P} = \{6,10,11,12\}, F_{1,1}^{T,P} = \{4,5\}, F_{1,13}^{T,P} = \{14,15\}, F_{3,0}^{T,P} = \{3,9\}\}.$$

Die Menge der ungruppierbaren Knoten ist leer.

3.1.2 Beobachtungen

Die hier aufgeführten Beobachtungen bilden die Grundlage für den in 3.1.3 vorgestellten Algorithmus zur Berechnung der Gruppen. Im Folgenden werden Mengen $F_{i,p}^{T,P}$ auch kurz als F-Mengen bezeichnet.

1. *In einer Gruppe können sich nur Knoten der Bezugsmenge* P *befinden.*

Jede Gruppe ist auch eine F-Menge. Für diese gilt $F_{i,p}^{T,P} = \{v_k \in V : z_i^{T,P}(k) = frei \wedge parent_i(v_k) = p\}$. Es können entsprechend nur Knoten in einer F-Menge auftauchen, die frei sind. Dann ist ein

Knoten k zum Zeitpunkt $i = 0$ laut Definition 6 genau dann frei, wenn er der Bezugsmenge P angehört. Für $z_i^{T,P}(k)$ gilt:

$$z_i^{T,P}(k) = \begin{cases} unfrei, & \text{wenn } |F^{T,P}_{i-1,parent_{i-1}(k)}| \geq 2 \\ z_{i-1}^{T,P}(k) & \text{sonst.} \end{cases}$$

$z_i^{T,P}(k)$ kann also nur dann frei sein, wenn auch $z_{i-1}^{T,P}(k)$ frei war. Ein Knoten, der im Zustand i unfrei ist, kann nie wieder für einen Zustand $j > i$ frei werden. Ist k für $i = 0$ unfrei (gehört also nicht der Bezugsmenge an) kann k nicht mehr als frei markiert werden, entsprechend niemals einer F-Menge und damit auch keiner Gruppe angehören. Daraus folgt sofort, dass eine Gruppe nur Knoten der Bezugsmenge enthalten kann.

2. *Sei $P_j = \{k \in P \mid depth(k) = j\}$. Sei weiterhin $\Omega^{T,P}$ die Gruppierung unter der Bezugsmenge P und Ω^{T,P_j} die Gruppierung für den DOM-Baum T unter der Bezugsmenge P_j. Dann gilt*

$$\Omega^{T,P} = \bigcup_{j=0}^{depth(T)} \Omega^{T,P_j}$$

Die separate Berechnung der Gruppierung für jede Tiefe und der anschließenden Vereinigung ist möglich, da ...

2.1. *... jede F-Menge und damit auch jede Gruppe ausschließlich Knoten derselben Tiefe enthält.*

Sei $k \in P$ ein Knoten der Tiefe t und $k' \in P$ ein Knoten der Tiefe $t' \neq t$. Seien außerdem $p = parent_i(k)$ und $p' = parent_i(k')$ für ein $0 \leq i \leq min(t,t')$. Dann muss p die Tiefe $t - i$ und p' die Tiefe $t' - i$ besitzen. Da $t \neq t'$, müssen p und p' in unterschiedlicher Tiefe in T liegen. p und p' können entsprechend nicht denselben Knoten darstellen. Aus diesem Grund sind $F_{i,p}^{T,P}$ und $F_{i,p'}^{T,P}$ disjunkt. k und k' können also nicht derselben F-Menge angehören.

2.2. *... ein Knoten $k \in P$ der Tiefe t keine F-Menge beeinflussen kann, in der ein beliebiger Knoten $k' \in P$ der Tiefe $t' \neq t$ auftaucht. Damit kann k auch nicht beeinflussen, ob und in welcher Gruppe sich ein Knoten einer Tiefe $t' \neq t$ befindet.*

Jede Gruppe ist auch eine Menge $F_{i,p}^{T,P}$ mit $i > 0$, die die Menge aller freien Knoten k darstellt, für die $parent_i(k) == p$ gilt. Ob ein Knoten frei ist, ist dabei abhängig von $F^{T,P}_{i-1,parent_{i-1}(k)}$ und k selbst. Es ist leicht zu sehen, dass $F^{T,P}_{i-1,parent_{i-1}(k)}$ in keinem Fall den Knoten k' enthalten kann, da $F^{T,P}_{i-1,parent_{i-1}(k)}$ den Knoten k und $F^{T,P}_{i-1,parent_{i-1}(k')}$ den Knoten k' enthalten muss, die Knoten k und k' aber nicht derselben F-Menge angehören können. Entsprechend kann kein Zustand $z_i(k)$ vom Knoten k' beeinflusst werden. Damit ist auch generell kein Einfluss eines Knotens der Tiefe t' auf die Bildung der F-Mengen und Gruppen der Tiefe t möglich.

3. *Die Gruppierung eines Baumes T unter einer Bezugsmenge P_j mit weniger als zwei Knoten ist leer.*

Diese Beobachtung ist sofort nachvollziehbar. Existiert nur ein Knoten oder gar keiner, kann die F-Menge nicht mindestens zwei Knoten enthalten und folglich auch keine Gruppe sein. Aus diesem Grund kann auch die Menge P_0 immer für die Berechnung der Gruppen vernachlässigt werden, da diese maximal einen Knoten - den Wurzelknoten - enthält.

3.1.3 Algorithmische Bestimmung von Gruppen

Zur algorithmischen Bestimmung der Gruppierung eines Webdokumentes d mit dem DOM-Baum $T = (V, E)$ und der Bezugsmenge P von d dient der Algorithmus *gruppiereKnoten*(T, P).

Partitionierung

Im ersten Schritt des Algorithmus wird die Bezugsmenge in verschiedene Mengen P_j partitioniert, wobei jede Menge P_j alle Knoten der Tiefe j enthält. Die Berechnung der Gruppen wird dann für jede Menge P_j, die mindestens 2 Knoten enthält, separat durchgeführt. Diese Vorgehensweise folgt aus den Beobachtungen 1 bis 3 und verfälscht, wie ich gezeigt habe, nicht das Endergebnis. Sie dient dazu, Knoten, die offensichtlich nicht gruppiert werden können, weil sie die Einzigen in ihrer Tiefe sind, von vornherein nicht zu betrachten.

Bestimmung von Ω

Wurde eine Menge P_j ausgewählt, wird zunächst für jeden Knoten $k \in P_j$ ein aktueller Vorfahre p_k bestimmt - dies ist anfangs der Knoten k selbst. Anschließend beginnt eine Iteration über i, beginnend mit $i = 1$. In jedem Iterationsschritt ist der aktuelle Vorfahre p_k eines Knotens k genau der i-te Vorfahre von k. Zu jedem Knoten p_k wird eine Menge $F_{i,p_k} = F_{i,parent_i(k)}$ gebildet, in die alle Knoten k eingefügt werden, die p_k als i-ten (und damit aktuellen) Vorfahren besitzen. Jede Menge F_{i,p_k}, die mindestens zwei Knoten enthält, wird als Gruppe in Ω eingefügt, während alle in Ω als Gruppe eingefügte Knoten aus P_j entfernt werden, um nicht erneut in eine Gruppe aufgenommen werden zu können. Dieser Schritt ersetzt die Änderung von $z_i(k)$ in den Zustand „unfrei“. Ich habe bereits angemerkt, dass ein Knoten, der einmal als unfrei markiert wurde, nicht mehr den Zustand frei annehmen kann. Aus diesem Grund muss der entsprechende Knoten auch nicht mehr betrachtet werden.

Abbruchbedingung für Schleifendurchläufe

Besitzt P_j nach dem Entfernen aller Knoten, die einer Gruppe angehören, keinen oder nur noch einen Knoten wird die Iteration abgebrochen, da eine Gruppe immer mindestens zwei Knoten enthalten muss. Enthält P_j noch mindestens zwei Knoten, so wird i erhöht und die Bestimmung von Gruppen mit einem neuen Abstand der übrigen Knoten zu ihren aktuellen Vorfahren versucht. Entspricht i der Tiefe j, ist der i-te Vorfahre der Wurzelknoten. Hier werden alle verbleibenden Knoten (wenn es mindestens 2 sind) gruppiert, da sie dieselbe Tiefe besitzen und somit alle den Wurzelknoten als aktuellen Vorfahren besitzen müssen. Andernfalls wären sie nicht gemeinsam in der P_j-Menge. Aus diesem Grund wird die WHILE-Schleife für jede Menge P_j auch maximal j-mal betreten. Sind weniger als zwei Knoten in P_j vorhanden, so wird die Iteration abgebrochen und eine neue Menge von P_k mit $k \neq j$ gewählt, die noch nicht bearbeitet wurde.

Die in Ω gesammelten Gruppen werden schließlich als Gruppierung von T unter der Bezugs-

menge P ausgegeben.

Algorithm 1 *gruppiereKnoten*(T,P)

Input: Baum $T=(V,E)$, Bezugsmenge $P \subseteq V$ von T
Output: Gruppierung Ω von T

1: $\Omega \leftarrow \emptyset$
2: $\forall 1 \leq j \leq depth(T)\colon P_j \leftarrow \{k \in P \mid depth(k) = j\}$
3: **for each** P_j where $|P_j| \geq 2$ **do**
4: $\quad \forall k\colon p_k \leftarrow k$
5: $\quad i \leftarrow 1$
6: $\quad$ **while** $(|P_j| \geq 2)$ **do**
7: $\quad\quad \mathbb{F} = \emptyset$ *//Die Menge $\mathbb{F}$ von Mengen dient nur als temporäre „Sammelstelle“ für die berechneten $F_{i,p}$, um gezielt in der Schleife in Zeile 12 darauf zugreifen zu können.*
8: $\quad\quad \forall k \in P_j\colon p_k \leftarrow parent(p_k)$.
9: $\quad\quad \forall k \in P_j\colon F_{i,p_k} \leftarrow \emptyset$
10: $\quad\quad \forall k \in P_j\colon F_{i,p_k} \leftarrow F_{i,p_k} \cup \{k\}$
11: $\quad\quad \forall k \in P_j\colon \mathbb{F} \leftarrow \mathbb{F} \cup \{F_{i,p_k}\}$
12: $\quad\quad$ **for each** $F_{i,p} \in \mathbb{F}$ where $|F_{i,p}| \geq 2$ **do**
13: $\quad\quad\quad \Omega \leftarrow \Omega \cup \{F_{i,p}\}$
14: $\quad\quad\quad P_j \leftarrow P_j \setminus F_{i,p}$
15: $\quad\quad$ **end for**
16: $\quad\quad i \leftarrow i+1$
17: $\quad$ **end while**
18: **end for**
19: **return** Ω

Laufzeit

Ich nehme folgende Operationen als elementare Operationen mit einem Berechnungsaufwand in $O(1)$ an.

- einer Variable eine leeren Menge, eine Zahl oder einen Knoten zuweisen
- Verschieben eines Knotens von einer Menge in eine andere Menge
- Löschen eines Knotens einer Menge
- Erhöhen eines Zahlenwertes
- Ausgeben eines Knotens

Unter diesen Annahmen benötigt der Algorithmus *gruppiereKnoten*(T,P) einen Berechnungsaufwand in $O(depth(T) \cdot |P|)$.

Zeile 1-2

Dass die Bestimmung der P_j-Mengen in $O(depth(T) \cdot |P|)$ liegt, ist leicht erkennbar. Hierfür ist die Berechnung aller Tiefen der Knoten aus P notwendig und kann bestimmt werden, indem

für jeden Knoten aus P solange der Vaterknoten, dessen Vaterknoten usw. bestimmt wird, bis man auf die Wurzel des Baumes stößt. Dies beansprucht für einen Knoten $k \in P$ eine Laufzeit in $O(depth(k)) \subseteq O(depth(T))$, entsprechend ist die Berechnung aller Tiefen in $O(depth(T) \cdot |P|)$ möglich. Diese Berechnung ist alternativ auch in $O(|V|)$ möglich, indem der Baum T auf dieselbe Art wie bei der Tiefensuche traversiert und die entsprechende Tiefe jeweils an die Kindknoten weitergegeben wird. Dies beansprucht einen Zeitaufwand $O(|E|)$, da jede Kante genau zweimal besucht werden muss. Da jeder Baum aber per Definition genau $|V|+1$ Kanten besitzt, kann $O(|E|)$ durch $O(|V|)$ ersetzt werden.

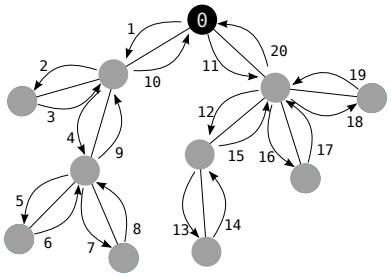


Abbildung 10: Traversierung eines Baumes nach dem Verfahren der Tiefensuche

Zeile 3
Die Überprüfung, ob P_j mindestens 2 Knoten enthält, liegt in $O(1)$, da nur die ersten zwei Knoten der Menge betrachtet werden müssen. Dieser Vergleich findet für jedes P_j maximal einmal statt. Entsprechend liegt der Aufwand über alle Schleifendurchläufe in $O(depth(T))$.

Zeile 4-5
Über alle Schleifendurchläufe betrachtet, benötigt Zeile 4 Zuweisungsschritte in $O(|P|)$, da alle P_j voneinander disjunkt sind und entsprechend in Zeile 4 für jeden Knoten $k \in P$ maximal einmal $p_k = k$ gesetzt wird. Zeile 5, deren Aufwand jeweils in $O(1)$ liegt, wird maximal $depth(T)$-mal aufgerufen. Über alle Schleifendurchläufe betrachtet, liegt der Berechnungsaufwand für Zeile 5 also in $O(depth(T))$.

Zeile 6
Die Überprüfung, ob eine Menge P_j mindestens zwei Knoten enthält, nimmt wie in Zeile 3 einen Berechnungsaufwand in $O(1)$ in Anspruch. Im schlechtesten Fall sind aber j Schleifendurchläufe (siehe nächster Punkt) für jedes P_j notwendig. Da maximal $|P|$ nicht-leere Mengen P_j existieren können, liegt der Berechnungsaufwand in $O(depth(T) \cdot |P|)$.

Schleifendurchläufe für die WHILE-Schleife ab Zeile 6
Die WHILE-Schleife ab Zeile 6 wird für jede betrachtete Menge P_j maximal j-mal durchlaufen, da für $i = j$ alle in P_j verbliebenen Knoten denselben Vorfahren - den Wurzelknoten *root* - teilen, da sie sonst nicht dieselbe Tiefe hätten. Handelt es sich um einen einzigen verbliebenen Knoten, wird die Schleife gar nicht erst betreten. Sind noch mindestens zwei Knoten vorhanden, so werden sie auf jeden Fall alle in die Menge $F_{j,root}$ aufgenommen, in Zeile 13 als Gruppe in Ω aufgenommen und anschließend aus P_j gelöscht, sodass spätestens danach die Schleife nicht

erneut durchlaufen wird.

Zeile 7-11, 16
Für die Zeilen 8 bis 11 liegt der Aufwand für einen einzigen Schleifendurchlauf der WHILE-Schleife in $O(|P_j|)$ und damit für alle Schleifendurchläufe für ein festes P_j in $O(j \cdot |P_j|)$. Über alle Schleifendurchläufe liegt der Aufwand entsprechend in $O(\sum_{j=1}^{depth(T)} j \cdot |P_j|)$. Dabei gilt:

$$\sum_{j=1}^{depth(T)} j \cdot |P_j| \quad < \quad depth(T) \cdot \sum_{j=1}^{depth(T)} |P_j| \quad = \quad depth(T) \cdot |P|$$

Damit liegt der Aufwand für die Berechnungen in den Zeilen 8 bis 11 über alle Schleifendurchläufe in $O(depth(T) \cdot |P|)$. In jedem Durchlauf, in dem alle Knoten einer Menge P_j in Zeile 8-11 betrachtet werden, werden zusätzlich Zeile 7 und 16 aufgerufen, die in einem Schleifendurchlauf weniger Berechnungsaufwand mit sich bringen als die Zeilen 8-11. Entsprechend liegt auch der Berechnungsaufwand für diese Zeilen über alle Schleifendurchläufe in $O(depth(T) \cdot |P|)$.

Zeile 12
In Zeile 12 muss jede Menge $F_{i,p}$, die im Verlauf des Algorithmus gebildet wird, einmal auf ihre Größe überprüft werden. Da dafür nur überprüft werden muss, ob mindestens zwei Knoten enthalten sind, benötigt die Überprüfung einer Menge $F_{i,p}$ einen Aufwand in $O(1)$. Insgesamt müssen aber im schlechtesten Fall $depth(T) \cdot |P|$ F-Mengen überprüft werden. Dies ist dann der Fall, wenn die einzige nicht-leere Menge der Partition von P die Menge $P_{depth(T)} = P$ ist und der einzige gemeinsame i-te Vorfahre aller Knoten aus P der Wurzelknoten ist. Über alle Schleifendurchläufe benötigt Zeile 12 also ebenfalls einen Berechnungsaufwand in $O(depth(T) \cdot |P|)$.

