# Datenstrukturen und Algorithmen zur verallgemeinerten Konstellationssuche auf der Basis von Objektrelationen

**Dissertation**

zur

Erlangung des Doktorgrades (Dr. rer. nat.)

der

Mathematisch-Naturwissenschaftlichen Fakultät

der

Rheinischen Friedrich-Wilhelms-Universität Bonn

vorgelegt von

Dirk Meyer

aus

Leverkusen

Bonn, Mai 2002

Angefertigt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Rheinischen Friedrich-Wilhelms-Universität Bonn

**Danksagung**

Als erstes möchte ich mich bei Herrn Prof. Dr. Clausen für die zahlreichen konstruktiven Diskussionen und der Betreuung meiner Dissertation bedanken. Ohne ihn wäre diese Arbeit nicht entstanden. Herrn Prof. Dr. Mathey gilt mein Dank für die Übernahme des Zweitgutachtens. Weiterhin möchte ich mich bei der gesamten Arbeitsgruppe von Herrn Prof. Dr. Clausen bedanken, insbesondere bei Vlora Arifi, Roland Engelbrecht, Sabine Fränzel, Heiko Goeman, Frank Kurth, Axel Mosig, Meinard Müller, Andreas Ribbrock und Klaus-Dieter Schneider. Schließlich gilt mein Dank allen Freunden, die mich in letzter Zeit geduldig motiviert haben.

# Inhaltsverzeichnis

# Kapitel 1

# Einleitung

Im DFG-Forschungsprojekt “MiDiLiB” wird die inhaltsbasierte Suche in digitalen Musikbibliotheken untersucht. Teil dieses Forschungsprojekts ist die Suche in klassischen Musikstücken, die in einer stark vereinfachten Partiturform vorliegen. Ein Benutzer gibt eine Notenfolge $Q$ als Anfrage vor und das Retrievalsystem soll alle Musikstücke der Datenbasis liefern, die $Q$ “enthalten”. In dem Forschungsprojekt wurde ein effizientes Retrievalsystem entwickelt, das detailliert in [4] beschrieben wird.

Um auch in großen, extern gelagerten Datenbeständen effizient suchen zu können, kommt in dem Retrievalsystem ein Index aus invertierten Listen zum Einsatz. Diese Indexvariante stammt aus der Volltextsuche und wurde an die Suche in Musikstücken angepaßt.

Das Konzept der Indexierung durch invertierte Listen wird allgemein in [5] beschrieben. Es basiert auf der Annahme, daß sich Konstellationen bzw. Anfragen als Ganzes modifizieren lassen. Die Menge der erlaubten Modifikationen wird durch eine mathematische Gruppe beschrieben. Jedes Teilobjekt einer Konstellation wird durch dasselbe Gruppenelement modifiziert.

In dieser Arbeit wird dieses Konzept verallgemeinert. Nicht mehr die Objekte an sich stehen im Mittelpunkt, sondern die Beziehungen *zwischen* den Objekten sind entscheidend. Dies führt zu einem verallgemeinerten Konstellationsbegriff, bei dem Konstellationen primär durch Objektrelationen beschrieben werden. Die Objektrelationen werden für eine Anwendung i.d.R. von einem Experten vorgegeben.

Als motivierende Anwendung wird in dieser Arbeit die Suche in Bauplänen verwendet. Eine typische Anfrage ist hier eine Konstellation aus Rohrleitungen. Dabei wird die Konstellation durch die Kreuzungen und Verästelungen der einzelnen Rohre untereinander beschrieben. Die Längen der Rohrleitungen spielen nur eine untergeordnete Rolle. Die verschiedenen Typen von Kreuzungen und Verästelungen zwischen Objekten sind Beispiele für Objektrelationen.

Für die Konstellationssuche werden verschiedene Trefferdefinitionen eingeführt. Hierdurch hat der Benutzer die Möglichkeit, die gesuchten Konstellationen näher im Kontext der bestehenden Objektbeziehungen zu beschreiben. Anstelle des Ausdrucks “Konstellationssuche” wird in dieser Arbeit oft der Begriff “Trefferberechnung” verwendet, da von einem Benutzer zusätzlich zu einer Anfragekonstellation eine Trefferdefinition ausgewählt wird.

Für die in dieser Arbeit betrachtete Konstellationssuche wird das Konzept der invertierten Listen weiter verallgemeinert.

Bei der allgemeinen Definition der Trefferbegriffe, Suchverfahren und -strukturen werden beschriftete Digraphen verwendet. Wie sich zeigen wird, lassen sich auf dieser Basis klassische Suchverfahren für die Trefferberechnung verwenden. Allerdings werden diese nicht den besonderen Anforderungen an eine Suche in großen Datenbeständen gerecht, da hier der Datentransfer zwischen Hauptspeicher und Massenspeicher eine entscheidende Rolle spielt.

Anhand einer prototypischen Implementierung der Suchverfahren wird demonstriert, daß sich durch die Verwendung der invertierten Listen im Vergleich zu klassischen Suchverfahren Laufzeitvorteile erzielen lassen.

**Relationale Datenbanken** In vielen praktischen Bereichen haben sich relationale Datenbanksysteme zur Speicherung von Informationen und zur Suche in Datenbeständen etabliert. Die Beschreibung von Objektbeziehungen durch Relationen legt die Verwendung von relationalen Datenbanken zur Konstellationssuche nahe. Dabei werden Relationen in Tabellen gespeichert. Die Suche nach einer angefragten Konstellation ergibt sich durch Verknüpfungen (Joins) der Tabellen.

Die Tabellen sind bei vielen Anwendungen sehr groß, da darin die Informationen über alle in der Datenbasis bestehenden Beziehungen zwischen Objekten gespeichert sind.

Bei typischen Anfragen werden für die Suche nur kleine Teile der Tabellen benötigt. Diese Teile lassen sich nicht im voraus bestimmen, sondern sind das Ergebnis der Verknüpfung von Tabellen.

Praktische Untersuchungen haben diese Aussagen bestätigt. Besteht eine Anfrage aus mehr als vier Objekten, waren die Datentenmengen für das Testsystem zu groß. Die Berechnungen wurden nach über 30 Minuten abgebrochen.

Aus diesem Grund scheiden relationale Datenbanksysteme bei der hier untersuchten Konstellationssuche aus, worauf in dieser Arbeit nicht weiter eingegangen wird.

**Aufbau der Arbeit** Die Arbeit ist so aufgebaut, daß als erstes ausführlich auf die verwendeten Trefferbegriffe eingegangen wird (Kapitel 2). Dabei erfolgt eine Darstellung der Trefferdefinitionen aus Anwendungen, bei denen ein Index aus invertierten Listen bereits zum Einsatz kommt (Abschnitt 2.1). Anschließend findet in Abschnitt 2.2 eine Vorstellung einer Architekturanwendung statt, an der die Unterschiede der alten und neu eingeführten Trefferbegriffe demonstriert werden.

Als nächstes folgt die genaue Definition der Trefferbegriffe. Wie sich im daran anschließenden Abschnitt herausstellt, lassen sich die Trefferberechnungen auch als spezielle Probleme aus der Graphentheorie darstellen. Hierbei werden Dokumente durch beschriftete Digraphen repräsentiert.

Schließlich werden die neuen Trefferbegriffe mit bereits bestehenden verglichen und weitere Anwendungsgebiete für die Konstellationssuche vorgestellt.

In Kapitel 3 erfolgt die Beschreibung der Trefferberechnung. Hier steht die formale Suche im Vordergrund. Zu Beginn des Kapitels wird auf die allgemeine Problematik eingegangen, die sich im Zusammenhang mit der Suche in großen, extern gelagerten Datenbeständen ergeben. In Abschnitt 3.2 findet die Vorstellung des Konzept der invertierten Listen statt. Die Beschreibung der formalen Trefferberechnung erfolgt in Abschnitt 3.3. Hierbei wird auch auf eine Erweiterung des Konzepts der invertierten Listen eingegangen, wie sie für die Trefferberechnung benötigt wird. Schließlich erfolgt im letzten Abschnitt des Kapitels die Besprechung von vier Verfahren zur Trefferberechnung. Alle vier basieren auf der Verwendung von invertierten Listen. Allerdings unterscheiden sich die Verfahren bei der Verwaltung von Zwischenergebnissen.

Auf die praktische Anwendung der Konstellationssuche wird in Kapitel 4 eingegangen. In den ersten beiden Abschnitten findet eine Beschreibung von Möglichkeiten der Optimierung im Zusammenhang mit den verwendeten invertierten Listen statt. Eine Vorstellung von praktischen Ergebnissen mit einer prototypischen Implementierung der Berechnungsverfahren erfolgt in Abschnitt 4.3. Hierbei werden die einzelnen Verfahren untereinander verglichen und Abhängigkeiten der Laufzeiten untersucht.

Den Abschluß dieser Arbeit bildet eine Zusammenfassung sowie ein Ausblick.

# Kapitel 2

# Treffer

In diesem Kapitel wird das allgemeine Umfeld der Konstellationssuche erläutert. Kern ist dabei die Einführung von vier neuen Trefferarten.

Das Kapitel ist so aufgebaut, daß erst auf klassische Anwendungen im Umfeld einer Suche in Datenbanken eingegangen wird. Im zweiten Abschnitt wird ein konkreter Anwendungsbereich vorgestellt. Dieser dient als Motivation und Demonstration für die im darauffolgenden Abschnitt definierten neuen Trefferbegriffe. Als nächstes wird die Trefferberechnung mit bekannten Problemen aus der Graphentheorie verglichen. Die neuen Trefferbegriffe werden anschließen Trefferbegriffen aus anderen Bereichen gegenübergestellt. Abgeschlossen wird dieses Kapitel mit einer kurzen Beschreibung von Anwendungsgebieten, in denen die Trefferbegriffe und Treffersuche eingesetzt werden können.

## 2.1 Einführung

Die Suche nach bestimmten Dingen begegnet uns in vielen Situationen. Beispielsweise bei der Suche nach einem bestimmten Buch im Bücherregal, nach bestimmten Wörtern in Dateien oder nach bestimmten Bildern in Bilddatenbanken.

Wichtig ist in all diesen Situationen, daß man eine Vorstellung von dem besitzt, was man eigentlich sucht. Man hat also eine (mehr oder weniger) genaue Beschreibung von Eigenschaften des Gesuchten, des "Treffers", im Kopf. Oft ist diese Beschreibung nicht eindeutig oder unvollständig. Somit liefert die Suche typischerweise eine Reihe von "Kandidaten", aus denen, etwa durch detaillierte Betrachtung der Kandidaten, einer oder mehrere als das Gesuchte ausgewählt werden. Wichtig sind bei der Suche mittels Computer die Eingabe- bzw. Beschreibungsmöglichkeiten, um einen Suchbegriff (bzw. die Treffer) möglichst genau spezifizieren zu können. Natürlich sollten alle angegebenen Merkmale des Gesuchten bei einer automatischen Suche berücksichtigt werden.