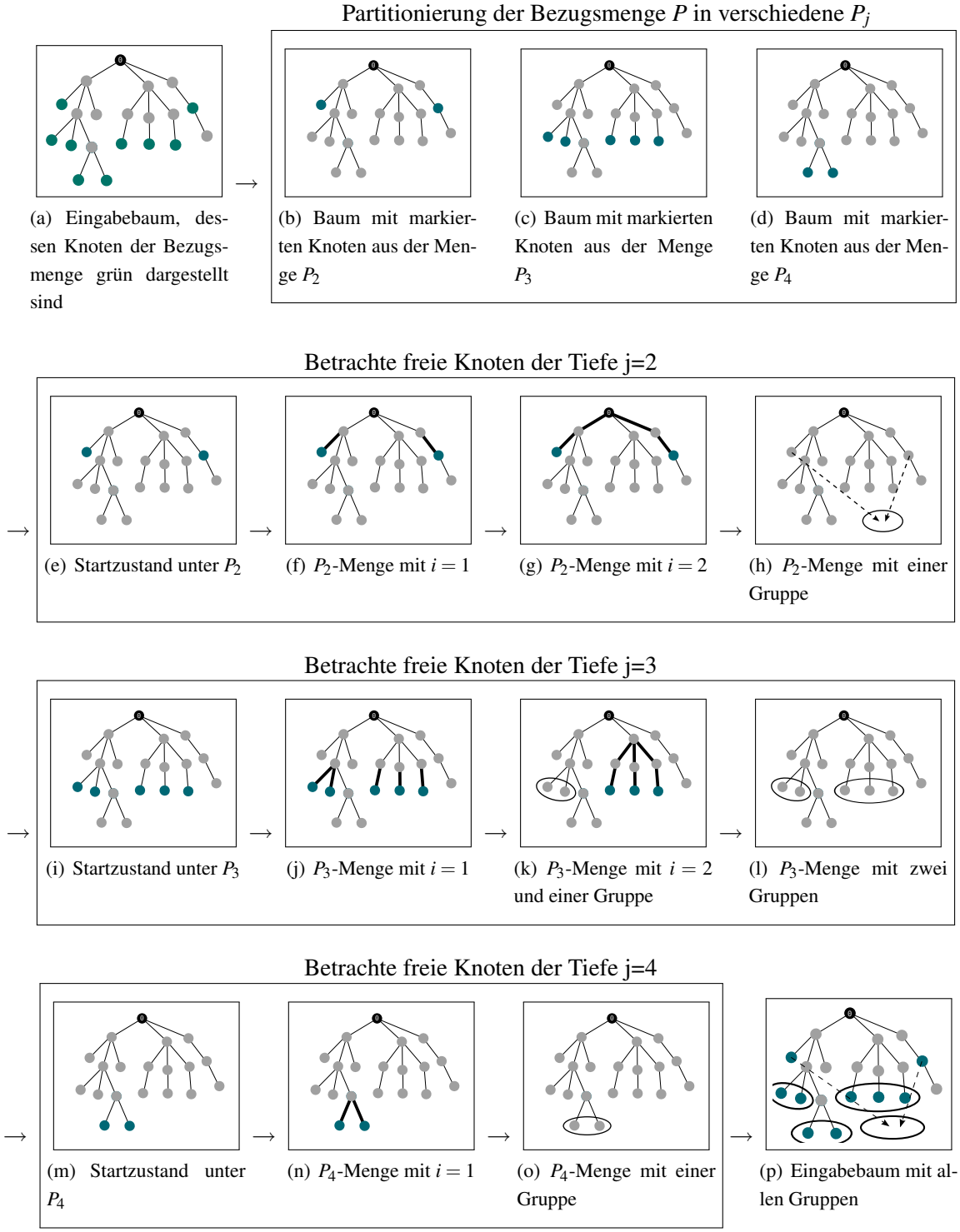
Zeile 13-14
Da jeder Knoten $k \in P$ maximal einmal in eine Gruppe aufgenommen werden kann, da er anschließend sofort aus der Menge P_j, der k angehört, entfernt wird, benötigen die Zeilen 13 und 14 über alle Schleifendurchläufe einen Berechnungsaufwand in $O(|P|)$.

Zeile 19
Die Rückgabe der Gruppen liegt in $O(|P|)$, da jeder Knoten aus P maximal einer Gruppe angehören kann.

Gesamt
Insgesamt liegt der Berechnungsaufwand für den Algorithmus *gruppiereKnoten*(T,P) also in $O(depth(T) \cdot |P|)$.

Ein Beispiel für eine Durchführung des Algorithmus *gruppiereKnoten*(T,P) ist in Abbildung 11 sichtbar.

Darstellung des Algorithmus *gruppiereKnoten* an einem Beispiel



(a) Eingabebaum, dessen Knoten der Bezugsmenge grün dargestellt sind

(b) Baum mit markierten Knoten aus der Menge P_2

(c) Baum mit markierten Knoten aus der Menge P_3

(d) Baum mit markierten Knoten aus der Menge P_4

(e) Startzustand unter P_2

(f) P_2-Menge mit $i = 1$

(g) P_2-Menge mit $i = 2$

(h) P_2-Menge mit einer Gruppe

(i) Startzustand unter P_3

(j) P_3-Menge mit $i = 1$

(k) P_3-Menge mit $i = 2$ und einer Gruppe

(l) P_3-Menge mit zwei Gruppen

(m) Startzustand unter P_4

(n) P_4-Menge mit $i = 1$

(o) P_4-Menge mit einer Gruppe

(p) Eingabebaum mit allen Gruppen

Abbildung 11: Beispiel für eine Durchführung des *gruppiereKnoten*-Algorithmus

3.1.4 Erweiterung

Manchmal kommt es vor, dass zwei <a href>-Pattern innerhalb eines Webdokumentes auf dasselbe Webdokument verweisen. Solche <a href>-Pattern nenne ich *verweisäquivalent*. Verweisäquivalente <a href>-Pattern können in der Browserinterpretation in einigen Fällen aus Sicht eines Nutzers als einzelner Hyperlink wahrgenommen werden. Ein Beispiel für solche Verweise stellt ein Ausschnitt aus der Website *http://uniweb.uni-luebeck.de* dar, bei dem der blau eingefärbte Text sowie das darüberstehende Bild jeweils auf dasselbe Zieldokument verweisen. Obwohl es sich technisch um zwei Verweise handelt, kann man sie auch als einen einzigen Verweis wahrnehmen.

Abbildung 12: Ausschnitt der vorläufigen neuen Startseite [24] der Universität zu Lübeck

Solche Verweise, die wie in Abbildung 12 optisch auch als ein einziger Verweis wahrgenommen werden können, können vom Algorithmus *gruppiereKnoten* als eigenständige Gruppe erkannt werden, wenn die <a href>-Pattern dieselbe Tiefe im DOM-Baum besitzen. In diesem Fall würde der Algorithmus vier verschiedene Gruppen für Abbildung 12 berechnen. Eigentlich wollen wir in diesem Beispiel aber nur eine einzige Gruppe erkennen. Aus diesem Grund sollten wir anstelle der Anzahl der <a href>-Pattern innerhalb einer Menge die Anzahl der unterschiedlichen Zieldokumente zählen, um zu verhindern, dass zwei Verweise auf dasselbe Webdokument eine eigenständige Gruppe bilden können. Für eine Menge F von <a href>-Pattern eines Webdokumentes beschreibt dafür im Folgenden $destinationCnt(F)$ eine Funktion, die die Anzahl der unterschiedlichen Zieladressen liefert.

Etwas anders verhält es sich mit zwei Verweisen auf dasselbe Zieldokument im Beispiel aus Abbildung 13, einem Ausschnitt der Informatik-Website der Universität zu Lübeck. Hier befindet sich innerhalb eines Textes ein blau eingefärbter Verweis mit der Bezeichnung *Anwendungen*, der zum selben Webdokument zeigt, auf das der Hyperlink mit der Bezeichnung *Anwendungsfächer als Brücke* rechts in der blauen Leiste verweist. Da diese Verweise jedoch in verschiedenen Gruppen auftreten, sind sie für uns unproblematisch.

Eine Änderung, die wir im Algorithmus *gruppiereKnoten* vornehmen müssen, um zu verhindern das Gruppen ausschließlich Verweise enthalten, die alle auf dasselbe Dokument zeigen, befindet sich in Zeile 12 (siehe *erweiterterGruppiereKnoten-Algorithmus*). Dort wurde $|F_{i,p}| \geq 2$ bereits durch $destinationCnt(F_{i,p}) \geq 2$ ersetzt. Allerdings kann diese Änderung bewirken, dass der Al-

Abbildung 13: Ausschnitt eines Webdokumentes [18] der alten Informatik-Website der Universität zu Lübeck

Algorithm 2 *erweiterterGruppiereKnoten-Algorithmus*

1: $\Omega \leftarrow \emptyset$
2: $\forall 1 \leq j \leq depth(T)\colon P_j \leftarrow \{k \in P \mid depth(k) = j\}$
3: **for each** P_j where $|P_j| \geq 2$ **do**
4: $\quad \forall k\colon p_k \leftarrow k$
5: $\quad i \leftarrow 1$
6: $\quad$ **while** $\boxed{(|P_j| \geq 2)\ \textbf{and}\ (i \leq j)}$ **do**
7: $\quad\quad \mathbb{F} = \emptyset$
8: $\quad\quad \forall k \in P_j\colon p_k \leftarrow parent(p_k).$
9: $\quad\quad \forall k \in P_j\colon F_{i,p_k} \leftarrow \emptyset$
10: $\quad\quad \forall k \in P_j\colon F_{i,p_k} \leftarrow F_{i,p_k} \cup \{k\}$
11: $\quad\quad \forall k \in P_j\colon \mathbb{F} \quad \leftarrow \mathbb{F} \cup \{F_{i,p_k}\}$
12: $\quad\quad$ **for each** $F_{i,p} \in \mathbb{F}$ where $\boxed{destinationCnt(F_{i,p}) \geq 2}$ **do**
13: $\quad\quad\quad P_j \leftarrow P_j \setminus F_{i,p}$
14: $\quad\quad\quad \Omega \leftarrow \Omega \cup \{F_{i,p}\}$
15: $\quad\quad$ **end for**
16: $\quad\quad i \leftarrow i + 1$
17: $\quad$ **end while**
18: **end for**
19: **return** Ω

gorithmus nicht mehr terminiert. Dafür müssen wir uns die Bedingung für das Betreten der WHILE-Schleife in Zeile 6 ansehen. Vorher war klar, dass, wenn bei Erreichen der Iterationsstufe $i = j$ noch mindestens 2 Knoten in P_j vorhanden sind, die Knoten gruppiert werden, P_j anschließend leer ist und die WHILE-Schleife für dieses P_j nicht erneut betreten wird. Dies ist jetzt nicht mehr gegeben, da mehrere Knoten in P_j vorhanden sein können, die alle auf dasselbe Webdokument zeigen. Das Problem können wir beheben, indem die WHILE-Schleife ab Zeile 6 nur unter der Bedingung $(|P_j| \geq 2)$ und $(i \leq j)$ betreten wird. Die resultierenden Änderungen sind im Algorithmus *erweiterter gruppiereKnoten-Algorithmus* eingerahmt.

Diese Änderungen wirken sich negativ auf die Laufzeit aus. Während die Überprüfung einer Menge $|F_{i,p}| \geq 2$ einen Aufwand von $O(1)$ beansprucht, benötigt das Zählen der unterschiedli-

chen Zieladressen in $F_{i,p}$ einen linearen Aufwand in $|F_{i,p}|$. Unter diesen Änderungen liegt der Berechnungsaufwand in $O(|P|^2 \cdot depth(T))$, da im schlechtesten Fall $|P| \cdot depth(T)$ verschiedene $F_{i,p}$ überprüft werden müssen und jede Menge $F_{i,p}$ wie im Beispiel in Abbildung 14 $|P|$ Elemente beinhalten kann.

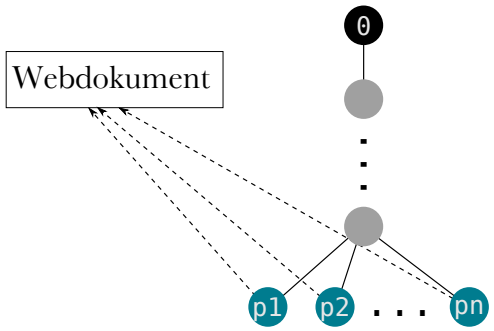


Abbildung 14: Worstcase-Szenario für den *erweiterten GruppiereKnoten-Algorithmus*

3.1.5 Beispiel

Ein Beispiel für einen XHTML-DOM-Baum ist in Abbildung 15 zu sehen[7]. Das <html>-Element ist schwarz, die <a href>-Pattern, also alle Knoten der Basismenge, sind blau und alle weiteren Elemente grau dargestellt.

Die Gruppierung dieses Baumes (ohne Anwendung der Erweiterung) ist Ω_{ifis}.

$$\begin{aligned}\Omega_{ifis} = \{ \quad & \Gamma^{ifis}_{1,12} = \{13,15\},\\ & \Gamma^{ifis}_{1,17} = \{18,20,22,24,27\},\\ & \Gamma^{ifis}_{1,76} = \{77,78\},\\ & \Gamma^{ifis}_{2,45} = \{47,49,51,53,55,57,59,61,63,65\}\\ & \Gamma^{ifis}_{4,41} = \{68,87\}\ \}\end{aligned}$$

Im Folgenden sind die für Abbildung 15 die resultierenden Gruppen mit Anwendung der Erweiterung durch die Knoten-Nummer im DOM-Baum, die URL der Zieladresse sowie der Bezeichnung der Hyperlinks, wie sie in der Interpretation des Webdokumentes sichtbar ist, aufgeführt. Die <a href>-Pattern der Knoten 13 und 15 sind verweisäquivalent und die einzigen Knoten aus P in ihrer Tiefe. Außerdem sind sie in der Interpretation des Webdokumentes nicht sichtbar. Aus diesem Grund gehören sie im erweiterten *gruppiereKnoten*-Algorithmus keiner Gruppe an.

[7] Für die Darstellung des Baumes habe ich den Quellcode des Webdokumentes *http://www.aharef.info/static/htmlgraph/sourcecode.html* benutzt und abgewandelt. Die Originaldarstellung des Baumes nach dem vorgegebenen Quellcode kann unter *http://www.aharef.info/static/htmlgraph/?url=http://www.ifis.uni-luebeck.de* angesehen werden.

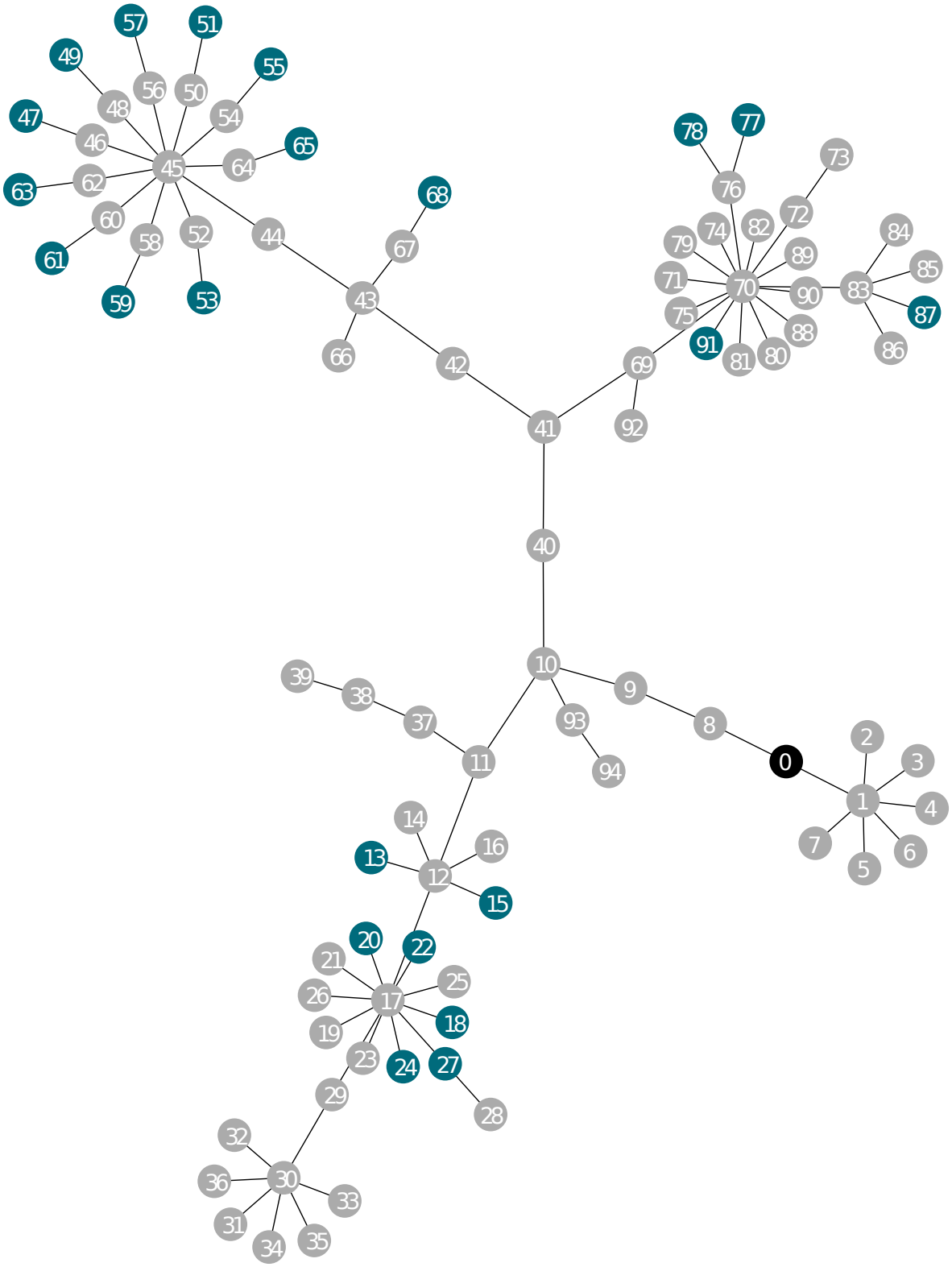


Abbildung 15: XHTML-DOM-Baum der Startseite [20] des Instituts für Informationssysteme

Gruppe $\Gamma_{1,17}$

18	http://www.ifis.uni-luebeck.de/index.php?id=4	IFIS Home
20	http://www.ifis.uni-luebeck.de/index.php?id=24	Universität
22	http://www.ifis.uni-luebeck.de/index.php?id=25	UnivIS
24	http://www.ifis.uni-luebeck.de/index.php?id=22	Impressum
27	http://www.ifis.uni-luebeck.de/index.php?id=ifis&L=1	[Bild]

Gruppe $\Gamma_{2,45}$

47	http://www.ifis.uni-luebeck.de/index.php?id=ifis	Startseite
49	http://www.ifis.uni-luebeck.de/index.php?id=mitarbeiter	Mitarbeiter
51	http://www.ifis.uni-luebeck.de/index.php?id=projekte	Projekte
53	http://www.ifis.uni-luebeck.de/index.php?id=publikationen	Veröffentlichungen
55	http://www.ifis.uni-luebeck.de/index.php?id=lehre	Lehre
57	http://www.ifis.uni-luebeck.de/index.php?id=30	Mitteilungen des Vorsitzenden des Diplom-Prüfungsausschusses
59	http://www.ifis.uni-luebeck.de/index.php?id=studentische-arbeiten	Studentische Arbeiten
61	http://www.ifis.uni-luebeck.de/index.php?id=stellenangebote	Stellenangebote
63	http://www.ifis.uni-luebeck.de/index.php?id=anfahrtsweg	Anfahrtsweg
65	http://www.ifis.uni-luebeck.de/index.php?id=links	Links

Gruppe $\Gamma_{1,76}$

77	http://www.informatik.uni-luebeck.de/index.php?id=40	Institute für Informatik
78	http://www.uni-luebeck.de/	Universität zu Lübeck

Gruppe $\Gamma_{4,41}$

68	http://www.ifis.uni-luebeck.de/index.php?id=stellenangebote	Neue Hiwi-Stellen
87	http://www.luebeck.de	Lübeck

In Abbildung 16 ist ein Screenshot des Webdokumentes *http://www.ifis.uni-luebeck.de* abgebildet. Die Hyperlinks der berechneten Gruppierungen sind mit blauen, abgerundeten Kästen zusammengefasst.

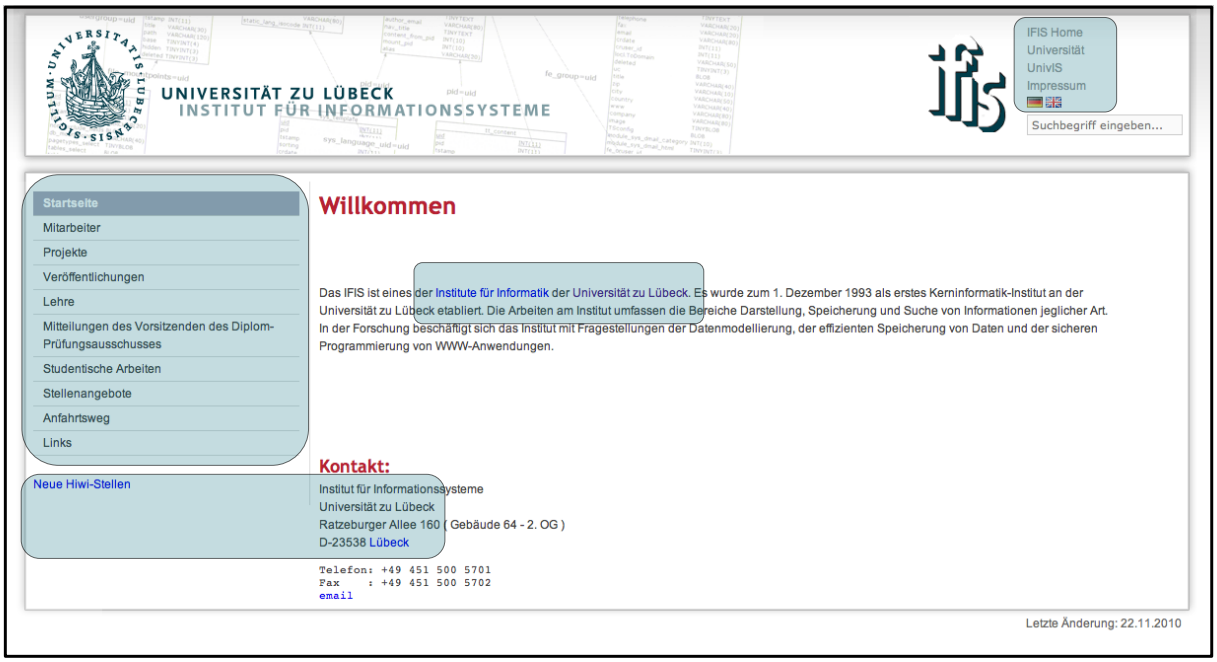


Abbildung 16: Screenshot der Startseite [20] des Instituts für Informationssysteme der Univerität zu Lübeck mit eingerahmten Gruppen.

3.2 Harmonierende Gruppen

In der Charakterisierung der Navigationssysteme und Navigationen habe ich darauf hingewiesen, dass die Ordnung der <a href>-Pattern von Navigationen innerhalb eines Navigationssystems häufig bestimmte Eigenschaften erfüllt. Am Ende dieses Abschnitts soll eine Formalisierung dieser Eigenschaften gegeben sein – eine Formalisierung eines geeigneten Vergleichs von zwei Gruppen.

3.2.1 Ähnlichkeiten von <a href>-Pattern

Als einen Unterpunkt der Systemeigenschaften betrachte ich die Darstellung von <a href>-Pattern. Dabei ist zunächst das Ziel, ähnliche <a href>-Pattern zu identifizieren, um dann die Gruppen, in denen sie enthalten sind, auf weitere Ähnlichkeiten zu untersuchen. Ich werde drei Äquivalenzen vorstellen, die Aufschluss über die Ähnlichkeit zwischen beliebigen <a href>-Pattern geben sollen. Dabei habe ich mich für mehrere Äquivalenzen entschieden, da Navigationssysteme auf sehr vielfältige Arten und Weisen umgesetzt werden können. Einige Na-

vigationen besitzen, obwohl sie demselben Navigationssystem angehören, verschiedene Farben, z.B. um verschiedene Themenbereiche voneinander abzuheben. Sind die <a href>-Pattern in einem solchen Fall durch Bilder umgesetzt, zeigen sie zwar auf dasselbe Dokument, besitzen aber nicht denselben Bezeichner. In anderen Fällen existieren, wie unter *http://www.informatik.uni-luebeck.de* innerhalb eines Webdokumentes mehrere <a href>-Pattern, die auf dasselbe Webdokument verweisen. Dann fällt es schwer, nur anhand des Zieldokumentes zu entscheiden, welche <a href>-Pattern dokumentübergreifend für einen Nutzer identisch erscheinen. In diesem Webdokument unterscheiden sich die <a href>-Pattern aber beispielsweise in ihren Bezeichnern. Das macht eine Zuordnung einfacher. Die Wahl der Äquivalenz kann so den Gegebenheiten der betrachteten Domains angepasst werden, um so gut wie möglich <a href>-Pattern unterschiedlicher Webdokumente zu erkennen, die für einen Nutzer identisch erscheinen.

Innerhalb zwei verschiedener Navigationen desselben Navigationssystems können <a href>-Pattern von einem Nutzer insbesondere dann als identisch wahrgenommen werden, wenn sie sehr ähnlich dargestellt sind und zu demselben Zieldokument verweisen. Um diese Ähnlichkeit messbar zu machen, habe ich bereits die Verweisäquivalenz eingeführt. An dieser Stelle wird sie der Vollständigkeit halber noch einmal aufgeführt.

Definition 7 (Verweisäquivalenz)
Zwei <a href>-Pattern p_1 und p_2 heißen genau dann zueinander verweisäquivalent (oder v-äquivalent), wenn sie auf dasselbe Webdokument verweisen.

Zusätzlich führe ich zwei weitere Äquivalenzen, die Bezeichner- und die Darstellungsäquivalenz ein.

Definition 8 (Bezeichneräquivalenz)
Zwei verweisäquivalente <a href>-Pattern p_1 mit dem Bezeichner δ_1 und p_2 mit dem Bezeichner δ_2 heißen genau dann bezeichneräquivalent (oder b-äquivalent) zueinander, wenn $\delta_1 = \delta_2$.

Zwei <a href>-Pattern können trotz desselben Bezeichners sehr unterschiedlich erscheinen, wenn sie sich in Farbe, Größe, Position u.ä. unterscheiden. Aus diesem Grund soll die Darstellungsäquivalenz zwei <a href>-Pattern beschreiben, die bezeichneräquivalent sind, und dessen Darstellung weitgehend übereinstimmt. Allerdings ist die Darstellung eines <a href>-Pattern abhängig von verschiedenen Faktoren. Möglicherweise wird ein und dasselbe <a href>-Pattern von zwei verschiedenen Browsern auf zwei verschiedene Weisen dargestellt. Dennoch enthält der Quellcode häufig eine Reihe von Informationen darüber, welche Eigenschaften verschiedene im Browser darzustellende HTML-Elemente aufweisen sollen. Jedes HTML-Element kann solche Informationen über die Elemente, die sich in seinem Einschluss befinden, im Elementnamen oder in seinen Attributwerten enthalten. So kann beispielsweise mit einem <H1>-Element erreicht werden, dass der Text im Einschluss größer dargestellt wird, während ein <TABLE>-Element die Positionierung beeinflussen kann.

Da die Darstellung eines Bezeichners eines <a href>-Patterns vor allem durch die Eigenschaften der (X)HTML-Elemente bestimmt wird, die das Pattern enthalten, sowie den Eigenschaften des <a href>-Patterns selbst, werden diese in einer *Pfadbeschreibung* zusammengefasst.

Der Tagname sowie alle Attributzuweisungen befinden sich im Anfangstag eines Elementes. Das Endtag, falls es vorhanden ist, enthält keine weiteren Informationen und dient nur dazu, das Element zu begrenzen. Aus diesem Grund enthält die Pfadbeschreibung mit Ausnahme des <a href>-Patterns ausschließlich die Anfangstags der betrachteten Knoten. Das jeweilige Anfangstag eines Elementes, das im DOM-Baum durch den Knoten k repräsentiert wird, wird mit der Funktion $anfangsTag(k)$ bestimmt.

Definition 9 (Pfadbeschreibung)
Sei $T = (V,E)$ ein DOM-Baum eines Webdokumentes und der Knoten $p \in V$ ein <a href>-Pattern mit $depth(p) = t$. Dann ist die **Pfadbeschreibung** *von p der String $path = anfangsTag(parent_t(p)) \circ anfangsTag(parent_{t-1}(p)) \circ \ldots \circ anfangsTag(parent_1(p)) \circ p$.*

Mithilfe der Pfadbeschreibung kann dann die Darstellung von Bezeichnern verschiedener <a href>-Pattern noch genauer betrachtet werden.

Definition 10 (Darstellungsäquivalenz)
Zwei bezeichneräquivalente <a href>-Pattern p_1 mit der Pfadbeschreibung $path_1$ und p_2 mit der Pfadbeschreibung $path_2$ heißen genau dann darstellungsäquivalent (oder d-äquivalent) zueinander, wenn $path_1 = path_2$.

Ein Beispiel für darstellungsäquivalente <a href>-Pattern ist in Abbildung 17 dargestellt. Die Verweis-, Bezeichner- und Darstellungsäquivalenzen kann man auch als Ähnlichkeitsgrade zwischen Hyperlinkdarstellungen betrachten. Verweisäquivalenzen weisen einen eher geringen Ähnlichkeitsgrad, Bezeichneräquivalenzen einen etwas höheren und Darstellungsäquivalenzen den höchsten betrachteten Ähnlichkeitsgrad auf. Je ähnlicher sich zwei Hyperlinkdarstellungen verschiedener Webdokumente sind, desto wahrscheinlicher ist es, dass sie Elemente desselben Navigationssystems in verschiedenen Navigationen darstellen. In Abbildung 17 sind zwei Webdokumente einer Domain abgebildet. Unter den dargestellten Webdokumenten ist jeweils ein Ausschnitt des HTML-Quellcodes des Webdokumentes zu sehen[8]. In der Navigation links unter der Überschrift *Das Institut* befindet sich ein Hyperlink zur Startseite. Innerhalb jedes Navigationsdokumentes dieser Domain ist die Pfadbeschreibung der <a href>-Pattern, die jeweils den Hyperlink zur Startseite erzeugen, identisch.

Basierend auf den eingeführten Äquivalenzen führe ich zusätzlich Klassen-IDs ein. Diese können genutzt werden, um eine Kodierung der Äquivalenzklassen vorzunehmen, sodass anstelle langer Webadressen deutlich kürzere IDs miteinander verglichen werden. Im Folgenden gibt $x \in \{v,b,d\}$ an, welche der Äquivalenzen gemeint ist. Dabei steht v für die Verweis-, b für die Bezeichner- und d für die Darstellungsäquivalenz.

[8] Für diese Abbildung habe ich den Browser *Google Chrome 10.0* mit aktivierten Entwicklertools benutzt.

(a) Startseite [22] des Instituts für Theoretische Informatik der Universität zu Lübeck

(b) Unterpunkt Stellenangebote

(c) Identische Pfadbeschreibungen der <a href>-Pattern in (a), (b) und (c), die jeweils den Hyperlink auf die Startseite erzeugen. Die Knoten des Pfades sind blau markiert.

Abbildung 17: Beispiel für darstellungsäquivalente <a href>-Pattern

Definition 11 (*x*-IDs, *x*-ID-Zuordnungen)
*Sei P eine Menge von <a href>-Pattern, und Σ ein beliebiges, nichtleeres Alphabet. Dann ist eine Funktion $id\colon P \to \Sigma^+$ eine **x-ID-Zuordnung** unter P, wenn gilt:*

$$\forall p,q \in P\colon \quad id(p) = id(q) \Longleftrightarrow p \text{ und } q \text{ sind zueinander } x\text{-äquivalent}$$

*Der Wert, den eine x-ID-Zuordnung einem <a href>-Pattern zuweist, heißt **x-ID**.*

Eine einfache *v*-ID-Zuordnung ist beispielsweise eine Funktion, die jedem <a href>-Pattern die URL, auf die das betrachtete <a href>-Pattern verweist, zuordnet. Ich werde im Folgenden vor Allem in Darstellungen für *x*-äquivalente <a href>-Pattern häufig durchnummerieren oder Buchstaben nutzen.

3.2.2 Ähnlichkeiten von Gruppen

Im Folgenden werde ich die Äquivalenzen zwischen <a href>-Pattern nutzen, um Gruppen miteinander zu vergleichen. <a href>-Pattern einer Gruppe besitzen eine feste Ordnung. Diese ist identisch zu der Reihenfolge, in der die <a href>-Pattern im Quellcode des Webdokumentes, in dem sie enthalten sind, aufgeführt sind. Diese Ordnung ist für den Vergleich von zwei Gruppen von Bedeutung, da ich voraussetze, dass sich innerhalb von zwei Navigationen, die demselben Navigationssystem angehören, die Reihenfolge der verweis-, bezeichner oder darstellungsäquivalenten Hyperlinks nicht unterscheidet, damit der Eindruck erhalten bleiben kann, es handle sich bei den zwei Navigationen um ein einziges Objekt.

Definition 12 (Miteinander *x*-harmonierende Gruppen, *x*-Harmoniegewichte)
Seien Γ_1 und Γ_2 Gruppen und id_x eine x-ID-Zuordnung unter einer Menge Ω, die $\Gamma_1 \cup \Gamma_2$ enthält. Sei weiterhin $G = (\Gamma_1 \dot\cup \Gamma_2, E)$ ein ungerichteter, bipartiter Graph, für den gilt, dass für jedes Tupel $(p_i^1, p_j^2) \in \Gamma_1 \times \Gamma_2$ genau dann eine Kante zwischen p_i^1 und p_j^2 existiert, wenn gilt $id_x(p_i^1) = id_x(p_j^2)$. Für ein Maximum-Matching $M \subseteq E$ von G der Größe $n = |M|$ seien $p_M^1 = p_{i_1}^1, \ldots, p_{i_n}^1$ bzw. $p_M^2 = p_{j_1}^2, \ldots, p_{j_n}^2$ die Folgen, die alle <a href>-Pattern aus Γ_1 bzw. Γ_2 enthalten, die von M überdeckt werden, und deren Reihenfolge dieselben wie in den Quellcodes der Webdokumente d_1 bzw. d_2 sind.
*Die Gruppen Γ_1 und Γ_2 **x-harmonieren** genau dann unter Ω **miteinander**, wenn ein nichtleeres Maximum-Matching M existiert, aus dem sich $id_x(p_{i_k}^1) = id_x(p_{j_k}^2)$ für alle $1 \leq k \leq n$ ergibt. Harmonieren Γ_1 und Γ_2 miteinander ist das **x-Harmoniegewicht** unter Ω durch n gegeben, andernfalls ist es 0.*

Einfach ausgedrückt, *x*-harmonieren Gruppen dann miteinander, wenn sie eine nichtleere Menge *x*-äquivalenter <a href>-Pattern besitzen, die in derselben Reihenfolge aufgeführt werden.

Anstelle von Maximum-Matchings genügt es für die Überprüfung, ob zwei Gruppen miteinander harmonieren, maximale Matchings zu betrachten. Dies ergibt sich aus dem folgenden Satz.

Behauptung. *Für zwei Gruppen Γ_1 und Γ_2 und einen ungerichteten, bipartiten Graphen $G = (\Gamma_1 \dot\cup \Gamma_2, E)$, für den gilt, dass für jedes Tupel $(p_i, p_j) \in \Gamma_1 \times \Gamma_2$ genau dann eine Kante zwischen*

p_i^1 und p_j^2 existiert, wenn $id_x(p_i^1) = id_x(p_j^2)$ gilt, ist jedes maximale Matching auch ein Maximum-Matching.

Beweis durch Widerspruch. Angenommen $M \subseteq E$ ist ein maximales Matching, aber kein Maximum-Matching des Graphen $G = (\Gamma_1 \dot\cup \Gamma_2, E)$. Dann muss in M mindestens eine Kante weniger enthalten sein als in einem Maximum-Matching. Daraus folgt, dass E zwei Kanten (p_i^1, p_j^2), (p_k^1, p_l^2) sowie entweder (p_i^1, p_l^2) oder (p_k^1, p_j^2) enthalten muss (siehe Abbildung 18), für die $p_i^1 \neq p_k^1$ und $p_j^2 \neq p_l^2$ gilt, und M (je nachdem, welche Kante sich in E befindet) die Verpaarung $(p_i^1, p_l^2) \in M$ bzw. $(p_k^1, p_j^2) \in M$ enthält.