Eine einfache Form der Suche, manuell oder maschinell, ist das Nachschlagen in einem Wörterbuch. Ist man z.B. an einer Übersetzung des englischen Wortes "right" interessiert, findet man in einem Englisch-Deutsch Wörterbuch mehrere Übersetzungsmöglichkeiten: right = rechts, right = richtig, right = rechte Seite, right = aufrichten. Jeder dieser Übersetzungen ist ein Ergebnis der Suche, ein **Treffer**. Betrachtet man alle einzelnen Ergebnisse zusammen, so bezeichnet man die Menge aller Ergebnisse als **Treffermenge**. Bei der Suche nach Übersetzungen des Wortes "right" besteht die Treffermenge aus den 4 angegebenen Übersetzungen (Treffern).

Besitzt man etwa bei der oben erwähnten Suche nach dem Wort "right" die Zusatzinformation, daß es sich dabei um ein Verb handelt, besitzt die Suche ein eindeutiges Ergebnis: die Treffermenge besteht aus einem Element, dem Treffer "right = aufrichten".

Schon dieses einfache Beispiel verdeutlicht, wie wichtig die Angabe von Eigenschaften ist, die das Gesuchte (ein Treffer) erfüllen muß. Eine zu unspezifische Charakterisierung der Suchkriterien kann zu einer großen Treffermenge führen, auch wenn der Benutzer nur nach einem bestimmten Treffer sucht.

Die Suche einzelner Wörter in Büchern (oder Textdatenbanken) ist eine relativ einfache Form der Suche. Auch die Suche nach Seiten in einem Buch bzw. nach Dokumenten in einer Textdatenbank, die mehrere gegebene Wörter enthalten, gestaltet sich einfach [50]. Komplexer wird die Suche, wenn zusätzlich zu den gegebenen Wörtern verlangt wird, daß sie in einer bestimmten Reihenfolge stehen, z.B. die gesuchten Wörter sollen hintereinander stehen.

Die Suche von Wörtern in einer Sammlung von Textdokumenten wird Volltextsuche genannt. In [50] werden Verfahren zur Volltextsuche beschrieben, die auch für große Textsammlungen im Gigabytebereich und darüber effizient anwendbar sind. Ein dort aufgeführte Suchstrategie, die Suche mittels invertierter Listen (inverted file index), bildet die Basis für die in meiner Arbeit behandelte Konstellationssuche. Zur Vorbereitung auf die Konstellationssuche, bzw. auf die Definition von Treffern, werden als nächstes einige Anwendungsformen der Suche im multimedialen Umfeld angesprochen.

### 2.1.1 Datenbasis, Abstraktion und gängige Trefferdefinitionen

Dem Leser mag folgende Situation bekannt vorkommen: Man hat irgendwo ein Fragment eines Liedes aufgeschnappt und möchte Titel und Interpreten des Liedes erfahren.

Im Forschungsbereich "Music Information Retrieval" [41] wird u.a. solchen Fragestellungen nachgegangen: Finde in einer Datenbank von Musikstücken diejenigen, die ein gegebenes Musikfragment enthalten. Wir gehen hier vereinfachend davon aus, daß die Anfrage und alle Musikstücke in der Datenbank in Partiturform gegeben sind. Ein Treffer ist dann ein Musikstück, das die gesuchte Notenfolge zu einem beliebigen Zeitpunkt enthält. Gesucht ist natürlich die Treffermenge, d.h. alle Musikstücke der Datenbank, die die Anfrage enthalten. Bevor auf eine formale Definition von Treffern eingegangen wird, werden kurz einige Begriffe eingeführt, die die weitere Diskussion erleichtern.

**Datenbasis, Dokument, Objekt** Betrachten wir folgende Art der Volltextsuche im Detail: Finde in einem Buch die Buchseiten, die alle Wörter einer Anfrage enthalten.

Für die Suche bzw. für ein Suchverfahren existieren zwei unterschiedliche Sichtweisen: eine reale und eine abstrakte. Ein reales Buch besteht aus (realen) Buchseiten, wobei jede (reale) Buchseite als reales Dokument aufgefaßt wird. Dabei ist egal, ob ein Buch bzw. eine Buchseite in Papierform, als TEX-Dokument oder in eingescannter Form vorliegt.

Jede Buchseite besitzt eine eindeutige Seitennummer. Eine Seitennummer ist i.d.R. eine arabische oder, z.B. bei einem Vorwort, eine römische Zahl. Diese Durchnummerierung (Referenzierung) läßt sich durch eine endliche Indexmenge $I$ erfassen, wobei für eine Seitennummer $i \in I$ des Buches gilt, daß $i$ eine römische oder arabische Zahl ist.

Allgemein gilt: Eine reale Datenbasis $\mathcal{D}'$ ist eine Folge von realen Dokumenten $D'_i$, d.h.$\mathcal{D}' = (D'_i)_{i \in I}$. Auf das Beispiel bezogen ist eine reale Datenbasis ein Buch bestehend aus einer Folge von realen Dokumenten: den Seiten eines Buches.

Findet die Suche in einer Sammlung von Büchern statt, ist ein (reales) Dokument $D'_i$ weiterhin eine Buchseite. Allerdings ist die Indexmenge $I$ anders definiert: für ein $i \in I$ ist $i =$ (ISBN, Seitennummer), d.h. mittels einer Kombination aus ISBN und Seitennummer wird eine Buchseite eindeutig referenziert.

Das Ergebnis einer Anfrage ist eine Teilmenge von $I$, d.h. Referenzen auf die realen Dokumente, die die Anfrage enthalten.

Die Daten bzw. Informationen, auf denen ein Suchalgorithmus aufbaut, basieren auf einer abstrakten Sichtweise. Diese entspricht einer Modellierung der realen Anschauung. Durch die abstrakte Darstellungsform werden die für die Suche relevanten Informationen in einer für den Computer "verträglichen" Datenstruktur dargestellt. Für die Volltextsuche in einem Buch gilt: Eine reale Buchseite wird abstrakt als Menge von Wörtern dargestellt, d.h. ein abstraktes Dokument ist eine Menge von Wörtern. Durch diesen Abstraktionsschritt gehen viele Eigenschaften einer realen Buchseite verloren: u.a. die Reihenfolge der Wörter, ob Wörter mehrfach auf einer Seite vorkommen und die Formatierung der Seite. Da es aber bei der hier verwendeten Variante der

Volltextsuche nur darauf ankommt, welche Wörter auf der Seite vorkommen, enthält die abstrakte Darstellungsform alle nötigen Informationen.

Im folgenden wird die reale Sichtweise vernachlässigt, d.h. wenn von einem Dokument die Rede ist, ist damit ein abstraktes Dokument bzw. eine abstrakte Darstellung eines realen Dokuments gemeint.

Dadurch ergibt sich für eine Anwendung folgendes Szenario: Sei $\mathcal{O}$ eine Menge von Objekten. Diese kann in der Theorie unendlich sein. Ein Dokument $D$ ist eine endliche Menge von Objekten, $D \subseteq \mathcal{O}$. Bzgl. einer Indexmenge $I$ ist eine Datenbasis $\mathcal{D}$ eine Folge von Dokumenten, d.h. $\mathcal{D} = (D_i)_{i \in I}$. Die Indexmenge $I$ stellt den Zusammenhang zwischen realen und abstrakten Dokumenten her: Für jedes reale Dokument $D'_i$ ist $D_i$ das entsprechende abstrakte Dokument.

Eine Anfrage $Q$ besteht, wie ein Dokument $D$, aus einer endlichen Menge von Objekten, d.h. $Q \subseteq \mathcal{O}$ mit $|Q| < \infty$.

Für die Volltextsuche ergibt sich $\mathcal{O}$ als die Menge aller Wörter und für eine Anfrage $Q$ ergibt sich der folgende Trefferbegriff:

$$i \text{ ist Treffer bzgl. } Q \Leftrightarrow Q \subseteq D_i. \tag{2.1}$$

Mit Hilfe der Referenz $i$ kann der Benutzer die realen Dokumente $D'_i$ betrachten. Auf eine Visualisierung der Treffer in realen Dokumenten wird nicht weiter eingegangen.

Als nächstes wird die Suche in Musikstücken kurz diskutiert im Hinblick auf den Abstraktionsschritt, die Datenbasis sowie die Dokumente und Objekte.

Geht man davon aus, daß alle (realen) Musikstücke in Partiturform vorliegen, ist die Darstellung im MIDI-Format [35] ein erster Abstraktionsschritt. Wird jede Note nur durch ihre Tonhöhe $p$ und Einsatzzeit $t$ beschrieben, erhält man eine noch abstraktere Darstellungsform. Hierfür ergibt sich folgendes Szenario: Objekte sind abstrakte Noten $(p, t)$, d.h. $\mathcal{O} = [0 : 127] \times \mathbb{N}$, wobei als Tonhöhenvorrat der MIDI-Bereich $[0 : 127]$ verwendet wird.

Als Indexmenge $I$ bietet sich eine Kombination aus Komponistenname und Opus an, d.h. $i = (k, o) \in I$ für Komponist $k$ und Opus $o$.

Dokumente $D$ und Anfragen $Q$ bilden endliche Teilmengen von $\mathcal{O}$. Eine Datenbasis $\mathcal{D} = (D_i)_{i \in I}$ ist eine Folge von Dokumenten.