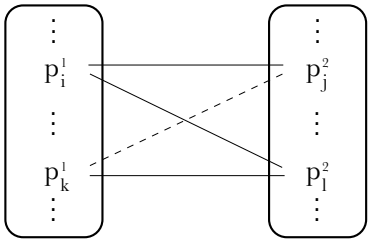


Abbildung 18: Schematische Darstellung der Gruppen Γ_1 und Γ_2.

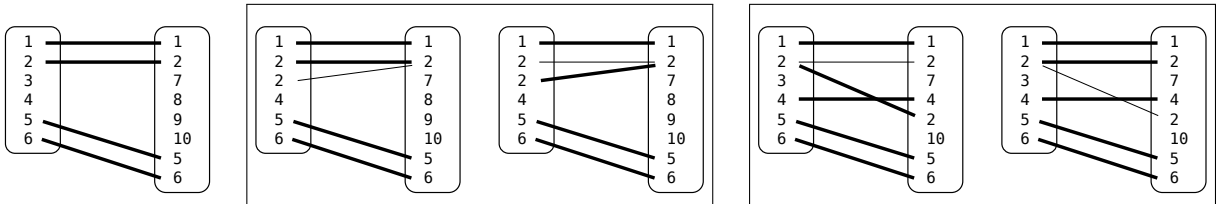
In diesem Fall können sich weder (p_i^1, p_j^2) noch (p_k^1, p_l^2) in M befinden. Aus diesem Grund muss M mindestens eine Kante weniger enthalten als ein Maximum-Matching. Wären beide Kanten (p_i^1, p_l^2) oder (p_k^1, p_j^2) in E enthalten, wäre M nicht maximal. Wurde $(p_i^1, p_l^2) \in M$ gewählt, gilt:

$$\begin{aligned}
(p_i^1, p_j^2), (p_k^1, p_l^2), (p_i^1, p_l^2) \in E \quad &\Rightarrow \quad (id_x(p_i^1) = id_x(p_j^2)) \wedge (id_x(p_i^1) = id_x(p_l^2)) \wedge (id_x(p_k^1) = id_x(p_l^2)) \\
&\Rightarrow \quad (id_x(p_j^2) = id_x(p_i^1)) \wedge (id_x(p_i^1) = id_x(p_l^2)) \wedge (id_x(p_l^2) = id_x(p_k^1)) \\
&\Rightarrow \quad id_x(p_j^2) = id_x(p_k^1) \\
&\Rightarrow \quad (p_k^1, p_j^2) \in E
\end{aligned}$$

Analog muss sich für den Fall $(p_i^1, p_j^2), (p_k^1, p_l^2), (p_k^1, p_j^2) \in E$ auch (p_i^1, p_l^2) in E befinden. Dies widerspricht der Annahme, dass entweder (p_i^1, p_l^2) oder (p_k^1, p_j^2) in E vorhanden sind, aber nicht beide. Somit kann M kein maximales Matching sein, das kein Maximum-Matching ist. □

Enthalten zwei Gruppen Γ_1 und Γ_2 jeweils mehrere <a href>-Pattern derselben x-ID, so ist für das x-Harmoniegewicht sichergestellt, dass jedes <a href>-Pattern aus Γ_1 maximal einem <a href>-Pattern aus Γ_2 zugeordnet werden kann, da dieses Gewicht der Größe eines Matchings entspricht, das sich zwischen den <a href>-Pattern aus Γ_1 und Γ_2 ergibt, und es in der Definition eines Matchings liegt, einen Knoten (in diesem Fall ein <a href>-Pattern) mit maximal einem anderen Knoten zu verpaaren. In Abbildung 19 sind drei verschiedene Szenarien für jeweils zwei miteinander harmonierende Gruppen dargestellt. In Abbildung 19(a) gibt es nur genau ein maximales Matching. Dieses erfüllt die Bedingung, dass die Reihenfolge der x-äquivalenten <a href>-Pattern identisch ist (Reihenfolgekriterium). Für die Gruppen aus (b) und (c) gibt es genau zwei verschiedene maximale Matchings, die beide das Reihenfolgekriterium erfüllen. Für (c) und (d) gibt es ebenfalls zwei verschiedene maximale Matchings, doch nur das Zweite erfüllt das

Reihenfolgekriterium. Dies ist aber für das Harmonieren von zwei Gruppen völlig ausreichend.



(a) Eindeutiges Matching, Reihenfolge gleich

(b) Matching 1, Reihenfolge gleich

(c) Matching 2, Reihenfolge gleich

(d) Matching 1, Reihenfolge ungleich

(e) Matching 2, Reihenfolge gleich

Abbildung 19: Harmonierende Gruppen, teilweise mit unterschiedlichen möglichen Matchings

3.2.3 Algorithmische Überprüfung von *x*-Harmonien

Die algorithmische Überprüfung, ob zwei Gruppen Γ_1 und Γ_2 miteinander *x*-harmonieren, umfasst bei gegebener *x*-ID-Zuordnung die Erstellung des bipartiten Graphen $G = (\Gamma_1 \dot{\cup} \Gamma_2, E)$, die Berechnung maximaler Matchings sowie die Überprüfung des Reihenfolgekriteriums.

Erstellung des bipartiten Graphen: Hierzu muss jeder Knoten aus Γ_1 mit jedem Knoten aus Γ_2 verglichen werden und gegebenenfalls eine Kante zwischen den Knoten in G eingefügt werden. Dies beansprucht einen Zeitaufwand in $O(|\Gamma_1| \cdot |\Gamma_2|)$, wenn wir für den Vergleich der IDs einen Berechnungsaufwand in $O(1)$ annehmen.

Bestimmung von maximalen Matchings: Im Gegensatz zur Bestimmung eines Maximum-Matchings lässt sich ein maximales Matching sehr einfach bestimmen. Dafür kann jeder Knoten der kleineren Menge von Γ_1 und Γ_2 gewählt werden und dann jedem Knoten, der einer Kante in E angehört, eine beliebige Kante, aber eben nur eine, zugewiesen werden. Dies ist in $O(\min(|\Gamma_1|, |\Gamma_2|))$ möglich.

Überprüfung des Reihenfolgekriteriums: Ob das Reihenfolgekriterium erfüllt werden kann, kann mit Hilfe eines Algorithmus zur Lösung des *Longest Common Subsequence* – Problems überprüft werden. Die Problemstellung eines solchen Problems ist, auf Eingabe von k Strings, in diesem Fall mit $k = 2$, die längste Teilsequenz, die in beiden Strings gemeinsam vorkommt, zu bestimmen. Dabei beschreibt eine Teilsequenz eine Menge von Symbolen innerhalb eines Strings, dessen Reihenfolge erhalten bleibt, die aber nicht direkt hintereinander aufgeführt werden müssen. Wenn wir die *x*-IDs von zwei Gruppen Γ_1 und Γ_2 nicht mehr als Knoten, sondern jeweils als einzelne Symbole und die geordnete Menge aller *x*-IDs der Gruppen als Strings dieser Symbole betrachten, dann können wir einen Algorithmus, der das Longest Common Subsequence – Problem löst, auf unsere Frage, ob die beiden Gruppen miteinander harmonieren, direkt anwenden.

Dabei ist für uns nur interessant, ob die ermittelte größte Teilsequenz genauso groß ist wie ein maximales Matching zwischen den Gruppen. (Aus diesem Grund haben wir zunächst den Graphen

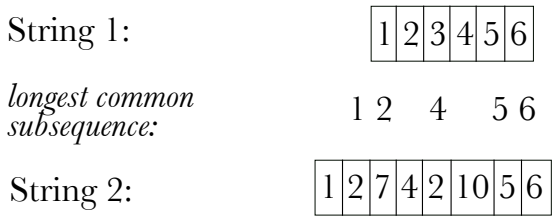


Abbildung 20: Longest Common Subsequence für das Beispiel aus Abbildung 19 (d) und (e)

und dann ein beliebiges maximales Matching erstellt.) Algorithmen, die das Longest Common Subsequence – Problem für zwei Strings lösen, sind in $O(n \cdot m)$ möglich [6], wobei n und m jeweils die Längen der betrachteten Strings beschreiben. Entsprechend kann auch die Feststellung, ob die Gruppen Γ_1 und Γ_2 miteinander x-harmonieren mit einer Laufzeit in $O(|\Gamma_1| \cdot |\Gamma_2|)$ bestimmt werden.

3.3 Verlinkungsstruktur in Navigationssystemen

Die Navigationssysteme, die ich in dieser Arbeit betrachte, sollen eine typische Verlinkungsstruktur aufweisen. Diese werde ich im Folgenden formal beschreiben und vorstellen, wie sie algorithmisch überprüft werden kann.

3.3.1 Formale Beschreibung

Sei im Folgenden $V_S \subseteq S$ eine Menge von Navigationen eines Navigationssystems S. Dann soll in Anlehnung an meine Charakterisierung von Navigationen und Navigationssystemen gelten, dass zwei verschiedene Navigationen $v_i, v_j \in V_S$ auch verschiedenen Webdokumenten angehören. Sei im Folgenden zu jeder Navigation $v_i \in V_S$ das Webdokument d_i das Webdokument, das v_i enthält. Dann kann ein Hyperlink, der von einer Navigation v_i ausgeht und auf ein Webdokument d_j zeigt, das eine Navigation $v_j \in V_S$ enthält, auch als Hyperlink zwischen v_i und v_j betrachtet werden. Im Folgenden verwende ich aus diesem Grund vereinfachend auch die Bezeichnung, ein Hyperlink verweise von einer Navigation v_i auf eine Navigation v_j, obwohl dies rein technisch natürlich nicht der Fall ist. Auf diese Weise kann man den Graphen G_S definieren.

$G_S = (V_S, E_S)$ ist ein gerichteter Graph. Für jedes <a href>-Pattern p einer Navigation $v_i \in V_S$, das auf eine Navigation $v_j \in V_S$ verweist, existiert genau eine Kante $(v_i, v_j) \in E_S$. Erfüllt der Graph G_S bestimmte Anforderungen, stellt er einen *Navigationsgraphen* dar. Dazu benötige ich jedoch zuerst einen *Hauptnavigationsgraphen.*

Definition 13 (Hauptnavigationsgraph)
Sei $G_S = (V_S, E_S)$ ein gerichteter Graph und sei V_H die Menge aller Knoten $v_h \in V_S$, für die gilt $\forall v \in V_S$ mit $v \neq v_h$: $(v, v_h) \in E_S$. Dann ist der ***Hauptnavigationsgraph*** *der (vollständige) Teilgraph von G_S, der durch V_H induziert wird.*

Definition 14 (Navigationsgraph)
Ein ***Navigationsgraph*** *ist ein gerichteter Graph $G_S = (V_S, E_S)$, der einen nicht-leere Hauptnavigationsgraphen $G_H = (V_H, E_H)$ mit $V_H = \{h_1, \ldots, h_n\}$ enthält, für den gilt: Der Graph $G_{neu} =$*

$(V_S, E_S \setminus E_H)$ enthält genau $n = |V_H|$ verschiedene, starke Zusammenhangskomponenten $G_{h_1} = (V_{h_1}, E_{h_1})$ bis $G_{h_n} = (V_{h_n}, E_{h_n})$. Dabei gilt für jede starke Zusammenhangskomponente G_{h_i}, dass der Knoten v_{h_i} in der Knotenmenge V_{h_i} enthalten ist und der Subgraph G_i von G_{h_i}, der durch die Knotenmenge $V_{h_i} \setminus \{h_i\}$ induziert wird, entweder leer oder ebenfalls ein Navigationsgraph ist. Sei G_H^i ein nichtleerer Hauptnavigationsgraph eines Graphen G_i. Dann heißt G_H^i ***Unternavigationsgraph*** *von G_H.*

Da ich innerhalb der Charakterisierung der Navigationen und Navigationssysteme beschreibe, dass die Verlinkungsstruktur eines Navigationssystems einem Navigationsgraphen entsprechen sollte, muss für jedes Navigationssystem, das in dieser Arbeit erkannt werden soll, eine nichtleere Menge von Navigationen $V_H \subseteq V_S$ existieren, auf die von jeder Navigation aus S (außer in Form einer Schleife auf sich selbst) verwiesen wird. Ein <a href>-Pattern, das auf eine Navigation des Hauptnavigationsgraphen verweist, nenne ich Hauptpunkt. Jedes <a href>-Pattern p einer Navigation $h_i \in V_H$, das auf eine Navigation $v \in V_S$ verweist, aber kein Hauptpunkt ist, nenne ich direkten Unterpunkt von h_i. Ein direkter Unterpunkt eines direkten Unterpunktes von h_i heißt dann Unterpunkt von h_i. Ebenso heißt jeder Unterpunkt eines direkten oder nicht direkten Unterpunktes von h_i ebenfalls Unterpunkt von h_i. Die Navigationen und Verlinkungen des Hauptnavigationsgraphen stellen ein eigenes Navigationssystem, ein Hauptnavigationssystem dar. Dieses fasse ich als eine eigene Navigationsebene auf und betrachte auch die Webdokumente, auf die die direkten Unterpunkte eines Webdokumentes des Hauptnavigationssystems verweisen, jeweils als eigene Navigationsebenen.

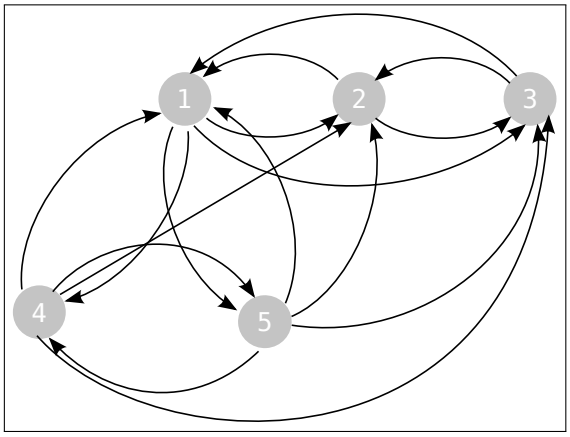
Behauptung. *Sei der Graph $G_S = (V_S, E_S)$ ein Navigationsgraph und $G_H = (V_H, E_H)$ der Hauptnavigationsgraph von G_S. Dann muss gelten, dass kein Knoten $h_1 \in V_H$ auf einen Unterpunkt eines anderen Knotens $h_2 \in V_H$ verweist.*

Beweis. Der Graph $G_{neu} = (V_S, E_S \setminus E_H)$ soll nach Definition 14 genau $n = |V_H|$ verschiedene, starke Zusammenhangskomponenten $G_{h_i} = (V_{h_i}, E_{h_i})$ besitzen. Dabei soll jede Zusammenhangskomponente G_{h_i} jeweils den Knoten $h_i \in V_H$ enthalten. Da ein Knoten h_i nur genau dann dem Hauptnavigationsgraphen angehört, wenn alle Knoten aus V_S auf ihn verweisen, muss also eine Kante von einem Unterpunkt eines Knoten h_i zu jedem weiteren Knoten $h_j \in V_H$ existieren. Würde nun auch von $h_j \neq h_i$ eine Kante zu diesem Unterpunkt von h_i existieren, so würden h_i und h_j derselben Zusammenhangskomponente angehören bzw. die Zusammenhangskomponenten G_{h_i} und G_{h_j} wären gleich. Dies widerspricht aber der Forderung, dass genau n verschiedene Zusammenhangskomponenten G_{h_i} existieren, die jeweils den Knoten h_i enthalten. □

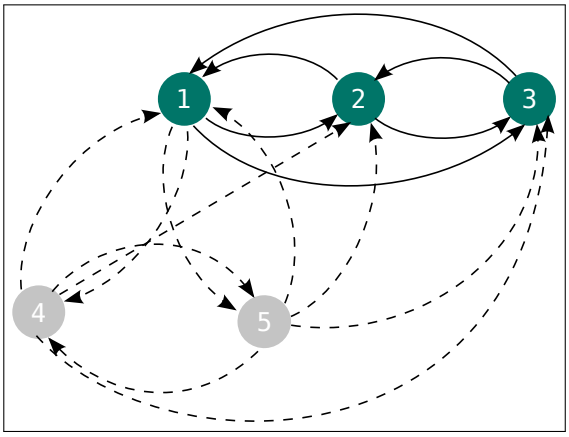
Ein direkter Unterpunkt ist entsprechend immer genau einer Navigation des Hauptnavigationsgraphen direkt zugeordnet. Das heißt, dass kein <a href>-Pattern einer Navigation h_i des Hauptnavigationsgraphen auf eine Navigation zeigt, auf das ein Unterpunkt einer von h_i verschiedenen Navigation des Hauptnavigationsgraphen verweist. Diese Regel schließt den unerwünschten Nebeneffekt ein, dass ein Webdokument, das von einer Navigationsebene verlinkt wird, in keiner anderen Navigationsebene desselben Navigationssystems auftauchen darf. Ein von mir konstruiertes Beispiel ist ein Navigationssystem, das als Hauptpunkte <a href>-Pattern

mit dem Bezeichner „Bachelor“ und „Master“ enthält und in dem ein direkter Unterpunkt des Hauptpunktes „Bachelor“ auf eine Navigation in einem Webdokument d verweist, das sowohl die Bachelor- als auch die Masterprüfungsordnung enthält. Entsprechend wird dasselbe Webdokument auch als direkter Unterpunkt des Hauptpunktes „Master“ verlinkt. Ein solcher Fall kann durchaus häufiger auftreten und eine Lösung des Problems wäre sinnvoll. In dieser Arbeit werde ich auf diesen Fall allerdings nicht näher eingehen. Aus der Definition geht außerdem hervor, dass jeder Unternavigationsgraph eines Navigationsgraphen wieder einen Navigationsgraphen darstellt und dass jede Navigationsebene für sich ein Hauptnavigationssystem ist.

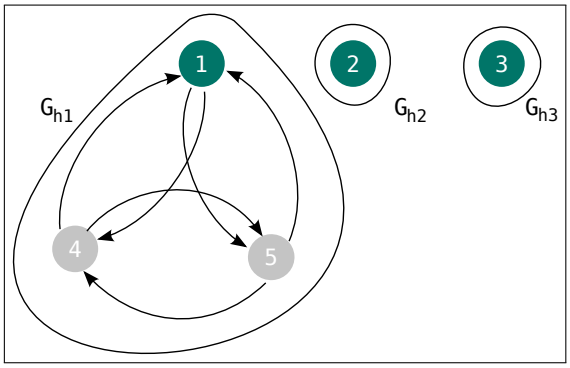
In Abbildung 21 ist ein Beispiel für einen Navigationsgraphen unter verschiedenen Blickwinkeln dargestellt. In (a) ist der Navigationsgraph vollständig dargestellt, in (b) sind die Kanten durchgezogen, die für die Analyse der Unternavigationsgraphen entfernt werden. Alle anderen sind gestrichelt dargestellt. In (c) sind die einzelnen starken Zusammenhangskomponenten eingerahmt und alle Kanten, die zwischen den starken Zusammenhangskomponenten existierten, entfernt. In (d) ist dann nur noch ein Unternavigationsgraph zu sehen. Da die Knoten gegenseitig aufeinander zeigen, erfüllen sie die Bedingung des Hauptnavigationsgraphen. Alle Kriterien, die die Definitionen *Navigationsgraph* und *Hauptnavigationsgraph* mit sich bringen, sind in diesem Beispiel also erfüllt.



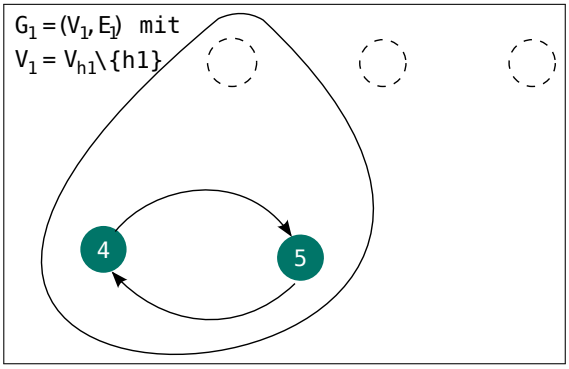
(a) Eingabegraph G_S



(b) Hauptnavigationsgraph $G_H = (V_H, E_H)$ mit V_H grün und E_H durchgezogen darsgestellt



(c) eingekreiste Zusammenhangskomponenten, G_{h_2} und G_{h_3} trivial, G_{h_1} nicht trivial



(d) Neuer Eingabegraph G_1 nach Entfernen des Hauptnavigationspunktes h_1

Abbildung 21: Überprüfung eines Navigationsgraphen

3.3.2 Algorithmische Überprüfung der Verlinkungsstruktur

In diesem Abschnitt werde ich beschreiben, wie gerichtete Graphen algorithmisch auf ihre Verlinkungsstruktur untersucht werden können. Für die Betrachtung der Laufzeit nehme ich dieselben Operationen als elementar, also mit einer Laufzeit in $O(1)$ an, wie bereits bei der Analyse des *gruppiereKnoten*-Algorithmus (siehe S. 27).

Algorithmus *bestimmeHauptnavigationsknoten*(G)

Ist ein Graph $G = (V, E)$ gegeben, kann mithilfe des Algorithmus *bestimmeHauptnavigationsknoten*(G) festgestellt werden, ob G einen Hauptnavigationsgraphen enthält oder nicht. Enthält G einen Hauptnavigationsgraphen, werden alle Knoten, die dem Hauptnavigationsgraphen angehören, zurückgegeben, andernfalls die leere Menge. Innerhalb des Algorithmus wird dafür für jeden Knoten v einzeln geprüft, ob von jedem von ihm verschiedenen Knoten v' eine Kante ausgeht, die auf v zeigt. Ist dies nicht der Fall, gehört v nicht dem Hauptnavigationsgraphen an $(isHauptNav \leftarrow false)$.

Algorithm 3 *bestimmeHauptnavigationsknoten*(G)

Input: gerichteter Graph $G = (V, E)$
Output: Knotenmenge V_H, die alle Knoten der Hauptnavigation enthält

```
1:  for each v ∈ V do
2:      istHauptNav ← true
3:      for each v' ∈ V where v' ≠ v do
4:          if (v', v) ∉ E then
5:              istHauptNav ← false
6:              break
7:          end if
8:      end for
9:      if (istHauptNav == true) then
10:         V_H ← V_H ∪ {v}
11:     end if
12: end for
13: return V_H
```

Die Laufzeit des Algorithmus liegt in $O(|V|+|E|)$. Unabhängig davon, wie viele Kanten existieren, wird jedem Knoten in V einmal $isHauptNav \leftarrow true$, und gegebenenfalls einmal $isHauptNav \leftarrow false$ zugewiesen. Gehört der betrachtete Knoten dem Hauptnavigationsgraphen an, wird er zusätzlich einmal in die Menge V_H eingefügt. Darüber hinaus muss jede existierende Kante in Zeile 4 maximal einmal betrachtet werden. Dies führt zu einer Laufzeit in $O(|V|+|E|)$.
Der Algorithmus *bestimmeHauptnavigationsknoten* ist Teil des Algorithmus *istNavigationsgraph*. Neben dem Algorithmus *bestimmeHauptnavigationsknoten* greift dieser auf den Algorithmus *bestimmeUndPruefeSZK* zurück, den ich im Folgenden vorstellen werde.

Algorithmus *bestimmeUndPruefeSZK* (G_{neu}, V_H)

Sei $G = (V,E)$ ein gerichteter Graph, der einen Hauptnavigationsgraphen enthält und G_{neu} der Graph, der aus G nach Entfernen aller Kanten, die zwischen allen Knoten des Hauptnavigationsgraphen existieren, entsteht. Dann sollen alle starken Zusammenhangskomponenten, die sich in G_{neu} befinden, berechnet und darauf überprüft werden, ob jede starke Zusammenhangskomponente genau einen Knoten des Hauptnavigationsgraphen enthält. Ist die Überprüfung erfolgreich, soll der Algorithmus die Knotenmengen V_{h_i} der Zusammenhangskomponenten G_{h_i} zurückgeben, ansonsten die leere Menge.

Ein Algorithmus zur Bestimmung aller starken Zusammenhangskomponenten eines gerichteten Graphen wurde von Tarjan in [12] bereits vorgeschlagen. Dieser Algorithmus benötigt eine Laufzeit in $O(|V|+|E|)$. Ohne eine wesentliche Veränderung der Laufzeit kann der Algorithmus von Tarjan so verändert werden, dass nach dem Erkennen einer starken Zusammenhangskomponente sofort geprüft wird, ob genau ein Hauptnavigationsknoten enthalten ist. Dafür müssen dem Algorithmus die Knoten des Hauptnavigationsgraphen bekannt sein. Aus diesem Grund stellen G_{neu} und auch V_H die Eingabe für *bestimmeUndPruefeSZK* dar. Da starke Zusammenhangskomponenten eine Partition des Graphen G_{neu} darstellen, liegt die Überprüfung für alle Zusammenhangskomponenten gemeinsam in $O(|V|)$. Die Gesamtlaufzeit des Algorithmus ist also auch mit der Überprüfung der starken Zusammenhangskomponenten linear in der Größe des Graphen.

Algorithmus *istNavigationsgraph* (G)

Mit dem Algorithmus *istNavigationsgraph*(G) kann entschieden werden, ob ein gegebener Graph $G = (V,E)$ einen Navigationsgraphen darstellt oder nicht. Die algorithmischen Schritte orientieren sich dabei inhaltlich sowie in ihrer Reihenfolge sehr an der Definition 14.

Korrektheit

Die Definition eines Navigationsgraphen enthält mehrere Anforderungen, die ich hier noch einmal in Punkten darstelle.

1. *Ein* **Navigationsgraph** *ist ein gerichteter Graph* $G_S = (V_S, E_S)$*, der einen nicht-leeren Hauptnavigationsgraphen* $G_H = (V_H, E_H)$ *mit* $V_H = \{h_1, \ldots, h_n\}$ *enthält*
2. *Der Graph* $G_{neu} = (V_{nav}, E_S \setminus E_H)$ *enthält genau* $n = |V_H|$ *verschiedene, starke Zusammenhangskomponenten* $G_{h_1} = (V_{h_1}, E_{h_1})$ *bis* $G_{h_n} = (V_{h_n}, E_{h_n})$*. Dabei gilt für jede starke Zusammenhangskomponente* G_{h_i}*, dass der Knoten* v_{h_i} *in der Knotenmenge* V_{h_i} *enthalten ist ...*
3. *... und der Subgraph* G_i *von* G_{h_i}*, der durch die Knotenmenge* $V_{h_i} \setminus \{h_i\}$ *induziert wird, entweder leer oder ebenfalls ein Navigationsgraph ist.*

Der Punkt 1 wird im Algorithmus sofort in Zeile 1 geprüft und für den Fall, dass kein Hauptnavigationsgraph vorhanden (V_H leer) ist, ausgegeben, dass G keinen Navigationsgraphen darstellt.

Algorithm 4 *istNavigationsgraph*(G)

Input: Graph $G = (V, E)$

Output: Navigationsgraph, kein Navigationsgraph

1: $V_H \leftarrow bestimmeHauptnavigationsknoten(G)$
2: **if** $(V_H == \emptyset)$ **then**
3: **return** kein Navigationsgraph
4: **else**
5: $E_{neu} \leftarrow E$
6: **for each** $(h_i, h_j) \in V_H \times V_H$ **do**
7: $E_{neu} \leftarrow E_{neu} \setminus (h_i, h_j)$
8: **end for**
9: $G_{neu} \leftarrow (V, E_{neu})$
10: $SZK \leftarrow \emptyset$
11: $SZK \leftarrow bestimmeUndPruefeSZK(G_{neu}, V_H)$
12: **if** $(SZK == \emptyset)$ **then**
13: **return** kein Navigationsgraph
14: **else**
15: **for each** $(V_{h_i} \in SZK)$ **do**
16: **if** $(|V_{h_i}| > 1)$ **then**
17: *Sei G_i der Subgraph von G_{neu}, der durch die Knotenmenge $V_{h_i} \setminus \{h_i\}$ induziert wird.*
18: **if** $(istNavigationsgraph(G_i) \neq true)$ **then**
19: **return** kein Navigationsgraph
20: **end if**
21: **end if**
22: **end for**
23: **return** ist Navigationsgraph
24: **end if**
25: **end if**

In den Zeilen 5 bis 8 wird Punkt 2 vorbereitet, indem die Kantenmenge $E_S \setminus E_H$ der Definition im Algorithmus als E_{neu} konstruiert wird. Zunächst wird dafür E_{neu} mit E gleichgesetzt. G_H muss ein vollständiger, gerichteter Graph sein, da es sich um einen Hauptnavigationsgraphen handelt, in der jeder Knoten auf jeden anderen Knoten zeigt. Entsprechend enthält E_H jede mögliche Kante $(h_i, h_j) \in V_H \times V_H$. In der Definition werden diese Kanten von E_S abgezogen. Analog wird dies für E_{neu} im Algorithmus in Zeile 6-8 durchgeführt. In Zeile 9 ist der Graph $G = (V, E_{neu})$ also identisch mit dem in der Definition geforderten Graphen $G = (V, E \setminus E_H)$. Für diesen werden alle Zusammenhangskomponenten berechnet und wie bereits beschrieben geprüft. Die Knotenmengen V_{h_i} der erfolgreich überprüften Zusammenhangskomponenten G_{h_i} werden in SZK eingefügt. War die Prüfung der Zusammenhangskomponenten nicht erfolgreich, nimmt SZK die leere Menge an und es wird ausgegeben, dass G keinen Navigationsgraphen darstellt. Damit ist auch Punkt 2 durch den Algorithmus abgedeckt.

Die dritte und letzte Anforderung wird in Zeile 15 bis 22 geprüft. Stellt jeder durch die Knotenmenge $V_{h_i} \setminus h_i$ induzierte, nichtleere Subgraph G_i von G einen Navigationsgraphen dar, wird

ausgegeben, dass G ein Navigationsgraph ist. Andernfalls wird in Zeile 19 „kein Navigationsgraph“ ausgegeben. Auch dieser Punkt wird also durch den Algorithmus abgedeckt.

Weitere Operationen und Überprüfungen werden durch den Algorithmus *istNavigationsgraph* nicht vorgenommen. Entsprechend ist die Ausgabe des Algorithmus korrekt.

Laufzeit

Die Laufzeit des Algorithmus *isNavigationsgraph*(G) mit $G = (V,E)$ liegt in $O(|V| \cdot (|V| + |E|)$. Dies werde ich zeilenweise zeigen.

Zeile 1-4: Der Algorithmus *bestimmeHauptnavigationsknoten*(G) liegt in $O(|V| + |E|)$. Dies habe ich bereits beschrieben.

Zeile 5-10: Die Zeilen 5-8 müssen jeweils maximal $|E|$ Kanten kopiert werden. Entsprechend liegt der Berechnungsaufwand dafür entsprechend in $O(|E|)$. Die Zuweisung in Zeile 9 beinhaltet das Kopieren der Knoten sowie der Kanten und liegt damit in $O(|V| + |E|)$.

Zeile 11-14: In der Beschreibung des Algorithmus *bestimmeUndPruefeSZK*(G_{neu}, V_H) habe bereits angemerkt, dass Algorithmen wie der Algorithmus von Tarjan existieren, für die die Bestimmung aller starken Zusammenhangskomponenten in $O(|V| + |E|)$ liegt und auch eine Überprüfung der Zusammenhangskomponenten auf die Anforderungen des Punktes 2 der Definition diese Laufzeit nicht erhöhen.

Zeile 15-22: Sind alle verbliebenen starken Zusammenhangskomponenten trivial, enthalten also nur noch einen Knoten, so liegt der Aufwand für diese Zeilen in $O(|V_h|)$, da die Größe von $|V_h|$ Zusammenhangskomponenten überprüft werden muss. In diesem Fall liegt die gesamte Laufzeit des Algorithmus in $O(|V| + |E|)$.

Wird die Rekursion jedoch genutzt - dies ist der Fall, wenn nicht-triviale Zusammenhangskomponenten $G_{h_i} = (V_{h_i}, E_{h_i})$ berechnet wurden - dann wird der Algorithmus für die Subgraphen $G_i = (V_i, E_i)$, die durch die jeweilige Knotenmenge $V_{h_i} \setminus \{h_i\}$ induziert werden, erneut aufgerufen. Für jeden dieser Graphen G_i bedeutet dies also einen zusätzlichen Aufwand in $O(|V_i| + |E_i|)$. Da die Menge aller starken Zusammenhangskomponenten aber eine Partition des Graphen G darstellt, kann der Berechnungsaufwand der nächsten Rekursionsebene maximal in $O(\sum_{h_i \in V_H}(|V_i| + |E_i|)) \subseteq O(|V| + |E|)$ liegen. Da vor jedem neuen Rekursionsschritt aus jeder nicht-trivialen starken Zusammenhangskomponente außerdem jeweils ein Knoten (der Knoten h_i aus jedem G_{h_i}) entfernt wird, kann es maximal $|V|$ Iterationsebenen geben. Entsprechend liegt der Berechnungsaufwand des gesamten Algorithmus in $O(|V| \cdot (|V| + |E|))$.

4 Navigationserkennung

In diesem Kapitel werde ich vorstellen, wie mit Hilfe der eingeführten Elemente Navigationen und Außenseiter erkannt werden können. Bisher habe ich dazu angenommen, dass als Eingabe ein Webdokument gegeben ist und das Ziel ist, für dieses Webdokument alle Hyperlinks zu bestimmen, die von einer Navigation oder einem Außenseiter ausgehen. Für die Feststellung, welche Gruppen eines Webdokumentes Navigationen oder Außenseiter darstellen, muss jede Gruppe einzeln betrachtet werden, da eine von mir getroffene Annahme für Navigationssysteme die ist, dass verschiedene Navigationen eines Navigationssystems auch verschiedenen Webdokumenten angehören. Entsprechend können sich nicht zwei Navigationen desselben Navigationssystems in demselben Webdokument befinden. Ebenso befinden sich auch Außenseiter eines Navigationssystems S nicht in demselben Webdokument wie Navigationen von S. Damit muss mit jeder Gruppe eines Webdokumentes auch jedes Mal ein neues Navigationssystem überprüft werden. Aus diesem Grund werde ich im Folgenden nur den Fall behandeln, dass eine einzige Gruppe (im Folgenden *Referenzgruppe*) eines Webdokumentes gegeben ist und für diese entschieden werden soll, ob es sich um eine Navigation, einen Außenseiter oder keins von beidem handelt.

4.1 Nachbardokumente und Gruppennachbarschaft

Da die Referenzgruppe als Eingabe bereits bekannt ist, sind nur noch Systemeigenschaften zu prüfen, da die an ein einziges Webdokument gebundenen Navigationseigenschaften bei der Berechnung der Gruppe bereits berücksichtigt wurden. Um im Folgenden auch die Systemeigenschaften untersuchen zu können, müssen weitere Webdokumente bestimmt werden, deren Gruppen gemeinsam mit der Referenzgruppe Aufschluss darüber geben können, ob die Referenzgruppe eine Navigation oder einen Außeseiter eines Navigationssystems darstellt oder nicht.

Eine Systemeigenschaft, die ich in Kapitel 3.3 beschrieben habe, fordert, dass jedes Navigationssystem ein Hauptnavigationssystem enthält. Das heißt, jede Navigation muss <a href>-Pattern enthalten, die auf die Navigationsdokumente des Hauptnavigationssystems zeigen. Schon allein aus diesem Grund bietet es sich an, zunächst Webdokumente zu betrachten, auf die <a href>-Pattern der Referenzgruppe zeigen. Dabei werden Transverse und Intrinsics unterschiedlich behandelt.

Transverse. Die von mir charakterisierten Navigationssysteme weisen die Eigenschaft auf, dass sich die Navigationen des Navigationssystems innerhalb derselben Domain befinden. Das heißt, dass ich nur innerhalb der Domain Navigationen des Navigationssystems erwarte und prüfe. Dennoch darf eine Navigation auch <a href>-Pattern enthalten, die auf Webdokumente außerhalb der Domain zeigen (Transverse). Ein Beispiel dafür ist in Abbildung 22 zu sehen. In der Navigation links ist unter anderem der Punkt *University* aufgeführt. Dieser führt zu einem Webdokument einer anderen Domain.