Für eine Anfrage $Q$ werden alle Musikstücke $D$ aus der Datenbasis gesucht, bei denen $Q$ zu einem beliebigen Zeitpunkt vorkommt. O.B.d.A. nehmen wir an, daß die erste Note einer Anfrage die Einsatzzeit 0 besitzt. Wenn man die komplette Anfrage $Q$ um $Z$ Zeiteinheiten verschiebt, muß jede einzelne Note $(p, t)$ der Anfrage, um $Z$ verschoben, in einem Trefferdokument $D \in \mathcal{D}$ enthalten sein, d.h. $(p, t + Z) \in D$. Damit ergibt sich für diese Anwendung folgende Definition von Treffern bzgl. einer Anfrage $Q$:

$$i \in I \text{ ist Treffer bzgl. } Q \;\Leftrightarrow\; \exists Z \in \mathbb{N} : \{(p, t + Z) \mid (p, t) \in Q\} \subseteq D_i. \tag{2.2}$$

**Beispiel 2.1.1.** In Abbildung 2.1 ist eine Anfrage $Q$ und eine Musikstück $D$ dargestellt. Verschiebt man die Anfrage um das Zeitintervall von fünf Viertelnoten, so stimmt die Anfrage mit dem kompletten 2. Takt überein. ◦

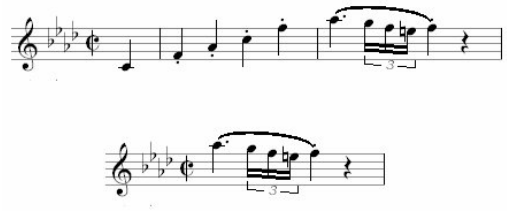


Abbildung 2.1: Oben der Beginn von Beethovens Klaviersonate Opus 2, Nr. 1, unten eine Anfrage.

Natürlich könnte man den Trefferbegriff noch erweitern, indem man zusätzlich zur zeitlichen Verschiebung auch transponierte Vorkommen der Anfrage erlaubt. Diese erlaubten "Operationen", etwa zeitliche Verschiebung und/oder Tonhöhentransponierung, lassen sich in vielen Fällen durch das in Abschnitt 2.1.2 beschriebene allgemeinere Konzept darstellen.

Allgemein formuliert findet eine Suche in einer **Datenbasis** (Datenbank) $\mathcal{D}$ statt. Eine Datenbasis ist eine endliche Folge von **Dokumenten**, d.h. $\mathcal{D} = (D_i)_{i \in I}$ für eine Indexmenge $I$. Ein Dokument der Datenbasis ist eine endliche Menge von **Objekten**. Die Menge aller möglichen Objekte wird mit $\mathcal{O}$ bezeichnet. Eine Anfrage $Q$ besteht, wie ein Dokument $D$, aus einer endlichen Menge von Objekten, d.h. $Q \subseteq \mathcal{O}$, $D \subseteq \mathcal{O}$.

Im folgenden wird eine Indexmenge $I$ einer Datenbasis mit $N$ Dokumenten als Menge $\{1, \ldots, N\}$ betrachtet und somit eine Datenbasis verkürzt durch $\mathcal{D} = (D_1, \ldots, D_N)$ notiert.

Umgangsprachlich ist ein Treffer bezüglich einer Anfrage $Q$ ein Dokument $D$ der Datenbasis, das $Q$ "enthält", wobei das Enthaltensein (Vorkommen) anwendungsspezifisch definiert wird.

**Granularität einer Datenbasis** Bei der Volltextsuche wurde als Dokument die Abstraktion einer Seite eines Buches angenommen. Treffer sind dann die Dokumente bzw. Buchseiten, die alle Wörter einer Anfrage enthalten. Bei einer Sammlung von Büchern kann ein gesamtes Buch als Dokument angesehen werden, d.h. hier wird nicht in einzelne Buchseiten unterteilt. Eine Indexmenge würde dann nur aus ISBN Einträgen bestehen. Im Vergleich zur vorherigen Sichtweise, daß ein Dokument eine Buchseite ist, liegt jetzt eine gröbere Granularität vor, da ein Dokument jetzt ein ganzes Buch ist. Eine feinere Granularität wäre etwa die Modellierung jeder einzelnen Zeile einer Buchseite als Dokument.

Bei der Suche in Musikstücken ist der Sachverhalt ähnlich. Komplette Musikstücke können z.B. in einzelne Sätze unterteilt werden. Faßt man einen einzelnen Satz als ein Dokument auf, erhält man eine feinere Granularität als wenn man jedes komplette Musikstück als Dokument ansieht.

Es bleibt also der Anwendung überlassen, welche Granularität für den Aufbau einer (abstrakten) Datenbasis gewählt wird.

### 2.1.2 Mustersuche aus gruppentheoretischer Sicht ($G$-Treffer)

Die bisher beschriebenen Beispiele lassen sich alle durch ein allgemeines Konzept erfassen, das in [5] beschrieben ist. Dieses Konzept basiert auf Gruppen. Wie sich später herausstellen wird, ist dieses Konzept ein Sonderfall der in dieser Arbeit vorgestellten Konzepte. Im folgenden wird das Konzept kurz vorgestellt. Für eine ausführliche Besprechung sei auf [5] verwiesen.

Sei $\mathcal{O}$ eine beliebige Menge von Datenobjekten. Z.B. kann man sich im Kontext von Musikdatenbanken unter $\mathcal{O}$ die Menge aller Noten vorstellen. Wie im letzten Abschnitt eingeführt ist ein Dokument $D$ eine Teilmenge von $\mathcal{O}$ und eine Datenbasis eine Folge $D_1, \ldots, D_N$ von Dokumenten. Eine Anfrage $Q$ ist, wie ein Dokument, eine Teilmenge von $\mathcal{O}$, also auch ein Dokument, allerdings typischerweise mit geringerer Kardinalität als die Dokumente der Datenbasis.

Treffer wurden bei der Suche in Textdokumenten einfach als diejenigen Dokumente $D$ der Datenbasis definiert, für die $Q$ eine Teilmenge darstellt, d.h. $Q \subseteq D$. Bei der Suche in Musikdatenbanken ist der Trefferbegriff etwas komplizierter: Eine Anfrage $Q$ darf in zeitlich verschobener Form in einem Dokument vorkommen. Diese Form von Verschiebungen von $Q$ lassen sich durch Gruppenoperationen ausdrücken.

Sei eine Gruppe $G$ gegeben. Eine Menge $\mathcal{O}$ wird zu einer **G-Menge** vermöge einer Abbildung $G \times \mathcal{O} \to \mathcal{O}$, $(g, m) \mapsto gm$, wenn $(gh)m = g(hm)$ und $1m = m$ für alle $g, h \in G, m \in M$ erfüllt sind. Man sagt auch: $G$ operiert auf $\mathcal{O}$. Dies induziert $G$-Operationen auf der Potenzmenge $2^{\mathcal{O}}$ von $\mathcal{O}$: $gT := \{gt | t \in T\}$ für $T \in 2^{\mathcal{O}}$.

Als **G-Treffer** wird ein Dokument $D$ mit $gQ \subseteq D$ bezeichnet. Beispielsweise lassen sich die oben erwähnten zeitlichen Translationen durch die Gruppe $(\mathbb{Z}, +)$ realisieren mit $gm = g(p, t) := (p, t + g)$.

Eine $G$-Operation auf $\mathcal{O}$ induziert eine Äquivalenzrelation auf $\mathcal{O}$. Zwei Elemente $m, m' \in \mathcal{O}$ heißen $G$-äquivalent, wenn ein $g \in G$ mit $m' = gm$ existiert. Die $G$-Äquivalenzklassen sind die

$G$-Bahnen, die mit $Gm := \{gm|g \in G\}$ bezeichnet werden. Für ein Repräsentantensystem $R$ der $G$-Bahnen gilt: $M = \sqcup_{r \in R} Gr$.

Für eine $G$-Menge $\mathcal{O}$ wird $G_m := \{g \in G|gm = m\}$ als Stabilisator von $m$ bezeichnet, der eine Untergruppe von $G$ darstellt.

Die Berechnung aller $G$-Treffer bzgl. einer Anfrage $Q$ und einer Datenbasis $\mathcal{D} = (D_1 \dots D_N)$ gestaltet sich wie folgt: Sei $G_{\mathcal{D}}(Q) := \{(g,i)|g \in G, i \in [1:N], gQ \subseteq D_i\}$. Für ein $m \in \mathcal{O}$ bezeichnet $G_{\mathcal{D}}(m) := \{(g,i)|g \in G, i \in [1:N], gm \in D_i\}$ die **G-invertierte Liste** von $m$. Damit ergibt sich $G_{\mathcal{D}}(Q) = \cap_{m \in Q} G_{\mathcal{D}}(m)$. Für ein Repräsentantensystem $R$ der $G$-Bahnen von $\mathcal{O}$ läßt sich jedes $m \in \mathcal{O}$ durch $m = g_m r_m$ darstellen mit $g_m \in G$ und $r_m \in R$ und es gilt

$$G_{\mathcal{D}}(Q) = \cap_{m \in Q} G_{\mathcal{D}}(r_m) g_m^{-1}. \tag{2.3}$$

Man sieht, daß die invertierten Listen unabhängig von einer Anfrage sind und somit vorab berechnet werden können. Diese Art von Index wird auch als "inverted file index" bezeichnet.

Für die Suche in Musikstücken ist $R = \{(p,0)|p \in [0:127]\}$ ein mögliches Repräsentantensystem. In $R$ ist also für jede Tonhöhe genau eine Note enthalten. Jede Note $(p,t)$ läßt sich mit $R$ durch $t(p,0)$ ausdrücken, wobei $t \in G$. Die invertierte Liste $G_{\mathcal{D}}(m)$ für eine Note $m = (p,t)$ besteht aus allen Noten mit Tonhöhe $t$. Mit obiger Formel (2.3) lassen sich so alle Treffer berechnen.

### 2.1.3 Unscharfe Suche

Bei den bisher besprochenen Anwendungen mußte eine Anfrage "exakt" in einem Dokument vorkommen. Bei der Volltextsuche in Büchern wurden nur Seiten als Treffer geliefert, die ein gesuchtes Wort exakt enthalten; abweichende Formen (Fälle), etwa Plural, wurden nicht als Treffer bezeichnet. Bei der Suche in Musikstücken mußten Tonhöhen und Einsatzzeiten übereinstimmen. Gerade die Angabe von exakten (absoluten) Tonhöhen bereitet vielen Anwendern große Schwierigkeiten, wodurch man diesen Trefferbegriff nicht gerade als anwenderfreundlich bezeichnen kann.

Eine Erweiterung der exakten Suche ist die unscharfe Suche. So würde man bei der Volltextsuche etwa Pluralformen etc. oder Tippfehler erlauben. Bei der Suche in Musikstücken könnte der Benutzer angeben, daß eine bestimmte Anzahl von Tonhöhen (oder Einsatzzeiten) nur ungefähr, etwa im Abstand von einem Ganztonschritt, übereinstimmen müssen.

Eine Variante der unscharfen Suche ist die, daß der Benutzer jedes Objekt seiner Anfrage nicht exakt festlegt, sondern mögliche Alternativen angibt. Bei einer Suche in Musikstücken etwa durch die Angabe eines Bereiches oder einer Menge von Tonhöhen für eine Note der Anfrage, bei der Volltextsuche z.B. "Fahrzeug" als Alternative für "Auto".