Für diese <a href>-Pattern muss im Navigationssystem grundsätzlich auch das Reihenfolgekriterium erfüllt sein. D.h. dass die verweisäquivalenten <a href>-Pattern von jedem Paar von Navigationen eines Navigationssystems, egal ob sie Intrinsics oder Transverse darstellen, in derselben Reihenfolge aufgeführt werden müssen. Transverse werden also ebenfalls betrachtet. Lediglich die Webdokumente, auf die sie zeigen, können unbetrachtet bleiben.

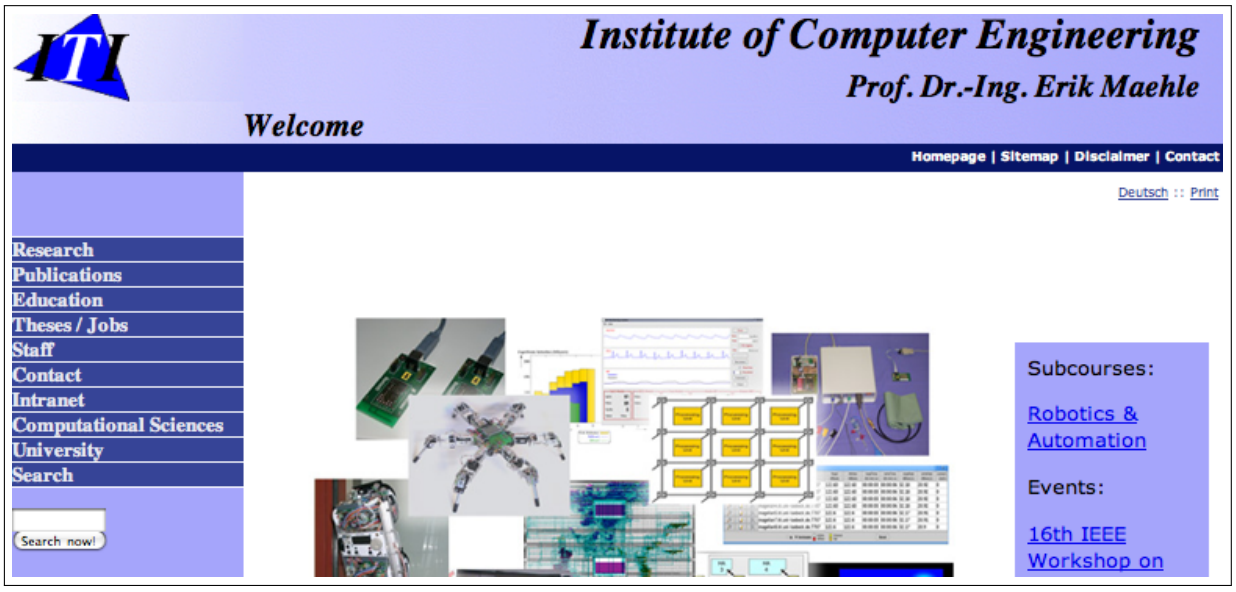


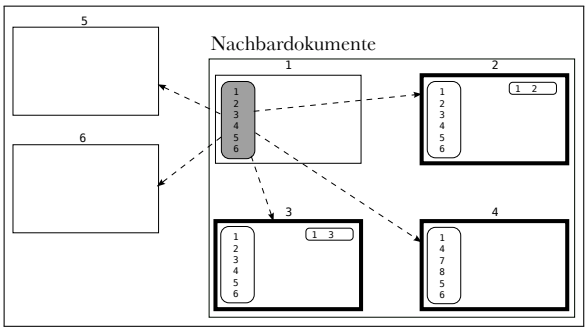
Abbildung 22: Ausschnitt der englischsprachigen Startseite [23] des Instituts für Technische Informatik der Universität zu Lübeck

Intrinsics. Ist ein Intrinsic in einer Navigation vorhanden, so muss das Webdokument, auf das der Intrinsic zeigt, auch allen Anforderungen als Navigationsdokument genügen. Das heißt, es muss ebenfalls eine Navigation beinhalten und gemeinsam mit den anderen Navigationsdokumenten die Verlinkungsstruktur einhalten. In der Praxis ist das nicht immer der Fall. In Abbildung 22 ist ein Hyperlink mit dem Bezeichner *Intranet* enthalten. Dieser führt zu einem Webdokument innerhalb der Domain, das keine Navigation enthält. Mit den in dieser Arbeit vorgestellten Modellen und Algorithmen wird ein solcher Fall noch nicht als Navigationssystem erkannt.

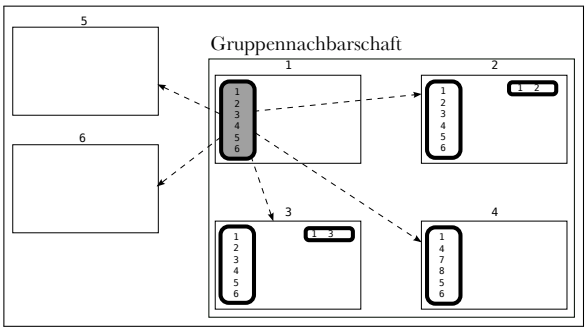
Mit den in Kapitel 3 vorgestellten Methoden können wir Gruppen für Webdokumente bestimmen, Gruppen miteinander vergleichen und für einen gegebenen Graphen die Verlinkungsstruktur überprüfen. Wie wir diesen Graphen erhalten, habe ich aber noch nicht beschrieben. Einen ersten Schritt zum Auffinden von Navigationsdokumenten habe ich bereits angemerkt. Ausgehend von der Referenzgruppe Γ_{Ref} werde ich jedes domaininterne Webdokument betrachten, auf das Γ_{Ref} zeigt. Diese Menge von Webdokumenten nenne ich *Nachbardokumente* von Γ_{Ref}. Wird für jedes Nachbardokument die Gruppierung bestimmt, erhalten wir außerdem eine Menge von Gruppen, unter denen sich für jedes Webdokument eine Navigation befinden sollte. Diese Menge von Gruppen und Γ_{Ref} heißt *Gruppennachbarschaft* von Γ_{Ref}.

Definition 15 (Nachbardokumente, Gruppennachbarschaft)
Sei d_{Ref} ein Webdokument und Γ_{Ref} eine Gruppe aus d_{Ref}. Dann ist die Menge der ***Nachbardokumente*** *von Γ_{Ref} die Menge $D_{Ref} = \{d_1, \ldots, d_n\}$, die alle d_i enthält, die derselben Domain wie d_{Ref} angehören und auf die mindestens ein <a href>-Pattern aus Γ_{ref} verweist. Sei außerdem Ω_i für jedes $d_i \in D_{ref}$ die Gruppierung von d_i. Dann ist die* ***Gruppennachbarschaft*** *von Γ_{Ref} die Menge $\Omega = \Omega_1 \cup \ldots \cup \Omega_n \cup \{\Gamma_{Ref}\}$.*

Enthält die Gruppennachbarschaft der Referenzgruppe ausschließlich die Referenzgruppe, wird diese nicht weiter auf Navigationseigenschaften untersucht. Ein Beispiel für Nachbardokumente und eine Gruppennachbarschaft ist in Abbildung 23 zu sehen.



(a) Nachbardokumente (fett eingerahmt) der grau dargestellten Referenzgruppe



(b) Gruppennachbarschaft (fett eingerahmt) der Referenzgruppe

Abbildung 23: Nachbardokumente und Gruppennachbarschaft der Referenzgruppe (grau bzw. grün ausgefüllt). Die Webdokumente 5 und 6 (links) liegen außerhalb der Domain, alle anderen Webdokumente innerhalb derselben Domain. Gestrichelte Pfeile zeigen, auf welche Webdokumente die Referenzgruppe verweist.

Angenommen, die Referenzgruppe stellt eine Navigation eines Navigationssystems S dar. Dann gehört Γ_{Ref} entweder dem Hauptnavigationssystem von S an oder aber einem Unternavigationssystem von S. Allerdings enthält auch jedes Unternavigationssystem wieder ein Hauptnavigationssystem. Betrachtet man von einem Unternavigationssystem das nächste Unternavigationssystem usw., so muss irgendwann ein Unternavigationssystem identisch zu einem Hauptnavigationssystem sein (d.h. es gibt keine Unterpunkte mehr). Entsprechend gibt es ein Hauptnavigationssystem S^H_{Ref}, in dem sich Γ_{Ref} befindet. Jedes Webdokument, das eine Navigation aller „darüberliegenden“ Hauptnavigationssysteme enthält, muss von einem <a href>-Pattern aus Γ_{Ref} verlinkt werden. Dies ergibt sich aus der Annahme, dass jede Navigation eines Navigationssystems (in Anlehnung an die Charakterisierung von Navigationen und Navigationssystemen) auf die Navigationen des Hauptnavigationssystems zeigt. Dies gilt dann rekursiv auch für jedes Hauptnavigationssystem eines Unternavigationssystems. Sind also alle Webdokumente der über S^H_{Ref} liegenden Hauptnavigationssysteme (sofern es sie gibt) und alle Webdokumente aus S^H_{Ref} selbst von <a href>-Pattern aus Γ_{Ref} verlinkt (was sie per Definition sein müssen), kann das gesamte Navigationssystem sukzessive rekonstruiert werden.

4.2 Harmoniegraph und harmonische Kombinationen

Bisher wissen wir jedoch nicht, ob Γ_{Ref} überhaupt eine Navigation darstellt und falls es so sein sollte, ebenfalls nicht, welche Gruppen dem Navigationssystem als Navigationen angehören. Aus diesem Grund betrachte ich als nächstes *harmonische Kombinationen*. Diese sollen Aufschluss darüber geben, welche Gruppen innerhalb der Gruppennachbarschaft im Vergleich mit Γ_{Ref} und untereinander das Reihenfolgekriterium erfüllen. Dies ist eine Voraussetzung für Navigationen eines Navigationssystems sowie für Außenseiter. Bevor ich eine harmonische Kombination defi-

nieren kann, ist hier zunächst die Definition einer *Kombination* gegeben.

Definition 16 (Kombination)
Sei Γ_{Ref} eine Gruppe, D_{Ref} die Menge der Nachbardokumente von Γ_{Ref}. Dann ist eine ***Kombination*** *für Γ_{Ref} eine Menge von Gruppen, die Γ_{Ref} und zusätzlich aus jedem Webdokument aus D_{Ref} genau eine Gruppe enthält.*

Jede Kombination kann damit bereits ein Navigationssystem oder einen Teil eines Navigationssystem darstellen. Würden wir aber jede mögliche Kombination auf die Verlinkungsstruktur testen, würde das aufgrund der möglicherweise hohen Anzahl der möglichen Kombinationen zu einem sehr hohen Aufwand führen. Aus diesem Grund führe ich zusätzlich *harmonische Kombinationen* ein.

Für die Bestimmung der harmonischen Kombination bediene ich mich der im Kapitel 3.2 eingeführten Definition der *x*-harmonierenden Gruppen. Innerhalb der Gruppennachbarschaft vergleiche ich jedes Paar von Gruppen unterschiedlicher Webdokumente und stelle in einem *Harmoniegraphen* durch eine Kante dar, welche Paare von Gruppen miteinander *v*-harmonieren.

Definition 17 (Harmoniegraph)
Sei Γ_{Ref} eine Gruppe, Ω die Gruppennachbarschaft von Γ_{Ref} und id_v eine v-ID-Zuordnung unter Ω. Dann ist der ***Harmoniegraph*** *von Γ_{Ref} der ungerichtete Graph $H_{Ref} = (\Omega, E)$, wobei für jedes ungeordnete Paar $(\Gamma_i, \Gamma_j) \in \Omega \times \Omega$, für das gilt, dass Γ_i und Γ_j aus verschiedenen Webdokumenten stammen und dass Γ_i und Γ_j unter Ω miteinander v-harmonieren, eine Kante zwischen Γ_i und Γ_j in E existiert.*

Ein Beispiel-Harmoniegraph ist in Abbildung 24 dargestellt. Dabei müssen die Kanten aus (a) und (b) gemeinsam betrachtet werden, da ich sie für einen besseren Überblick getrennt dargestellt habe.

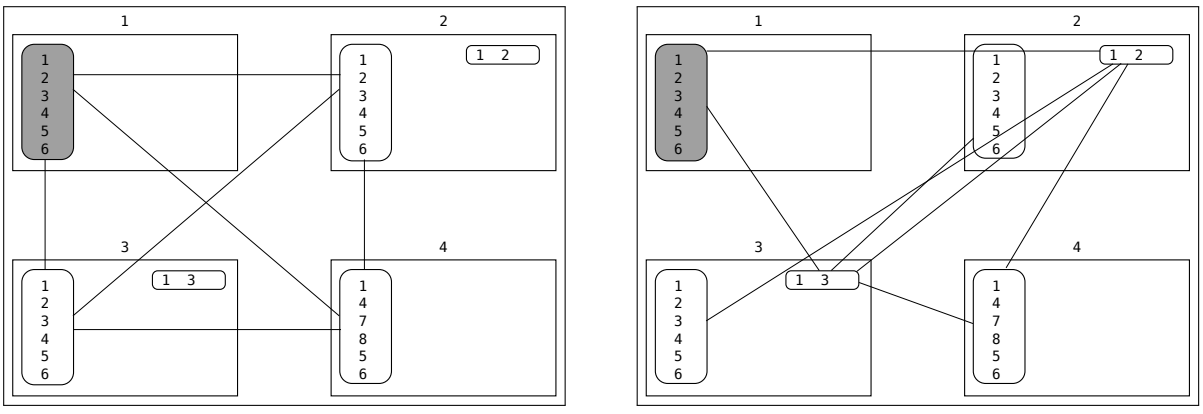


Abbildung 24: Vereint man alle Kanten aus (a) und (b), erhält man den Harmoniegraphen.

Komplexität der Bestimmung des Harmoniegraphen. Seien d_1 bis d_n die Nachbardokumente einer Referenzgruppe Γ_{Ref} und Ω_i die Gruppierung eines Webdokumentes d_i. Sei weiterhin $\Omega = \Omega_1 \cup \ldots \cup \Omega_n \cup \{\Gamma_{Ref}\}$ die Gruppennachbarschaft von Γ_{Ref}. Dann müssen alle Gruppen (außer Γ_{Ref} selbst) mit Γ_{Ref} verglichen werden. Dies entspricht $|\Omega| - 1$ Vergleichen. Zusätzlich muss jede Gruppe eines Webdokumentes mit jeder anderen Gruppe eines anderen Webdokumentes verglichen werden. Wurden bereits für ein Webdokument alle Harmonien bestimmt, muss keine Gruppe dieses Webdokumentes erneut mit einer anderen Gruppe eines anderen Webdokumentes verglichen werden, da Harmonien ungerichtet sind und die Gruppen entsprechend nicht erneut „aus der anderen Richtung“ überprüft werden müssen. So erfordert die Erstellung des Harmoniegraphen

$$|\Omega| - 1 + \sum_{j=1}^{n-1} (|\Omega_j| \cdot \sum_{i=j+1}^{n} |\Omega_i|)$$

Vergleiche zwischen einzelnen Gruppen. Jeder Vergleich einer Gruppe Γ_k^i mit einer Gruppe Γ_l^j nimmt dabei einen Berechnungsaufwand in $O(|\Gamma_k^i| \cdot |\Gamma_l^j|)$ in Anspruch (siehe Kapitel 3.2, Definition 12). Werden alle Gruppen einer Gruppierung $\Omega_i = \{\Gamma_1^i, \ldots, \Gamma_{|\Omega_i|}^i\}$ mit allen Gruppen der Gruppierung $\Omega_j = \{\Gamma_1^j, \ldots, \Gamma_{|\Omega_j|}^j\}$ verglichen, so liegt der Berechnungsaufwand in

$$O(\sum_{k=1}^{|\Omega_i|} \sum_{l=1}^{|\Omega_j|} |\Gamma_k^i| \cdot |\Gamma_l^j| \ ) = O((\sum_{k=1}^{|\Omega_i|} |\Gamma_k^i|) \cdot (\sum_{l=1}^{|\Omega_j|} |\Gamma_l^j|) \ ).$$

Etwas einfacher lässt sich dieser Aufwand beschreiben, wenn $m_i = \sum_{k=1}^{|\Omega_i|} |\Gamma_k^i|$ die Summe aller <a href>-Pattern beschreibt, die in der Gruppierung Ω_i auftreten und $m = m_1 + \ldots + m_n + |\Gamma_{Ref}|$ die Summe aller <a href>-Pattern innerhalb der Gruppennachbarschaft beschreibt. Dann liegt der Berechnungsaufwand für den Vergleich in $O(m_i \cdot m_j)$. Für die Berechnung aller Harmonien, die für den Harmoniegraphen von Bedeutung sind, liegt der Berechnungsaufwand entsprechend in

$$O(|\Gamma_{Ref}| \cdot \sum_{i=1}^{n} m_i + \sum_{j=1}^{n-1} \sum_{i=j}^{n} m_j \cdot m_i) \subseteq O(m^2).$$

Das heißt, der Berechnungsaufwand des Harmoniegraphen ist quadratisch in der Anzahl aller <a href>-Pattern, die sich innerhalb der Gruppennachbarschaft der Referenzgruppe befinden.

Mit Hilfe des Harmoniegraphen kann dann auch eine harmonische Kombination definiert werden.

Definition 18 (harmonische Kombination)
*Sei H_{Ref} der Harmoniegraph und D_{Ref} die Gruppennachbarschaft einer Gruppe Γ_{Ref}. Dann ist eine **harmonische Kombination** für Γ_{Ref} eine Kombinatione für Γ_{Ref}, dessen Gruppen einen vollständigen Subgraphen von H_{Ref} induzieren.*

Ein Beispiel für eine harmonische Kombination ist in Abbildung 24 (a) erkennbar. Jeweils eine Gruppe aus jedem Nachbardokument sowie die Referenzgruppe selbst, bilden einen vollständigen Subgraphen des Harmoniegraphen H_{Ref}. Dieser enthält bei n Nachbardokumenten $n+1$ Knoten.