Eine Anfrage $\hat{Q}$ besteht dann nicht mehr aus einzelnen Objekten, sondern aus nichtleeren *Mengen* von Objekten. Für $\hat{Q} = \{\{\text{Fahrzeug, Auto}\}, \{\text{Straße, Weg}\}\}$ werden bei einer Volltextsuche alle Dokumente als Treffer berechnet, die das Wort "Fahrzeug" oder "Auto" und zusätzlich das Wort "Straße" oder "Weg" enthalten. Damit ist eine Anfrage $\hat{Q}$ eine Menge von Objektmengen $\hat{q}$. Jedes $\hat{q} \in Q$, $\hat{q} = \{q_1, \dots q_l\}$, besteht aus mindestens einem Objekt $q_1$ und möglicherweise aus $l-1$ zusätzlichen Alternativen $q_2, \dots q_l$. Ein Treffer "enthält" pro Objektmenge $\hat{q} \in Q$ ein Objekt $q' \in \hat{q}$. Für die unscharfe Suche nach einer Anfrage $\hat{Q} = \{\hat{q_1}, \dots, \hat{q_m}\}, \hat{q_i} \subseteq \mathcal{O}$ gilt:

$$\begin{aligned} D \in \mathcal{D} \text{ ist Treffer bzgl. } \hat{Q} \Leftrightarrow \quad & \exists Q := \{q_1, \dots, q_m\} : \\ & q_i \in \hat{q_i}, 1 \le i \le m \wedge \\ & Q \text{ ist ein Treffer bzgl. der exakten Suche.} \end{aligned} \tag{2.4}$$

Bei dieser Definition wird deutlich, daß sich Treffer bzgl. einer unscharfen Suche auf die Suche nach exakten Treffer reduzieren läßt. In [5] wird detaillierter auf eine unscharfe Suche eingegangen und gezeigt, wie sich diese in das in Abschnitt 2.1.2 beschriebene Konzept integrieren läßt.

Eine andere Form der unscharfen Suche wird durch die Angabe eines Distanzmaßes möglich. Zum Beispiel läßt sich mit einem Distanzmaß auf Notenfolgen die Ähnlichkeit bzw. Unähnlichkeit zwischen zwei Melodien oder Melodiefragmenten messen. Ein gängiges Distanzmaß wird z.B. in [34]

beschrieben. Im Gegensatz zur exakten Suche werden hier Noten einer Notenfolge im Kontext ihrer benachbarten Noten analysiert. Ein Distanzmaß $\delta$ liefert für zwei Notenfolgen einen Wert, durch den die Unähnlichkeit dieser beiden Notenfolgen ausgedrückt wird. Für zwei gleiche Notenfolgen sollte das Distanzmaß den Wert 0 liefern. Je höher der Distanzwert für zwei Notenfolgen, um so unähnlicher sind diese (bzgl. dieses Distanzmaßes).

Allgemein gilt: Gegeben sei ein Distanzmaß $\delta$, das für zwei Objektmengen $A, B$ eine Unähnlichkeit (Distanz) dieser Objektmengen berechnet. Ein Distanzmaß ist also eine Abbildung $\delta : 2^{\mathcal{O}} \times 2^{\mathcal{O}} \to \mathbb{N}$. In der Mathematik wird für ein Distanzmaß gefordert, daß $\delta(x, y) = 0 \Leftrightarrow x = y$, $\delta(y, x) = d(x, y)$ für alle $x, y$ aus $2^{\mathcal{O}}$ und die Dreiecksungleichung gilt. Für die weitere Besprechung haben diese Eigenschaften allerdings keine Bedeutung.

In Abhängigkeit von einem Distanzmaß $\delta$ gibt ein Benutzer zusätzlich zur Anfrage $Q$ einen Schwellwert $\beta$ an. Gesucht sind dann in der Datenbank $\mathcal{D}$ alle Dokumente $D$, die einen Ausschnitt $T_D \subseteq D$ besitzen, dessen Distanzwert zu $Q$ unterhalb des Schwellwertes $\beta$ liegt, d.h. $\delta(T_D, Q) \leq \beta$. Somit ergibt sich bzgl. Anfrage $Q$, Distanzmaß $\delta$ und Schwellwert $\beta$ folgender Trefferbegriff:

$$D \in \mathcal{D} \text{ ist Treffer } \Leftrightarrow \exists T \subseteq D : \delta(T, Q) \leq \beta. \tag{2.5}$$

### 2.1.4 Fazit

Im Hinblick auf die inhaltliche Suche in Dokumenten wurde auf die Unterscheidung in die reale und abstrakte Sichtweise eingegangen. Dabei werden zur Suche Abstraktionen von realen Dokumenten als Datenbasis verwendet, d.h. die Datenbasis wird als Folge von (abstrakten) Dokumenten angesehen. Hierfür dient eine Indexmenge $I$ als Referenzierung, d.h. für jedes $i \in I$ existiert sowohl ein reales Dokument $D'_i$ als auch das entsprechende abstrakte Dokument $D_i$. Dabei werden reale Dokumente nur noch zur späteren Visualisierung der Treffer einer Anfrage benötigt und bleiben im folgenden unberücksichtigt. Ein Treffer bzgl. einer Anfrage $Q$ wird hier als Referenz $i \in I$ aufgefaßt. Die Treffermenge ist somit eine Teilmenge von $I$.

Für den weiteren Verlauf dieser Arbeit wird die Bezeichnung “Dokument” für ein abstraktes Dokument verwendet. Weiterhin wird auch das zu einem Treffer $i \in I$ gehörige Dokument $D_i$ als Treffer bezeichnet.

Alle bisher aufgeführten Trefferbegriffe der verschiedenen Anwendungsgebiete sind ähnlich aufgebaut. Sie sind alle in dem in Abschnitt 2.1.2 besprochenen Konzept enthalten. Eine Anfrage wird auf der Basis einer Gruppenoperation komplett durch ein Gruppenelement modifiziert. Die so modifizierte Anfrage muß in einem Trefferdokument enthalten sein.

Die unscharfe Suche erlaubt eine gewisse Fehlertoleranz zwischen Anfrage und Treffern. Gerade von benutzerfreundlichen Systemen wird diese verlangt.

## 2.2 Architekturanwendung

In diesem Abschnitt wird eine Anwendung aus dem Architekturbereich vorgestellt, die sich mit den bisher vorgestellten Trefferbegriffen nicht erfassen läßt. Hier werden erweiterte bzw. verallgemeinerte Trefferbegriffe benötigt. Nach der Vorstellung des Architekturanwendungsgebietes in Abschnitt 2.2.1 werden im darauffolgenden Abschnitt Trefferbegriffe für diese Anwendung eingeführt, die in Abschnitt 2.2.3 an Beispielen diskutiert werden.

Beim Entwurf neuer Gebäude greift ein Architekt oft auf brauchbare Teilentwürfe aus alten Plänen zurück. Eine Automatisierung der zeitaufwendigen manuellen Suche in alten Bauplänen ist wünschenswert. Beschränken wir uns auf bestimmte Gebäudekomponenten, etwa Strom- und Wasserleitungen sowie Zu- und Abluftkanäle, so lassen sich diese in abstrakter Form als Strecken darstellen. Betrachtet man jede Etage als eigenständigen, zweidimensionalen Bauplan, so lassen sich Leitungen und Kanäle als Strecken in der Ebene modellieren. Diese Form der Abstraktion wird im objektorientierten Installationsmodell “ARMILLA” [19] und “A4” [22] verwendet. In diesen Modellen wird zusätzlich davon ausgegangen, daß alle Strecken (als abstrakte Darstellungsform von Leitungen und Kanälen) parallel zu einer den beiden Achsen der Ebene verlaufen. Diese

starke Einschränkung ermöglicht kosten- und zeitoptimierte Entwurfs- und Verwaltungsphasen von hochinstallierten Gebäuden [22].

In diesem Szenario besteht die Datenbasis $\mathcal{D}$ aus einer Folge von Gebäudebauplänen. Ein Dokument, ein Bauplan einer Etage, besteht aus einer Menge von Strecken. Ein Objekt ist eine achsenparallele Strecke in der Ebene $\mathbb{Z} \times \mathbb{Z}$, wobei eine Strecke durch ihre beiden Endpunkte definiert ist. Als Menge aller Objekte $\mathcal{O}$ ergibt sich somit

$$\mathcal{O} := \{(x_1, y_1, x_2, y_2) \in \mathbb{Z}^4 \mid (x_1 < x_2 \wedge y_1 = y_2) \vee (x_1 = x_2 \wedge y_1 < y_2)\}.$$

Eine Strecke $(x_1, y_1, x_2, y_2) \in \mathcal{O}$ mit $y_1 = y_2$ wird waagerechte Strecke oder Objekt mit **waagerechter Ausrichtung** genannt. Alle anderen Strecken werden als senkrechte Strecken oder Objekte mit **senkrechter Ausrichtung** bezeichnet. Für sie gilt $x_1 = x_2$. Eine Anfrage $Q$ an eine Datenbasis aus Bauplänen ist, wie ein Dokument, eine Menge von Strecken, d.h. $Q \subseteq \mathcal{O}$.

Hier wird wieder die in Abschnitt 2.1.1 besprochene Unterscheidung in reale und abstrakte Dokumente deutlich. Während ein realer Bauplan aus sehr komplexen Objekten besteht, die neben den dreidimensionalen geometrischen Daten auch semantische Informationen enthalten, werden alle Objekte abstrakt als Strecken dargestellt. Standardmäßig ist die Granularität einer Datenbasis so definiert, daß ein Bauplan in mehrere abstrakte Dokumente unterteilt wird, wobei jede Etage des Plans einem Dokument entspricht. Weiterhin findet eine Fokussierung auf bestimmte reale Objekte statt [36]. Dies bedeutet, daß zum Zeitpunkt der Erstellung der abstrakten Datenbasis nicht alle realen Objekte übernommen werden. Es wird z.B. eine eigene Datenbasis für Stromleitungen und eine andere für Belüftungsschächte erstellt. Auf die Fokussierung wird nicht weiter eingegangen. Sie wird z.B. in [36] ausführlich untersucht. Bei der im weiteren verwendeten Definition der Objekte unterscheiden sich diese allein durch ihre geometrischen Ausprägungen in der Ebene.

### 2.2.1 Treffermotivation

Im letzten Abschnitt wurde auf die Definition der Datenbasis im Architekturbereich eingegangen. Darauf basierend wird in diesem Abschnitt die Suche in einer Datenbasis aus abstrakten Bauplänen besprochen. Dabei gestaltet sich die Beschreibung von Treffern komplexer als in den Anwendungen aus Abschnitt 2.1. Nun steht die Anordnung der Objekte zueinander im Vordergrund. Im folgenden wird darauf eingegangen, wie sich solche Konstellationen aus Objekten beschreiben lassen. Hierbei spielt die Lage eines Objektes *relativ* zu anderen Objekten, unter Berücksichtigung bestimmter Zusatzbedingungen, eine zentrale Rolle.

Im Forschungsprojekt FABEL [47] wurden Verfahren des fallbasierten Schließens [27] für eine abstrahierte Architekturdomäne untersucht. Bestandteil des Projektes war die Auswahl von "typischen" Fällen: Ausschnitte aus Bauplänen, denen Experten eine besondere Bedeutung zuordnen. Diese Auswahl bildet die sogenannte Fallbasis für das fallbasierte Schließen. Die Evaluierung der Baupläne durch Experten brachte einige Erkenntnisse zum Vorschein, die auch für eine "Volltextsuche" in Bauplänen wichtig sind. Ein Auszug aus dem ermittelten Katalog sind folgende Eigenschaften [37]:

1. Typische Anfragen sind Konstellationen von Leitungen oder Kanälen. Innerhalb einer solchen Konstellation kann davon ausgegangen werden, daß jedes Objekt mit mindestens einem anderen Objekt "verbunden" ist (z.B. bei Verästelungen von Leitungen).

2. Für den Vergleich von Konstellationen (Objektmengen) bzw. bei der Suche nach "ähnlichen" Konstellationen sind weder die genaue absolute Position noch die exakte Größe der Objekte wichtig.

3. Eine Konstellation wird hauptsächlich durch die zwischen den Objekten auftretenden Verbindungen (Kreuzungen) charakterisiert. Verbindungen treten zwischen Objekten mit unterschiedlicher Ausrichtung (waagerecht + senkrecht) auf und werden in drei **Kreuzungskategorien** unterteilt: +-Kreuzungen, $\top$-Kreuzungen und $L$-Kreuzungen.

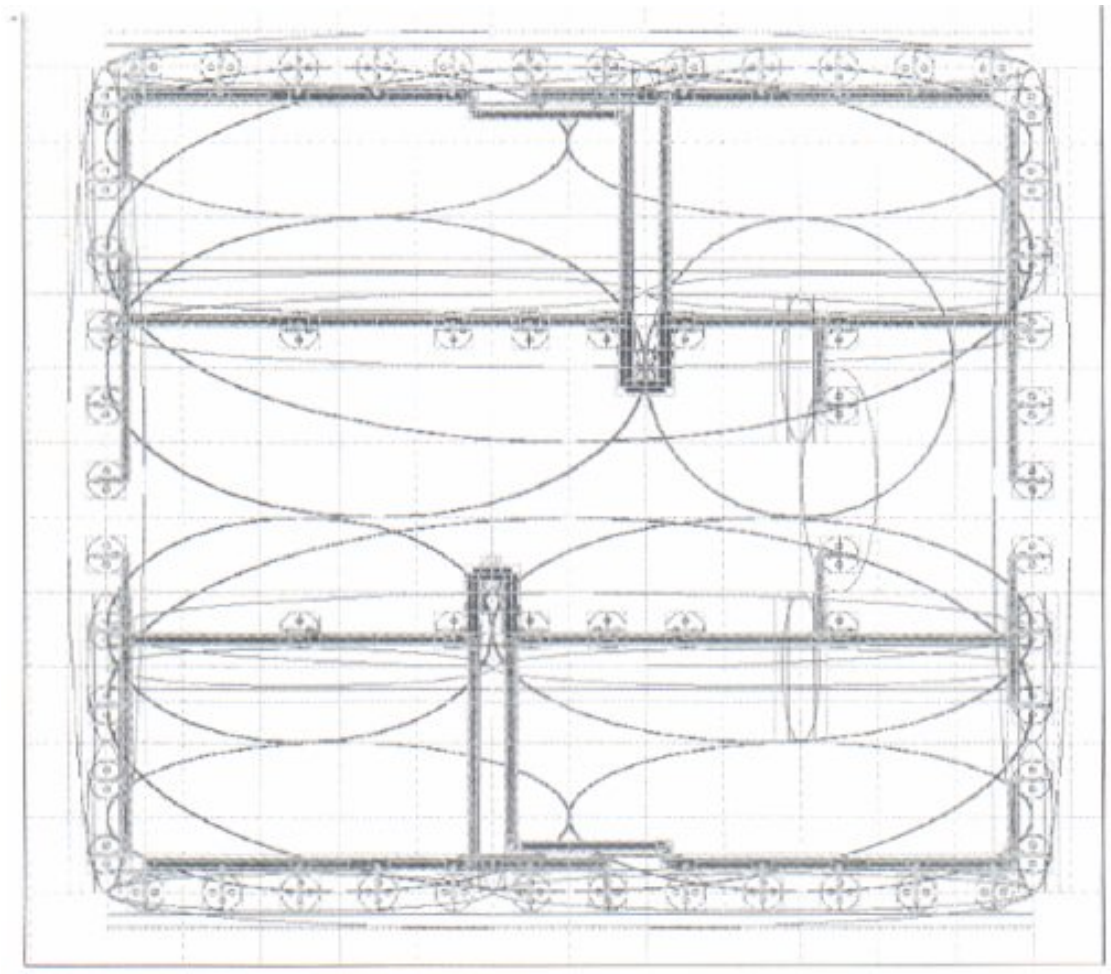


Abbildung 2.2: Eine Etage eines Bauplans im A4-System (aus [22]). Dicke Linien stellen Belüftungsschächte dar. Sie bilden die für die Suche verwendeten abstrakten Objekte.

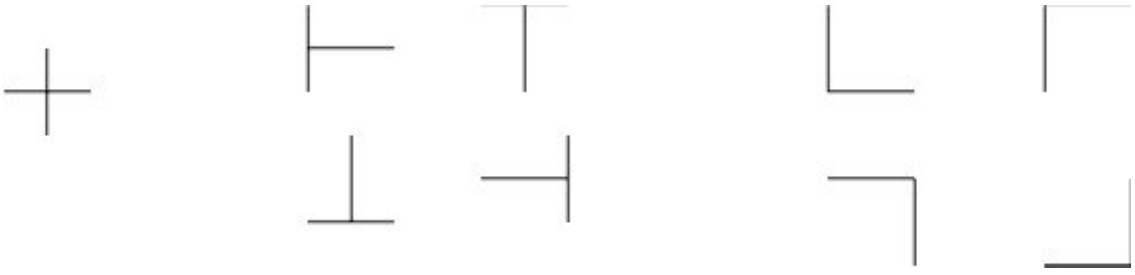


Abbildung 2.3: Die 9 Kreuzungstypen: links der +-Kreuzungstyp, in der Mitte die $\top$-Kreuzungstypen und rechts die $L$-Kreuzungstypen.

Aus den im letzten Punkt aufgeführten Kreuzungskategorien lassen sich insgesamt 9 **Kreuzungstypen** ableiten, die in Abbildung 2.3 dargestellt sind. Innerhalb des Forschungsprojekts entstanden einige Arbeiten, die diese Kreuzungstypen verwenden (z.B. [32] und [6]). Dabei stand mehr die Diskussion der Beschreibungsmöglichkeiten durch diesen Ansatz im Vordergrund als effiziente Verfahren. Weiterhin basieren die Arbeiten auf dem Vergleich zweier Pläne, wobei sich die verwendeten Verfahren nicht für die Anwendung auf große Dokumentsammlungen eignen. Dagegen steht nun die Effizienz im Vordergrund: Es soll in einer Art "Volltextsuche" in großen Datenbanken gesucht werden. Bisher wurde davon ausgegangen, daß die Datenbasis in den Hauptspeicher paßt. Dagegen basiert meine Arbeit auf der Annahme, daß die Dokumente extern gespeichert sind und der wahlfreie Zugriff auf extern gelagerte Daten als sehr zeitintensiv angesehen wird. Zwar wurden in diesem Kontext Ansätze verfolgt, Fälle mittels sehr rudimentärer Heuristiken zu indexieren. Diese Ansätze haben sich allerdings als nicht praktikabel erwiesen: Die verwendeten Heuristiken waren viel zu grob und es wurden zu viele "Treffer" ermittelt, die nicht die gewünschten strukturellen bzw. anwendungsspezifischen Anforderungen erfüllen [18].

Die oben aus dem Katalog aufgeführten Eigenschaften stellen die Basis für die folgenden Trefferbegriffe dar. Viele der hier nicht aufgeführten Eigenschaften des Katalogs lassen sich durch Vorverarbeitungsschritte realisieren (siehe [32] oder [36]), etwa die Kombination von Objekten gleicher Ausrichtung zu einem großen Objekt oder die Gruppierung von Objekten ohne Ausrichtung in regelmäßige Muster. Die kombinierten Objekte können bei einer späteren Suche mitberücksichtigt werden.