Mit Hilfe der harmonischen Kombinationen stelle ich sicher, dass nur noch Gruppen betrachtet werden, die das Reihenfolgekriterium erfüllen. Jede Kombination, die keine harmonische Kombination ist, erfüllt dieses Kriterium nicht und kann aus diesem Grund aus weiteren Betrachtungen ausgeschlossen werden. Im ungünstigsten Fall sind alle Kombinationen harmonische Kombinationen, von denen keine die Verlinkungsstruktur aus Definition 14 aufweist. In diesem Fall muss jede Kombination bestimmt, auf ihre Eigenschaft als harmonische Kombination untersucht und schließlich ihre Verlinkungsstruktur überprüft werden. Dabei ist bereits das Auffinden aller harmonischen Kombinationen, also aller Cliquen der Größe $n+1$ aus H_{Nav}, die jeweils eine Gruppe aus jedem Nachbardokument der Referenzgruppe sowie die Referenzgruppe selbst enthalten, exponentiell in der Anzahl der Nachbardokumente von Γ_{Ref}.

Komplexität der Bestimmung aller harmonischen Kombinationen. Mit einer Brute-Force-Strategie können alle Kombinationen aufgezählt und jeweils darauf überprüft werden, ob sie eine Clique in H_{Nav} bilden. Dabei existieren $\Pi_{i=1}^{n}\Omega_i$ verschiedene Kombinationen. Die Überprüfung, ob eine Kombination eine Clique in H_{Nav} induziert, benötigt einen Berechnungsaufwand in $O(n^2)$, da jede Kante einmal überprüft werden muss. Entsprechend liegt der gesamte Berechnungsaufwand, um auf diese Weise alle Kombinationen zu ermitteln, die einen vollständigen Graphen in H_{Ref} induzieren, in $O(n^2 \cdot \Pi_{i=1}^{n}|\Omega_i|)$ und ist damit exponentiell in der Anzahl der Nachbardokumente von Γ_{Ref}. Vermutlich lässt sich der durch $O(n^2)$ abgeschätzte Aufwand für jede einzelne Überprüfung, ob eine gewählte Kombination eine Clique induziert, noch verringern, indem Ergebnisse vorher betrachteter Kombinationen berücksichtigt werden. Das größte Problem stellt aber die mit der Anzahl der Webdokumente der Gruppennachbarschaft exponentiell wachsende Menge der zu untersuchenden Kombinationen dar.

Eine Verringerung der zu betrachtenden Kombinationen ist in einigen Fällen möglich, indem alle Gruppen, die nicht mit der Referenzgruppe v-harmonieren aus der Gruppennachbarschaft entfernt werden, da diese Gruppen in keinem Fall einer harmonischen Kombination mit der Referenzgruppe angehören können. Harmonieren allerdings alle Gruppen der Gruppennachbarschaft mit der Referenzgruppe, wird durch diese Maßnahme keine Laufzeitverbesserung erzielt.
Ein weiterer Ansatz, um möglichst wenig Kombinationen betrachten zu müssen, ist, die Ähnlichkeiten zwischen der Referenzgruppe und den anderen Gruppen der Gruppennachbarschaft mit Hilfe der in Kapitel 3.2 eingeführten Verweis-, Bezeichner und Darstellungsäquivalenzen zu nutzen, um die Gruppen innerhalb der jeweiligen Gruppierungen zu sortieren. Dafür kann für jede Gruppe der Gruppennachbarschaft mit Γ_{Ref} ein Gewicht bestimmt werden, das die Summe der v-, b- und d-Harmoniegewichte darstellt. D.h. dass für jedes verweisäquivalente <a href>-Pattern ein Punkt, für bezeichneräquivalente <a href>-Pattern zwei Punkte und für darstellungsäquivalente <a href>-Pattern drei Punkte vergeben wird, da alle darstellungsäquivalenten <a href>-Pattern auch bezeichneräquivalent und alle bezeichneräquivalenten <a href>-Pattern auch verweisäquivalent sind. Die Gruppen innerhalb eines Webdokumentes können dann absteigend sortiert werden. Die erste Kombination, die dann auf die Eigenschaften einer harmonischen Kombination geprüft wird, ist dann die, die aus jedem Webdokument die am höchsten bewertete Gruppe enthält. Unter der Annahme, dass Navigationen desselben Navigationssystems eine hohe Ähnlichkeit zueinander aufweisen, kann dieses Vorgehen das Auffinden einer harmonischen

Kombination, die tatsächlich einem Navigationssystem angehört, stark beschleunigen. Handelt es sich bei der Referenzgruppe jedoch weder um eine Navigation noch einen Außenseiter, tritt kein beschleunigender Effekt ein, da dennoch alle Kombinationen überprüft werden müssen.

Ein dritter Ansatz, der sinnvoll sein kann, wenn die Anzahl der Webdokumente sehr groß ist, ist der Verzicht auf die Betrachtung einzelner Gruppen. Stattdessen könnte eine alternative Überprüfung der Verlinkungsstruktur für Mengen von Webdokumenten ab einer bestimmten Größe vorgesehen werden. Der Gedanke dahinter ist, dass mit einer steigenden Anzahl von zu untersuchenden Webdokumenten die Wahrscheinlichkeit sinkt, dass diese Menge von Webdokumenten zufällig (also ohne tatsächlich ein Navigationssystem zu enthalten) eine bestimmte Verlinkungsstruktur aufweist.

4.3 Navigation oder Außenseiter

Wurde eine harmonische Kombination gefunden, soll für diese Kombination die Verlinkungsstruktur überprüft werden. Ist die Referenzgruppe ein Außenseiter, so kann die von mir geforderte Verlinkungsstruktur nicht eingehalten werden. Der Grund ist, dass innerhalb eines Navigationsgraphen kein Knoten existieren darf, auf den nicht mindestens ein von ihm verschiedener Knoten zeigt. Aber genau diese Eigenschaft erfüllt jeder Knoten, der einen Außenseiter darstellt. Aus diesem Grund muss entschieden werden, ob die Verlinkungsstruktur für alle Gruppen der Kombination oder für alle außer der Referenzgruppe überprüft wird. Dies zu entscheiden, ist einfach. Jede Gruppe der Kombination (außer Γ_{Ref} selbst) muss darauf überprüft werden, ob sie ein <a href>-Pattern enthält, das auf Γ_{Ref} verweist. Ist dies in keiner Gruppe der Fall, wird die Verlinkungsstruktur der Gruppen der Kombination ohne Γ_{Ref} untersucht, da diese Eigenschaft auf Außenseiter, nicht aber auf Navigationen zutrifft. Andernfalls wird die Verlinkungsstruktur aller Gruppen der Kombination untersucht.

Wird die Verlinkungsstruktur der Kombination mit bzw. ohne der Referenzgruppe mit positivem Ergebnis überprüft, wird ausgegeben, dass die Referenzgruppe eine Navigation bzw. einen Außenseiter darstellt. Andernfalls wird eine neue, noch nicht betrachtete harmonische Kombination gewählt. Kann für Γ_{Ref} keine harmonische Kombination mehr gefunden werden, die mit bzw. ohne Γ_{Ref} die Verlinkungsstruktur erfüllt, stellt die Referenzgruppe weder eine Navigation noch einen Außenseiter unter den in dieser Arbeit vorgestellten Charakteristika dar.

Bis hier habe ich alle für die Navigationserkennung notwendigen Schritte beschrieben. Im folgenden Abschnitt fasse ich diese Schritte zusammen.

4.4 Überblick über die Schritte der Navigationserkennung

Zusammenfassend stelle ich in Abbildung 25 die wesentlichen Schritte der Navigationserkennung schematisch dar. Im Anschluss daran werde ich überprüfen, ob alle Eigenschaften, die ich in der Charakterisierung von Navigationen und Navigationssystemen in Kapitel 2.1 aufgeführt habe, durch diesen Algorithmus erfüllt werden. Abschließend die betrachte ich die Komplexität des dargestellten Algorithmus zur Navigationserkennung und den Einfluss einzelner Parameter auf die Laufzeit.

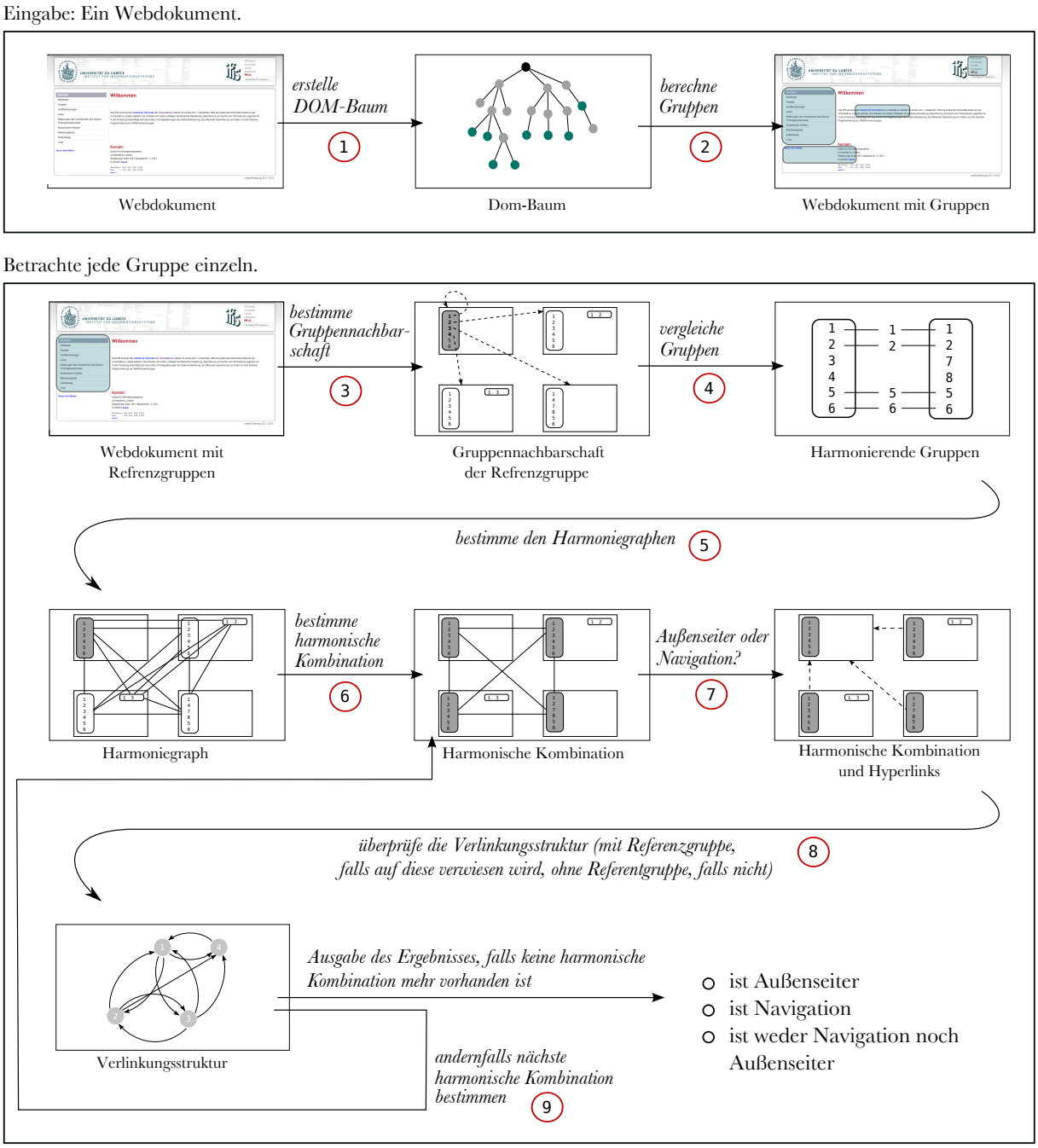


Abbildung 25: Darstellung des Algorithmus zur Navigationserkennung.

Anmerkungen zum Algorithmus: Alle Hyperlinks, die keiner Gruppe angehören, stellen auch keine Navigationslinks dar. Enthält die Gruppennachbarschaft nach Schritt (3) ausschließlich die zu betrachtende Gruppe, wird der Algorithmus für diese Gruppe sofort mit der Ausgabe „ist weder Navigation noch Außenseiter“ beendet. Dasselbe gilt, wenn innerhalb des Schrittes (6) keine harmonische Kombination gefunden werden kann. In diesem Fall ist repräsentieren alle <a href>-Pattern dieser Gruppe Inhaltslinks. Stellt die Gruppe einen Außenseiter oder eine Navigation dar, so werden alle enthaltenen Hyperlinks (auch Transverse) als Navigationslinks klassifiziert.

In Kapitel 2.1 habe ich ein Navigationssystem S als eine Menge von Navigationen N_i mit folgenden Eigenschaften beschrieben:

1. **Die <a href>-Pattern innerhalb einer Navigation N_i werden innerhalb des Quellcodes des Webdokumentes d_i, das N_i enthält, als Gruppe dargestellt.**

   Diese Eigenschaft wird erfüllt, da alle Schritte von (3) bis (9) nur für Gruppen durchgeführt werden.

2. **Die Ordnungen der <a href>-Pattern aller Paare $(N_i, N_j) \in S \times S$, die in N_i und N_j verweisäquivalent sind, soll identisch sein.**

   Diese Eigenschaft wird durch die Schritte (4) bis (6) sichergestellt.

3. **Jedes <a href>-Pattern einer Navigation, das einen Intrinsic darstellt, zeigt auf ein Navigationsdokument.**

   Diese Eigenschaft ergibt sich aus der Forderung, dass jedes Webdokument d_i, auf das innerhalb der Referenzgruppe Γ_{Ref} von einem Intrinsic verwiesen wird, eine Navigation enthalten muss. Andernfalls gäbe es innerhalb des Harmoniegraphen von Γ_{Ref} keine harmonische Kombination mit einer Gruppe aus d_i, die die Verlinkungsstruktur erfüllt. Dies ist aber die Voraussetzung dafür, dass die Refrenzgruppe als Navigation erkannt wird.

4. **Zwei verschiedene Navigationen $N_i, N_j \in S$ gehören verschiedenen Webdokumenten an.**

   Dies ist durch die Definition einer Kombination gegeben, die die Refrenzgruppe und aus jedem Nachbardokument der Referenzgruppe genau eine Gruppe enthalten muss.

5. **Ein Navigationssystem S erstreckt sich über eine Webdokumentenmenge D_S, die ausschließlich Webdokumente derselben Domain enthält und für die $|D_S| \geq 2$ gilt.**

   Dass ein Navigationssystem ausschließlich Webdokumente einer Domain enthalten kann, wird durch die Konstruktion der Gruppennachbarschaft sichergestellt, in die nur Webdokumente Einzug finden, die der Domain der betrachteten Referenzgruppe angehören. Auch dass sich S über mindestens zwei Webdokumente erstreckt, ist gegeben, da jedes dieser Webdokumente genau eine Navigation enthalten muss und eine Anforderung an die Gruppennachbarschaft ist, dass diese neben der Referenzgruppe mindestens eine weitere Navigation enthält.

6. **Die Verlinkungen von einer Navigation $N \in S$ auf Navigationsdokumente von S bilden einen *Navigationsgraphen*.**

   Diese Eigenschaft ist durch den Schritt (8) sichergestellt.

Für die Betrachtung der Komplexität des Algorithmus führe ich im Folgenden den Berechnungsaufwand für die einzelnen in Abbildung 25 mit einer rot eingekreisten Nummer versehenen Schritte des Algorithmus auf.

Der Berechnungsaufwand für die Erstellung des DOM-Baumes (1) eines Webdokumentes ist linear in der Größe des Quellcodes eines Webdokumentes, da jedes Einfügen eines Knotens in den DOM-Baum mit Beginn eines Starttags und jedes „Zurückwandern“ von einem Knoten zum Vaterknoten innerhalb des DOM-Baumes parallel zum Auftreten eines Endtags innerhalb des Quellcodes möglich ist. Anschließend folgt die Berechnung der Gruppen mit dem *erweiterten gruppiereKnoten*-Algorithmus (2). Hierfür ist ein Aufwand in $O(depth(T) \cdot |P|^2)$ erforderlich, wobei $|P|$ die Anzahl aller <a href>-Pattern innerhalb des Webdokumentes beschreibt und $depth(T)$ die maximale Tiefe des betrachteten DOM-Baumes. Nach der Berechnung der Gruppen des Webdokumentes, muss jede Gruppe einzeln darauf untersucht werden, ob sie eine Navigation, einen Außenseiter oder keins von beidem darstellt.

Wurde eine Referenzgruppe Γ_{Ref} gewählt, muss für die Bestimmung der Gruppennachbarschaft zunächst jedes Nachbardokument der Referenzgruppe bestimmt werden (3). Dafür müssen $|\Gamma_{Ref}|$ <a href>-Pattern betrachtet werden. Maximal können dann $|\Gamma_{Ref}|$ Webdokumente Nachbardokumente darstellen, für die dann jeweils die Schritte (1) und (2) erneut ausgeführt werden müssen, um die Gruppennachbarschaft zu bestimmen. Wurde die Gruppennachbarschaft bestimmt, muss für jeweils zwei Gruppen Γ_k^i und Γ_l^j überprüft werden, ob sie miteinander harmonieren (4). Diese Überprüfung ist mit Hilfe von Algorithmen zur Bestimmung der längsten Teilfolge möglich. Der Aufwand hierfür liegt in $O(|\Gamma_k^i| \cdot |\Gamma_l^j|)$. Um alle Harmonien zwischen Gruppen zu überprüfen, die für die Erstellung des Harmoniegraphen notwendig sind (5), ist ein quadratischer Aufwand in der Summe aller <a href>-Pattern der Gruppen innerhalb der Gruppennachbarschaft notwendig. Sei P_{sum} die Menge aller <a href>-Pattern der Referenzgruppe sowie der Nachbardokumente der Referenzgruppe. Dann befinden sich innerhalb der Gruppennachbarschaft maximal $|P_{sum}|$ <a href>-Pattern. Die Laufzeit für die Berechnung des Harmoniegraphen liegt dann in $O(|P_{sum}|^2)$.