Die für dieses Anwendungsszenario wichtigen Kreuzungstypen lassen sich wie folgt formal darstellen: Jede Kreuzung (Verbindung) tritt immer zwischen genau zwei Objekten auf. Damit lassen sich die 9 Kreuzungstypen durch 9 zweistellige Relation auf $\mathcal{O}$ darstellen. Dabei stehen zwei Objekte in Relation zueinander, wenn sie sich kreuzen. Somit ergeben sich folgende 9 Relationen als Realisierung der 9 Kreuzungstypen:

1. $R_{+} := \{\big((x_1, y_1, x_2, y_2), (x'_1, y'_1, x'_2, y'_2)\big) \in \mathcal{O} \times \mathcal{O} \mid x_1 < x'_1 < x_2 \wedge y'_1 < y_1 < y'_2\}$
2. $R_{\top} := \{\big((x_1, y_1, x_2, y_2), (x'_1, y'_1, x'_2, y'_2)\big) \in \mathcal{O} \times \mathcal{O} \mid x_1 < x'_1 = x'_2 < x_2 \wedge y'_2 = y_1 = y_2\}$
3. $R_{\dashv} := \{\big((x_1, y_1, x_2, y_2), (x'_1, y'_1, x'_2, y'_2)\big) \in \mathcal{O} \times \mathcal{O} \mid y'_1 < y_1 = y_2 < y'_2 \wedge x_2 = x'_1 = x'_2\}$
4. $R_{\perp} := \{\big((x_1, y_1, x_2, y_2), (x'_1, y'_1, x'_2, y'_2)\big) \in \mathcal{O} \times \mathcal{O} \mid x_1 < x'_1 = x'_2 < x_2 \wedge y'_1 = y_1 = y_2\}$
5. $R_{\vdash} := \{\big((x_1, y_1, x_2, y_2), (x'_1, y'_1, x'_2, y'_2)\big) \in \mathcal{O} \times \mathcal{O} \mid y'_1 < y_1 = y_2 < y'_2 \wedge x_1 = x'_1 = x'_2\}$
6. $R_{\lfloor} := \{\big((x_1, y_1, x_2, y_2), (x'_1, y'_1, x'_2, y'_2)\big) \in \mathcal{O} \times \mathcal{O} \mid x_1 = x'_1 = x'_2 \wedge y'_1 = y_1 = y_2\}$
7. $R_{\lceil} := \{\big((x_1, y_1, x_2, y_2), (x'_1, y'_1, x'_2, y'_2)\big) \in \mathcal{O} \times \mathcal{O} \mid x_1 = x'_1 = x'_2 \wedge y'_2 = y_1 = y_2\}$
8. $R_{\rceil} := \{\big((x_1, y_1, x_2, y_2), (x'_1, y'_1, x'_2, y'_2)\big) \in \mathcal{O} \times \mathcal{O} \mid x_2 = x'_1 = x'_2 \wedge y'_2 = y_1 = y_2\}$
9. $R_{\rfloor} := \{\big((x_1, y_1, x_2, y_2), (x'_1, y'_1, x'_2, y'_2)\big) \in \mathcal{O} \times \mathcal{O} \mid x_2 = x'_1 = x'_2 \wedge y'_1 = y_1 = y_2\}$

Die Menge aller Relationen einer Anwendung wird mit $\mathcal{R}$ bezeichnet. Für die hier besprochene Anwendung in der Architekturdomäne ergibt sich als Menge aller Relationen

$$\mathcal{R}_{Kreuzungen} := \{R_{+}, R_{\top}, R_{\dashv}, R_{\perp}, R_{\vdash}, R_{\lfloor}, R_{\lceil}, R_{\rceil}, R_{\rfloor}\}.$$

Alle Relationen $R \in \mathcal{R}_{Kreuzungen}$ sind offensichtlich antireflexiv ($\forall (o, o') \in R$ gilt $o \neq o'$), antisymmetrisch und nicht transitiv. Natürlich hätten alle Relationen auch symmetrisch definiert werden können. In Hinsicht auf eine praktische Anwendung ist dies allerdings nicht geschehen. Vielmehr sind die Relationen so definiert, daß für jede Relation $R \in \mathcal{R}_{Kreuzungen}$ aus $(o, o') \in R$ folgt, daß $o$ eine waagerechte und $o'$ eine senkrechte Ausrichtung besitzt. Für die Theorie ist diese Wahl willkürlich, in der Praxis kann diese Wahl zu unterschiedlichem Laufzeitverhalten und Speicherplatzbedürfnissen führen.

### 2.2.2 Trefferbegriffe

Während im FABEL-Projekt komplette Fälle miteinander verglichen wurden [47], soll bei der als nächstes beschriebenen "Volltextsuche" nach Plänen gesucht werden, die eine Anfrage "enthalten". Um diese Anwendung auch für große Datenmengen effizient zu gestalten, werden die oben aufgeführten Eigenschaften, dargestellt durch Relationen, ausgenutzt. Dieses Ausnutzen von bestimmten Eigenschaften oder Beschreibungsmerkmalen ist nicht auf die Architekturdomäne beschränkt. Allgemeine Definitionen der Trefferbegriffe für (fast) beliebige Anwendungsdomänen werden in Abschnitt 2.3 angegeben. Doch kommen wir nun zu einem ersten Trefferbegriff für die "Volltextsuche" in Bauplänen. Weitere, ähnlich aufgebaute Trefferdefinitionen werden folgen.

Sei $Q$ eine Anfrage mit $m$ Objekten, $\mathcal{D}$ eine Datenbasis aus Bauplänen und $\mathcal{R}_{Kreuzungen}$ die Menge von (Kreuzungs-) Relationen, die die 9 Kreuzungstypen repräsentieren. Eine Anforderung an die Anfrage ist die, daß jedes Objekt der Anfrage mit mindestens einem anderen Objekt der Anfrage verbunden ist:

$$\forall q \in Q \; \exists q' \in Q, R \in \mathcal{R}_{Kreuzungen} : (q, q') \in R \vee (q', q) \in R. \tag{2.6}$$

Somit werden einzelne Objekte ohne Anbindung an andere Objekte als Teil einer Anfrage ausgeschlossen. Dies ist für die Architekturanwendung konsistent mit der Expertenbeobachtung "jedes Objekte ist mit mindestens einem anderen verbunden".

Für eine Anfrage $Q$ werden die Dokumente $D$ der Datenbasis gesucht, die eine Teilmenge $T \subseteq D$ enthalten, wo

1. die Anzahl der Objekte übereinstimmen, d.h. $|T| = |Q|$ und
2. wenn zwei Objekte $q, q'$ der Anfrage mit dem Kreuzungstyp $\gamma$ verbunden sind, d.h. $(q, q') \in R_\gamma$, dann sollen in $T$ ebenfalls zwei "entsprechende" Objekte $o, o'$ existieren, die mit dem Kreuzungstyp $\gamma$ verbunden sind, d.h. $(o, o') \in R_\gamma$.

Formal läßt sich dieser Sachverhalt durch eine injektive Abbildung $\Psi : Q \to D$ modellieren, wobei $T = Bild(\Psi)$ ist und für ein Objekt $q$ der Anfrage das Objekt $\Psi(q)$ als das $q$ "entsprechende" Objekt in $T$ angesehen werden kann.

Somit ergibt sich folgende Trefferdefinition: Ein Dokument $D \in \mathcal{D}$ ist genau dann ein Treffer, wenn eine injektive Abbildung $\Psi : Q \to D$, $\Psi$ existiert mit:

$$\forall R \in \mathcal{R}_{Kreuzungen} \forall q, q' \in Q : (q, q') \in R \Rightarrow \big(\Psi(q), \Psi(q')\big) \in R. \tag{2.7}$$

Durch die Injektivität der sogenannten **Trefferabbildung** $\Psi$ ist $|Q| = |Bild(\Psi)|$ sichergestellt. Der 2. Punkt der oben aufgeführten Suchanforderungen wird durch die an die Trefferabbildung gestellte Zusatzbedingung eingehalten: Die Verbindungen zwischen Anfrageobjekten liegen auch im Dokumentteil $T := Bild(\Psi)$ zwischen "entsprechenden" Objekten vor.

Diese Trefferdefinition ist vollkommen unabhängig von der absoluten Lage und Größe einzelner Objekte sowie der *exakten* relativen Lage der Objekte untereinander. Vielmehr müssen die Kreuzungstypen zwischen den Objekten übereinstimmen, wodurch die grobe relative Lage der Objekte zueinander gewährleistet wird. Damit sind Treffer skalierungsinvariant bzgl. einer Anfrage, insofern die Relationen (Verbindungen) zwischen den Objekten bestehenbleiben.

**Beispiel 2.2.1.** In Abbildung 2.4 ist eine Anfrage $Q$ und ein Dokument $D$ angegeben. Durch folgende Tabelle wird eine injektive Abbildung, eine Trefferabbildung $\Psi : Q \to D$, nach Gleichung (2.7) definiert:

| $q$ | $\Psi(q)$ |
|---|---|
| a | t |
| b | u |
| c | v |
| d | w |
| e | x |

Der Leser möge sich davon überzeugen, daß dies die einzige Trefferabbildung nach Gleichung (2.7) ist. ◦

Die oben angegebene Trefferdefinition ist sehr allgemein gehalten. Unberücksichtigt bleibt u.a. bei der Definition, ob zusätzlich zu den durch eine Anfrage bzw. Trefferabbildung geforderten Relationen noch weitere Relationen zwischen den "entsprechenden" Dokumentobjekten bestehen dürfen: Gilt ein Dokument $D$ mit Trefferabbildung $\Psi : Q \rightarrow D$ noch als Treffer, wenn Objekte $o, o' \in Bild(\Psi)$ existieren, so daß $(o, o') \in R$ und $\left(\Psi^{-1}(o), \Psi^{-1}(o')\right) \notin R$ für eine Relation $R \in \mathcal{R}$?[1] Da diese Fragestellung stark von einer konkreten Anfrage abhängt, werden dem Benutzer nicht nur eine, sondern mehrere verschiedene Trefferdefinitionen zur Verfügung gestellt, aus denen er sich zum Zeitpunkt seiner Anfrage eine auswählen kann.

Für den Fall, daß im Bildbereich einer Trefferabbildung $\Psi$ keine weiteren Relationen als den geforderten auftreten dürfen, ändert sich die Zusatzbedingung an $\Psi$ unter (2.7) in:

$$\forall R \in \mathcal{R}_{Kreuzungen} \forall q, q' \in Q \text{ gilt } (q, q') \in R \Leftrightarrow \left(\Psi(q), \Psi(q')\right) \in R. \tag{2.8}$$

Statt der Implikation wird hier die Äquivalenz zwischen Relationen in der Anfrage und Relationen im Dokumentbereich $Bild(\Psi)$ gefordert. Ein Beispiel für diese Unterscheidung ist in Abbildung 2.5 dargestellt.

Eine weitere Unterscheidungsart bei Trefferbegriffen ist, ob in einem Trefferdokument $D$ mit $Bild(\Psi) = T \subset D$ Verbindungen (Relationen) zwischen Objekten in $T$ und Objekten außerhalb von $T$, also $D \setminus T$, bestehen dürfen. In Abbildung 2.6 ist dies exemplarisch dargestellt. Bei Bauplänen, besonders bei der Verlegung von Belüftungskanälen, ist diese Unterscheidung stark von der Anfrage und den Vorstellungen des Architekten abhängig. Eine Auswahl des Trefferbegriffs ist somit wünschenswert. Eine "Abgeschlossenheit" des Trefferbereiches $Bild(\Psi)$ wird durch folgende Bedingung an die Trefferabbildung $\Psi$ realisiert:

$$\forall o \in Bild(\Psi) \text{ gilt: } \nexists o' \in D \setminus Bild(\Psi), R \in \mathcal{R} : (o, o') \in R \vee (o', o) \in R. \tag{2.9}$$

Diese Bedingung wird zusätzlich zu den unter (2.7) bzw. (2.8) formulierten Anforderungen an die Trefferabbildung gestellt.

**Fazit** In diesem Abschnitt wurden vier Trefferarten, d.h. vier verschiedene Trefferdefinitionen, beschrieben. Für eine Trefferabbildung muß man sich für eine der Anforderungen unter (2.7) bzw. (2.8) entscheiden. Zusätzlich kann die unter (2.9) formulierte Einschränkung an die Trefferabbildung gefordert werden. Mit dieser Auswahl an Trefferanforderungen bietet sich den Benutzern die Möglichkeit, zum Zeitpunkt einer Anfrage die Trefferart genauer zu spezifizieren. Die Auswirkungen dieser Auswahl ist in Abbildung 2.7 exemplarisch dargestellt, wobei sich folgende Treffer ergeben:

| Treffertyp | Bedingung | Treffer |
|---|---|---|
| S-Treffer | (2.7) | a), b), c), d) |
| SC-Treffer | (2.7) + (2.9) | a), c) |
| I-Treffer | (2.8) | a), b) |
| IC-Treffer | (2.8) + (2.9) | a) |

Die in der ersten Spalte aufgeführten Trefferbezeichnungen werden in Abschnitt 2.3 eingeführt. Sie dienen zur Bezeichnung von Treffern, die die in der 2. Spalte aufgeführten Bedingungen erfüllen.

### 2.2.3 Diskussion der Treffermodellierung

In diesem Abschnitt wird untersucht, inwieweit sich die verschiedenen Vorstellungen, die ein Benutzer (Architekt) von den gesuchten Treffern besitzt, in dem in den letzten Abschnitten beschriebenen

[1] Die Abbildung $\Psi$ ist bijektiv zwischen $Q$ und $Bild(\Psi)$. Somit ist die Umkehrabbildung $\Psi^{-1}$ für Elemente aus dem Bildbereich eindeutig bestimmt.

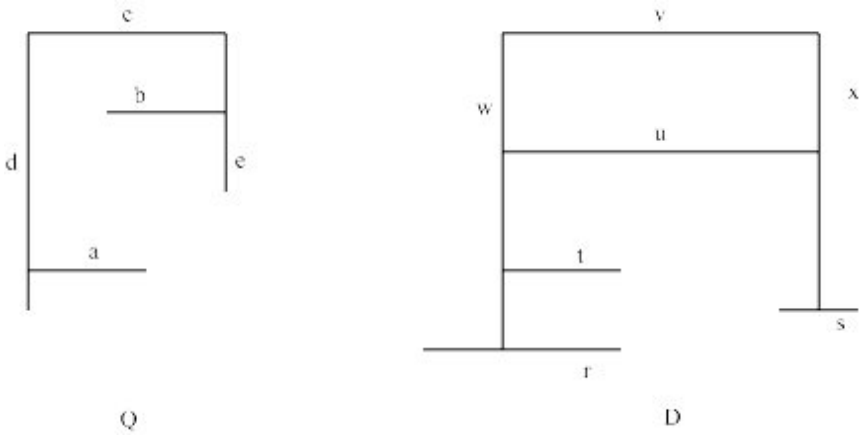



Abbildung 2.4: Links ist eine Anfrage $Q$ mit fünf Objekten $a, \ldots, e$, rechts ein Dokument $D$ mit sieben Objekten $r, \ldots, x$.

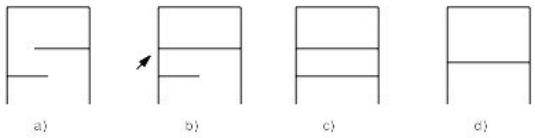



Abbildung 2.5: Soll bei einer Suche nach Konstellation a) die Konstellation b) als Treffer gefunden werden? Zu beachten ist, daß in Konstellation b) eine Relation (Verbindung) vorkommt (markiert mit einem Pfeil), die in der Anfrage a) nicht enthalten ist. Hier kann der Benutzer durch die Auswahl der Trefferdefinition selber entscheiden, ob Konstellation b) ein Treffer darstellt (mittels (2.7)) oder nicht (mittels (2.8)). Entscheidet er sich für (2.7), werden sowohl b) als auch c) als Treffer erkannt. Konstellation d) ist auf keinen Fall ein Treffer, da hier die Objektanzahl nicht mit der Anzahl der Anfrageobjekte übereinstimmt.

Abbildung 2.6: Soll bei einer Suche nach der linken Konstellation die rechts dargestellte als Treffer gefunden werden? Zu beachten ist, daß innerhalb der rechten Objektmenge ein Objekt existiert (mittlere senkrechte Objekt), das in der Anfrage nicht vorkommt. Hier kann der Benutzer durch die Auswahl der Trefferdefinition selber entscheiden (mittels (2.9)), ob die rechte Konstellation ein Treffer darstellt oder nicht.

Kontext modellieren lassen. Treffer hängen nicht nur von der Angabe des Anfragedokumentes ab, sondern auch von der Auswahl des Trefferbegriffs und der Menge von Relationen $\mathcal{R}$.

Auf die verschiedenen Trefferbegriffe wurde in Abschnitt 2.2.2 eingegangen. Im folgenden wird exemplarisch motiviert, wie vielfältig Anforderungen an Treffer durch das Zusammenspiel eines Anfragedokumentes und einer Menge von Relationen $\mathcal{R}$ gestellt werden können. Dabei werden verschiedene Relationen zur Beschreibung von Eigenschaften von Objektkonstellationen vorgestellt und die Auswirkungen, falls diese Relationen Bestandteil von $\mathcal{R}$ sind, auf eine Anfrage $Q$ diskutiert. Die Datenbasis $\mathcal{D}$ ist in Abbildung 2.8 dargestellt. Als Anfrage $Q$ wird Dokument $D_0$ verwendet.

Bisher wurden zur Beschreibung von Konstellationen nur Kreuzungen bzw. die verschiedenen Kreuzungstypen verwendet. Alle in Abbildung 2.8 dargestellten Dokumente sind Treffer bzgl. der Relationenmenge $\mathcal{R}_{Kreuzungen}$, einem beliebigen Trefferbegriff aus Abschnitt 2.2.2 und der Anfrage $Q = D_0$. Anfrage und Dokumente sind eher von akademischer Natur und dienen rein zur Demonstration. Für Dokument $D_0$ der Datenbasis ergeben sich folgende 4 Trefferabbildungen $\Psi : Q \rightarrow D_0$:

| $q$ | $\Psi_1(q)$ | $\Psi_2(q)$ | $\Psi_3(q)$ | $\Psi_4(q)$ |
|---|---|---|---|---|
| a | a | a | a | a |
| b | b | b | b | b |
| c | c | c | d | d |
| d | d | d | c | c |
| e | e | e | e | e |
| f | f | f | k | k |
| g | g | g | j | j |
| h | h | i | i | h |
| i | i | h | h | i |
| j | j | j | g | g |
| k | k | k | f | f |

Gleiches gilt für die restlichen Dokumente aus Abbildung 2.8. Auch hier besteht die Menge aller Trefferabbildungen aus den 4 oben angegebenen Abbildungen.

Als nächstes wird auf einige besondere Eigenschaften eingegangen, die in der Anfrage vorhanden sind, und die, je nach Anwendung bzw. Treffervorstellung des Benutzers, auch von den Treffern gefordert werden. Viele dieser Eigenschaften lassen sich aus der Gestaltpsychologie ableiten [26]. Die automatische Erkennung solcher immanenten Eigenschaften wird z.B. in [39] untersucht. Im Kontext der hier vorgestellten Suche liegt es allerdings am Anfragenden, durch die Angabe bzw. Auswahl von Relationen, die gewünschten Eigenschaften zu beschreiben bzw. zu selektieren.

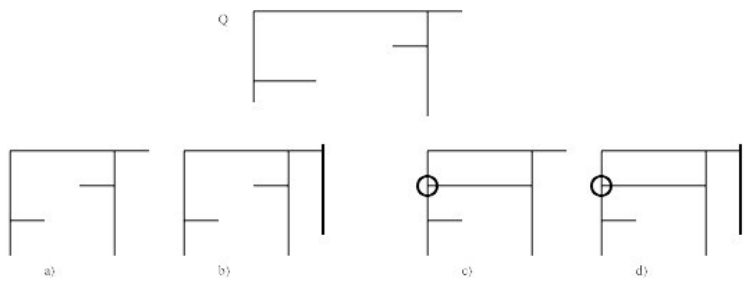



Abbildung 2.7: Oben ist eine Anfrage $Q$ dargestellt, darunter 4 potentielle Treffer, die je nach Auswahl von Zusatzbedingungen an die Trefferabbildung erkannt werden sollen oder nicht. Objekte, die nicht zu einem Treffer gehören, sind fett dargestellt (rechte senkrechte Objekt in b) und d)). Relationen, die nicht im Suchmuster vorkommen, sind durch einen Kreis markiert.

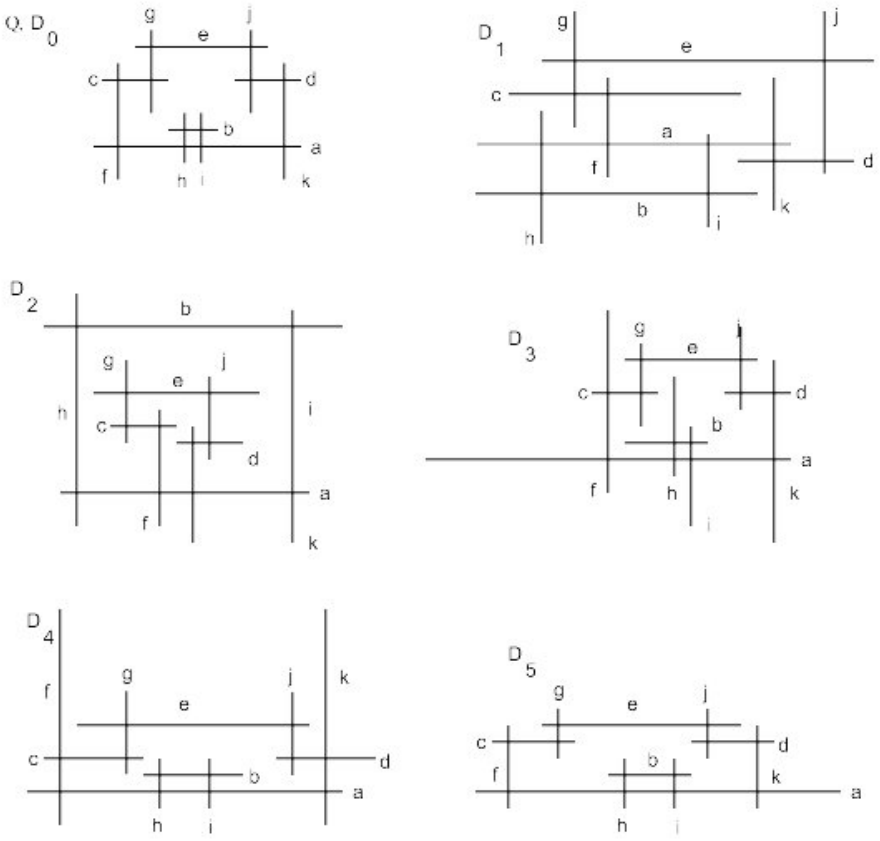



Abbildung 2.8: Sechs Dokumente aus der Architekturdomäne. Als Anfrage $Q$ wird Dokument $D_0$ der Datenbasis $\mathcal{D} = (D_0, D_1, D_2, D_3, D_4, D_5)$ verwendet. In jedem Dokument sind Objekte mit Buchstaben gekennzeichnet. Als einzige der Kreuzungsrelationen $\mathcal{R}_{Kreuzungen}$ kommen +-Kreuzungen vor.