Mithilfe des Harmoniegraphen müssen anschließend harmonische Kombinationen bestimmt werden (6). Dieser Aufwand ist exponentiell in der Anzahl der Nachbardokumente der Referenzgruppe und wächst gleichzeitig polynomiell mit der Anzahl der Gruppen innerhalb der einzelnen Nachbardokumente. Nachdem eine harmonische Kombination ermittelt wurde (sofern eine existiert), muss die Referenzgruppe als mögliche Navigation oder als möglicher Außenseiter klassifiziert werden (7). Die Komplexität für die Überprüfung, ob ein <a href>-Pattern einer Gruppe der betrachteten Kombination auf die Referenzgruppe zeigt, kann mit $O(|P_{sum}|)$ nach oben abgeschätzt werden, da maximal jedes <a href>-Pattern der Gruppennachbarschaft (außer Γ_{Ref} selbst) betrachtet werden muss. Dies ist dann der Fall, wenn jedes Webdokument genau eine Gruppe enthält und in dieser alle <a href>-Pattern des Webdokumentes vorhanden sind. Wurde entschieden, ob Γ_{Ref} als Navigation oder Außenseiter angenommen wird, wird die Verlinkungs-

struktur der Kombination mit bzw. ohne Γ_{Ref} mit Hilfe des Algorithmus *istNavigationsgraph* analysiert (8). Als Eingabe für diesen Algorithmus dient der gerichtete Graph $G = (V,E)$, dessen Knotenmenge V die zu prüfenden Gruppen enthält. In E existiert zwischen zwei Knoten, die zwei Gruppen Γ_k^i (aus dem Webdokument d_i) und Γ_l^j (aus dem Webdokument d_j) repräsentieren, genau eine Kante von Γ_k^i nach Γ_l^j, falls ein <a href>-Pattern in Γ_k^i existiert, das auf das Webdokument d_j zeigt. Der Algorithmus *istNavigationsgraph*(G) benötigt dabei eine Laufzeit in $O(|V| \cdot (|V| + |E|))$. Die Komplexität ist entsprechend kubisch in der Anzahl der Nachbardokumente und damit auch kubisch in der Mächtigkeit der Referenzgruppe. War die Überprüfung der Verlinkungsstruktur nicht erfolgreich, wird eine neue harmonische Kombination bestimmt (9). Je nachdem, ob eine harmonische Kombination, die die geforderte Verlinkungsstruktur aufweist, gefunden werden kann oder nicht, wird ausgegeben, dass die Referenzgruppe eine Navigation bzw. einen Außenseiter oder keins von beidem darstellt.

Insgesamt ist die Laufzeit für die Feststellung, ob eine Gruppe eine Navigation, einen Außerseiter oder keins von beidem darstellt, abhängig von verschiedenen Parametern:

- **Größe des Quellcodes**
  Der Aufwand für die Bestimmung des DOM-Baumes eines Webdokumentes d_i ist linear in der Größe des Quellcodes von d_i.
- **Tiefe des betrachteten DOM-Baumes:**
  Der Aufwand für die Bestimmung der Gruppen eines Webdokumentes d_i wächst linear mit der Tiefe des DOM-Baumes von d_i.
- **Anzahl der <a href>-Pattern innerhalb eines Webdokumentes:**
  Der Aufwand für die Bestimmung der Gruppen eines Webdokumentes d_i wächst quadratisch mit der Anzahl aller <a href>-Pattern aus d_i.
- **Anzahl der <a href>-Pattern innerhalb der Referenzgruppe:**
  Der Berechnungsaufwand für das Auffinden harmonischer Kombinationen innerhalb des Harmoniegraphen einer Referenzgruppe Γ_{Ref} wächst exponentiell mit der Anzahl der domaininternen <a href>-Pattern innerhalb von Γ_{Ref}, während der Berechnungsaufwand für die Überprüfung der Verlinkungsstruktur einer harmonischen Kombination kubisch mit der Anzahl der <a href>-Pattern innerhalb der Referenzgruppe wächst.
- **Anzahl der <a href>-Pattern innerhalb der Gruppennachbarschaft:**
  Der Berechnungsaufwand für das Erstellen des Harmoniegraphen einer Referenzgruppe Γ_{Ref} wächst quadratisch mit der Anzahl der <a href>-Pattern innerhalb der Gruppennachbarschaft von Γ_{Ref}.
- **Anzahl der Gruppen innerhalb eines Webdokumentes:**
  Der Berechnungsaufwand für das Auffinden harmonischer Kombinationen innerhalb des Harmoniegraphen einer Referenzgruppe Γ_{Ref} wächst polynomiell mit der durchschnittlichen Anzahl der Gruppen der Nachbardokumente von Γ_{Ref}.

5 Zusammenfassung und Ausblick

In dieser Arbeit habe ich mich mit der Erkennung von Navigationen befasst und dazu eine Vielzahl von Modellen und Algorithmen vorgestellt. Die Zielsetzung dazu war, auf Eingabe eines Webdokumentes zu bestimmen, welche Hyperlinks in diesem Webdokument einer Navigation angehören und welche nicht.

Zunächst habe ich in den Grundlagen Begriffe des World Wide Web beschrieben (wie Quellcode, Webdokument, Domain, Hyperlink) und habe <a href>-Pattern eingeführt, die Hyperlinks darstellen, die innerhalb eines (X)HTML-Quellcodes durch <a>-Elemente, die das Attribut *href* enthalten, umgesetzt werden. Anschließend habe ich Kalbachs[7] Definitionen des Begriffes Webnavigation beschrieben und aufbauend auf dieser Definition sowie eigenen Untersuchungen eine eigene Charakterisierung von Navigationen und Navigationssystemen vorgestellt. Als wesentliche Merkmale habe ich dabei unter anderem aufgeführt, dass eine Navigation eine jeweils in einem einzigen Webdokument auftretende Menge von <a href>-Pattern ist und ein Navigationssystem eine Menge von mindestens zwei Navigationen darstellt, wobei jede Navigation einem anderen Webdokument derselben Domain angehören muss. Ebenso sollen die <a href>-Pattern jeder Navigation in einer bestimmten Struktur im jeweiligen (X)HTML-Quellcode aufgeführt sein und die Verlinkungen der Navigationen eines Navigationssystems auf die Navigationsdokumente ebenfalls einer bestimmten Struktur folgen. Ebenso habe in die Bezeichnung Außenseiter eingeführt, wobei Außenseiter navigationsähnliche Mengen von <a href>-Pattern innerhalb eines Webdokumentes beschreiben. Das Webdokument, das den Außenseiter enthält, kann jedoch nicht über eine Navigation des Navigationssystems erreicht werden. Zusätzlich habe auf die Problematik der Formulierung einer geeigneten Charakterisierung von Navigationen hingewiesen, dass keine einheitlichen Standards zur Gestaltung von Navigationen innerhalb des World Wide Web existieren und es, selbst wenn es sie gäbe, es jedem Autor eines Webdokumentes selbst überlassen wäre, sie einzuhalten oder nicht. Dennoch habe ich Eigenschaften von Navigationen und Navigationssystemen beschrieben, die in meinen Untersuchen in vielen Fällen erfüllt wurden. Dazu zählen beispielsweise eine einheitliche Darstellung von Hyperlinks innerhalb eines Quellcodes, eine webdokumentübergreifende Ähnlichkeit zwischen Navigationen desselben Navigationssystems (Reihenfolgekriterium, dieselben Bezeichner, ähnliche Formatierung) und eine auffällige Verlinkungsstruktur zwischen Navigationen und Navigationsdokumenten. Abschließend habe ich in den Grundlagen zwei Anwendungsgebiete der Navigationserkennung, die Optimierung von Kleinbergs HITS-Algorithmus als Verfahren zum Ranking von Webdokumenten sowie die automatische Erstellung von Sitemaps, vorgestellt.

Einen Schwerpunkt dieser Arbeit bildet die Modellierung, Überprüfung und Analyse der Gruppen. Als Gruppen habe ich dabei Mengen von <a href>-Pattern innerhalb eines einzigen Webdokumentes bezeichnet, die einer bestimmten Struktur folgen. Diese Struktur habe ich definiert und beschrieben, wie mit linearem Aufwand in der Anzahl der im Webdokument vorhandenen <a href>-Pattern sowie in der Tiefe des (X)HTML-DOM-Baumes algorithmisch alle <a href>-Pattern ermittelt werden können, die dieser Struktur folgen. Ich habe auf mögliche Probleme hingewiesen und diese in einer Erweiterung des Algorithmus, der jedoch einen quadra-

tischen Aufwand in der Anzahl der <a href>-Pattern innerhalb des Webdokumentes erfordert, berücksichtigt. Anschließend habe ich verschiedene Abstufungen von Äquivalenzen zwischen <a href>-Pattern vorgestellt (Verweis-, Bezeichner-, Darstellungsäquivalenz) und beschrieben, wie aufbauend auf diesen Äquivalenzen Gruppen miteinander verglichen und als miteinander harmonierend angenommen werden können. Eine Harmonie besteht dann zwischen zwei Gruppen, wenn sie mindestens ein verweisäquivalentes <a href>-Pattern aufweisen und die Reihenfolge aller verweisäquivalenten <a href>-Pattern in beiden Gruppen dieselbe ist. Einen weiteren Schwerpunkt dieser Arbeit bildet die Definition, algorithmische Überprüfung und Analyse der Verlinkungsstruktur. Dafür habe ich unter anderem einen Hauptnavigationsgraphen, Navigationsgraphen und Navigationsebenen beschrieben und einen in der Anzahl der Knoten kubischen Algorithmus vorgestellt, mit dem überprüft werden kann, ob ein Graph einen Navigationsgraphen darstellt oder nicht.

Die Algorithmen zur Bestimmung von Gruppen, harmonierenden Gruppen sowie der Überprüfung der Verlinkungsstruktur habe ich abschließend in einer Beschreibung der Navigationserkennung genutzt. Entstanden ist ein Algorithmus mit exponentiellem Berechnungsaufwand in der Anzahl der <a href>-Pattern einer zu analysierenden Gruppe. Aber auch weitere Parameter von Webdokumenten, die ich ebenfalls vorgestellt habe, beeinflussen die Laufzeit. Dabei ist ein exponentieller Aufwand in der Praxis nur für kleine Probleminstanzen in akzeptabler Zeit umsetzbar.

Es bleibt zu prüfen, wie groß die einzelnen Parameter im World Wide Web überlicherweise werden, welchen positiven Einfluss die von mir vorgestellten Strategien für eine Verringerung des Berechnungsaufwandes in der Praxis aufweisen und welche weiteren Stategien existieren, um den Berechnungsaufwand zu minimieren. Ebenso sind in den von mir vorgestellten Algorithmen noch keine Toleranzen erlaubt. Für eine Anwendung in der Praxis wäre das Zulassen geringer Abweichungen von den geforderten Eigenschaften sinnvoll, um eine größere Menge von Navigationen innerhalb des World Wide Web zu erkennen und um auch auf diese Weise möglicherweise Laufzeitverbesserungen zu erzielen.

A Literaturverzeichnis

[1] Bernard, M.L., Developing Schemas for the Location of Common Web Objects. In Human Factors and Ergonomics Society Annual Meeting Proceedings Vol. 45, 2001, pp. 1161 – 1165

[2] Bharat, Henzinger, 1998. Improved algorithms for topic distillation in a hyperlinked environment. In Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (Melbourne, Australia, Aug. 24 –28). ACM, New York, pp. 104 –111.

[3] Brin S., Page, L., The Anatomy of a Large-Scale Hypertextual Web Search Engine, in the Proceedings of the 7th International World Wide Web Conference, Brisbane, Australia, 1998.

[4] Cai, D., He X., Wen, J., and Ma, W., Block-level link analysis. In ACM SIGIR Conference (SIGIR), 2004

[5] Cai, D., Yu S., Wen, J.-R. and Ma, W.-Y., Extracting content structure for web pages based on visual representation, Proceedings of the 5th Asia Pacific Web Conference, Springer, p. 406–417, 2003

[6] Hirschberg, D. S., A linear space algorithm for computing maximal common subsequences, In Communications of the ACM, 18(6), p. 341–343 1975

[7] Kalbach, J., Handbuch der Webnavigation, ISBN 9783897218659, Safari Books Online, O'Reilly, S. 4-5, 214ff, 2008

[8] Kleinberg, J., Authoritative sources in a hyperlinked environment. Journal of the ACM, 46(5), 1999.

[9] Li, Longzhuang and Shang, Yi and Zhang, Wei, Improvement of HITS-based Algorithms on Web Documents, Proceedings of the 11th international conference on World Wide Web, p. 527–535, 2002

[10] Morville, P., Rosenfeld, L., Information architecture for the World Wide Web, O'Reilly

[11] Shaikh, A.D.,Lenz, K., Where's the Search? Re-examining User Expectations of Web Objects, Usability News Vol. 8, 2006

[12] Tarjan, R., Depth-first search and linear graph algorithms. In SIAM Journal on Computing. Bd. 1, Nr. 2, S. 146-160., 1972

[13] Statistisches Bundesamt Deutschland – 73% der privaten Haushalte haben einen Internetzugang, *http://www.destatis.de/jetspeed/portal/cms/Sites/destatis/Internet/DE/Presse/pm/2009/12/PD09__464__IKT,templateId=renderPrint.psml*, letzter Zugriff: 22.03.2011

[14] JIM-STUDIE 2010, http://www.mpfs.de/fileadmin/JIM-pdf10/JIM2010.pdf, letzter Zugriff: 22.03.2011

[15] Ranking: Das sind die wertvollsten marken der Welt – Nachrichten Wirtschaft – WELT ONLINE, *http://www.welt.de/wirtschaft/article7373744/Das-sind-die-wertvollsten-Marken-der-Welt.html*, letzter Zugriff: Februar 2011

[16] The Wall Street Journal - Digital Network. Yahoo plans to shut AltaVista, Other Sites, http://online.wsj.com/article/SB10001424052748703395204576024024258782758.html, letzter Abruf: 17. Januar 2011

[17] Universität zu Lübeck – Wikipedia, *http://de.wikipedia.org/wiki/Universität_zu_Lübeck*, letzter Zugriff: März 2011

[18] Informatik: Universität zu Lübeck, *http://www.informatik.uni-luebeck.de/index.php?id=4*, letzter Zugriff: November 2010

[19] Informatik: Universität zu Lübeck, *http://www.informatik.uni-luebeck.de/index.php?id=1458*, letzter Zugriff: März 2011

[20] Startseite: IFIS Uni Lübeck,*http://www.ifis.uni-luebeck.de*, letzter Zugriff: November 2010

[21] Studentische Arbeiten: IFIS Uni Lübeck, *http://www.ifis.uni-luebeck.de/index.php?id=studentische-arbeiten*, November 2010

[22] Institut für Theoretische Informatik – Startseite, *http://www.iti.uni-luebeck.de*, letzter Zugriff: November 2010

[23] Institute of Computer Engineering – Homepage, *http://www.iti.uni-luebeck.de/index.php?id=23&L=1*, letzter Abruf: November 2010

[24] Universität: Universität zu Lübeck, *http://uniweb.uni-luebeck.de*, letzter Zugriff: November 2010

B Abbildungsverzeichnis

1 Ausschnitt des Webdokumentes[19] aus der Rubrik Informatik der Website der Universität zu Lübeck . . . 10
2 Ausschnitt des Webdokumentes [17] mit dem Wikipedia Artikel über die Universität zu Lübeck . . . 12
3 Linktypen . . . 12
4 Einteilung von Navigationseigenschaften . . . 13
5 DOM-Baum und Interpretation mit eingerahmten Gruppen der Startseite [20] des Instituts für Informationssysteme der Universität zu Lübeck . . . 14
6 Navigationen aus Sicht der Startseite [20] sowie aus Sicht des Webdokumentes [21] nach Verfolgen des Hyperlinks mit dem Bezeichner *Studentische Arbeiten.* . . . 15
7 Darstellung eines Webgraphen $G_{[S_\sigma]}$ und G_{σ}. . . . 17
8 Mutually Reinforcing Relationship . . . 19
9 Beispiel-DOM-Baum T . . . 24
10 Traversierung eines Baumes nach dem Verfahren der Tiefensuche . . . 28
11 Beispiel für eine Durchführung des *gruppiereKnoten*-Algorithmus . . . 30
12 Ausschnitt der vorläufigen neuen Startseite [24] der Universität zu Lübeck . . . 31
13 Ausschnitt eines Webdokumentes [18] der alten Informatik-Website der Universität zu Lübeck . . . 32
14 Worstcase-Szenario für den *erweiterten GruppiereKnoten-Algorithmus* . . . 33
15 XHTML-DOM-Baum der Startseite [20] des Instituts für Informationssysteme . . . 34
16 Screenshot der Startseite [20] des Instituts für Informationssysteme der Univerität zu Lübeck mit eingerahmten Gruppen. . . . 36
17 Beispiel für darstellungsäquivalente <a href>-Pattern . . . 39
18 Schematische Darstellung der Gruppen Γ_1 und Γ_2. . . . 41
19 Harmonierende Gruppen, teilweise mit unterschiedlichen möglichen Matchings . . . 42
20 Longest Common Subsequence für das Beispiel aus Abbildung 19 (d) und (e) . . . 43
21 Überprüfung eines Navigationsgraphen . . . 45
22 Ausschnitt der englischsprachigen Startseite [23] des Instituts für Technische Informatik der Universität zu Lübeck . . . 51
23 Nachbardokumente und Gruppennachbarschaft der Referenzgruppe (grau bzw. grün ausgefüllt). Die Webdokumente 5 und 6 (links) liegen außerhalb der Domain, alle anderen Webdokumente innerhalb derselben Domain. Gestrichelte Pfeile zeigen, auf welche Webdokumente die Referenzgruppe verweist. . . . 52
24 Vereint man alle Kanten aus (a) und (b), erhält man den Harmoniegraphen. . . . 53
25 Darstellung des Algorithmus zur Navigationserkennung. . . . 57

Ich versichere, die Bachelorarbeit selbständig und lediglich unter Benutzung der angegebenen Quellen und Hilfsmittel verfasst zu haben. Ich erkläre weiterhin, dass die vorliegende Arbeit noch nicht im Rahmen eines anderen Prüfungsverfahrens eingereicht wurde.

Lübeck, den 1. Mai 2011