**Längenvergleiche** Ein Aspekt, der in der Anfrage erfüllt ist und z.B. in Dokument $D_1$ nicht auftritt, ist die Lägengleichheit von Objekten. In $Q$ besitzen jeweils die Objekte $c$ und $d$, $f$ und $k$, $g$ und $j$ sowie $h$ und $i$ die gleiche Länge. Bei den entsprechenden Objekten in $D_1$ ist dies nicht der Fall.

Allgemein kann man jede Art von Lägenverhältnis und Lägendistanz durch Relationen beschreiben. Im folgenden wird nur die Gleichheit von Längen erfaßt. Dies geschieht durch eine zweistellige Relation auf der Menge aller Objekte $\mathcal{O}$. Sei Objekt $o = (x_1, y_1, x_2, y_2)$ und Objekt $o' = (x'_1, y'_1, x'_2, y'_2)$. Für ein Objekt $o \in \mathcal{O}$ wird mit $L_x(o) := x_2 - x_1$ die horizontale Länge, mit $L_y(o) := y_2 - y_1$ die vertikale Länge und mit $L(o) = max\{L_x(o), L_y(o)\}$ die Länge von $o$ bezeichnet. Für Objekte mit horizontaler Ausrichtung ergibt sich

$$R_- := \big\{(o, o') \in \mathcal{O} \times \mathcal{O} \mid L_x(o) = L_x(o') > 0\big\}$$

und für Objekte mit vertikaler Ausrichtung

$$R_| := \big\{(o, o') \in \mathcal{O} \times \mathcal{O} \mid L_y(o) = L_y(o') > 0\big\}.$$

Hierbei werden nur Objekte mit gleicher Ausrichtung verglichen. Möchte man die Längengleichheit zwischen Objekten beliebiger Ausrichtung beschreiben, so läßt sich dies durch folgende Relation $R_{gl}$ ($gl$ für gleich lang) ausdrücken:

$$R_{gl} := \big\{(o, o') \in \mathcal{O} \times \mathcal{O} \mid L(o) = L(o')\big\}.$$

Alle drei Relationen sind offensichtlich reflexiv, symmetrisch und transitiv.

Teilweise wird auch nur eine lokale Variante der Lägengleichheit gefordert. Diese betrifft nur Objekte, die mit einem gemeinsamen Objekt in Relation stehen. Dies kann wiederum durch folgende Relationen ausgedrückt werden:

$$\begin{aligned} R_{-_l} &:= \big\{(o, o') \in R_- \mid \exists p \in \mathcal{O},\ R, R' \in \mathcal{R}_{Kreuzungen} : (o, p) \in R \ \wedge\ (o', p) \in R'\big\}, \\ R_{|_l} &:= \big\{(o, o') \in R_| \mid \exists p \in \mathcal{O},\ R, R' \in \mathcal{R}_{Kreuzungen} : (p, o) \in R \ \wedge\ (p, o') \in R'\big\}. \end{aligned}$$

In Abbildung 2.8 sind die Anfrageobjekte $c$ und $d$ diejenigen, die nicht durch die lokale Längengleichheit erfaßt. Im Vergleich zur globalen Längengleichheit lassen sich in der Praxis durch die lokale Variante teilweise erhebliche Einsparungen bzgl. Speicherplatz und Berechnungszeit erzielen.

Wählt man $\mathcal{R} = \mathcal{R}_{Kreuzungen} \cup \{R_-, R_|\}$, so ist Dokument $D_1$ kein Treffer mehr, wohl aber alle restlichen Dokumente aus Abbildung 2.8.

**Horizontale und vertikale Reihenfolge von Objekten** Betrachtet man Dokument $D_2$ genauer, so stellt sich heraus, daß die Reihenfolge der Objekte bzgl. ihrer horizontalen und vertikalen Lage nicht mit der Anfrage übereinstimmen. In $D_2$ ist z.B. $b$ das oberste Objekt, während dies in $Q$ für $e$ gilt. Analog gilt, daß in $D_2$ $g$ links von $f$ liegt, wobei in $Q$ die beiden Objekte in der entgegengesetzten Reihenfolge liegen, $f$ links von $g$. Weiterhin besitzt $Q$ die Eigenschaft, daß die Objekte $c$ und $d$ auf der gleichen Höhe liegen, also mit horizontaler Ausrichtung die gleichen $y$-Koordinaten besitzen. Die erwähnten Eigenschaften, wiederum getrennt nach der Ausrichtung der Objekte, lassen sich durch folgende Relationen beschreiben:

$$\begin{aligned} R_{--} &:= \big\{(o, o') \in \mathcal{O} \times \mathcal{O} \mid y_1 = y_2 = y'_1 = y'_2\big\} \\ R_{l} &:= \big\{(o, o') \in \mathcal{O} \times \mathcal{O} \mid x_1 = x_2 = x'_1 = x'_2\big\} \\ R_{<} &:= \big\{(o, o') \in \mathcal{O} \times \mathcal{O} \mid y_1 = y_2 < y'_1 = y'_2\big\} \\ R_{\vee} &:= \big\{(o, o') \in \mathcal{O} \times \mathcal{O} \mid x_1 = x_2 < x'_1 = x'_2\big\}. \end{aligned}$$

Fügt man diese Relationen zu $\mathcal{R}$ hinzu, d.h. $\mathcal{R} = \mathcal{R}_{Kreuzungen} \cup \{R_-, R_|, R_{--}, R_l, R_<, R_\vee\}$, so ist bzgl. $Q$ Dokument $D_2$ kein Treffer mehr. Für alle Dokumente der Treffermenge $\{D_3, D_4, D_5\}$ gilt, daß nur noch eine Trefferabbildung pro Dokument existiert, die jeweils $\Psi_1$ aus der obigen Tabelle gleicht.

Die exakte parallele Lage von zwei Objekten, dessen jeweiligen Endpunkte auf gleicher Höhe liegen, wurde mit den bisherigen Relationen noch nicht erfaßt. In der Anfrage betrifft dies alle Objekte mit jeweils gleicher Länge, in Dokument $D_3$ trifft dies nicht zu. Fügt man die beiden folgenden Relationen noch zu $\mathcal{R}$ hinzu, so stellt $D_3$ keinen Treffer mehr dar.

$$\begin{aligned} R_{=} &:= \left\{(o,o') \in R_{-} \mid x_1 = x_2 = x'_1 = x'_2\right\} \\ R_{\parallel} &:= \left\{(o,o') \in R_{\parallel} \mid y_1 = y_2 = y'_1 = y'_2\right\} \end{aligned}$$

Weiterhin kann man in noch feinere Kreuzungstypen klassifizieren, was in Abbildung 2.9 angedeutet ist. Damit würde man u.U. Dokument $D_3$ als Treffer ausschließen, da in $D_0$ der Schnittpunkt zwischen den Objekten $a$ und $f$ im linken Drittel liegt, während sich dieser für $a$ und $f$ in $D_3$ im mittleren Drittel befindet.

Auch lassen sich funktionale Zusammenhänge manuell vom Architekten ausdrücken, z.B. wenn bestimmte Objekte immer gemeinsam ausgetauscht werden müssen. Auch dies läßt sich durch Relationen modellieren, wobei hier die zueinander in Relation stehenden Objekte in der Anfrage auch manuell gekennzeichnet werden müssen.

**Fazit** Treffer hängen stark von der ausgewählten Menge von Relationen ab. Erst dadurch erhalten Anfragen eine gewisse Semantik, d.h. es wird festgelegt, welche Eigenschaften der Anfrage alle Treffer enthalten müssen.

Die Beispiele haben gezeigt, wie man mit Hilfe der Relationen bestimmte Eigenschaften der Anfrage beschreiben kann und wie schrittweise die Treffermenge verfeinert wird. Es liegt also am Experten, die für ein Anwendungsgebiet inhärenten Eigenschaften der Konstellationen durch Relationen zu definieren.

## 2.3 Trefferdefinitionen

Nachdem im vorherigen Abschnitt verschiedene Trefferarten für die Anwendung im Architekturbereich beschrieben wurden, werden jetzt Treffer allgemein für beliebige Anwendungen definiert.

Sei $\mathcal{O}$ die Menge aller möglichen Objekte einer Anwendung und $\mathcal{D}$ eine Datenbasis mit Dokumenten $D \subseteq \mathcal{O}$.

Ausgangspunkt für die folgenden Trefferbegriffe bzgl. einer Anfrage $Q \subseteq \mathcal{O}$ ist die Eigenschaft, daß Objekte nicht isoliert miteinander verglichen werden, wie es bei der exakten Suche in Abschnitt 2.1.1 der Fall war, sondern daß Beziehungen *zwischen* zwei oder mehreren Objekten entscheidend sind. Beispiele für Beziehungen zwischen Objekten sind die im vorherigen Abschnitt beschriebenen verschiedenen Verbindungen (Kreuzungen) zwischen Leitungen oder Kanälen.

Beziehungen zwischen Objekten werden durch Relationen $R$ modelliert. Weiterhin wird, abhängig von einer Anwendung, zwischen verschiedenen Arten (Typen) von Beziehungen unterschieden. Bei der Suche in Bauplänen wurde die Beziehung "Verbindung" in 9 verschiedene Beziehungstypen (Kreuzungstypen) unterschieden (vergleiche Abbildung 2.3).

Jeder Beziehungstyp $\gamma$ wird durch eine eigene Relation $R_\gamma$ modelliert, wobei sich die Stelligkeit der Relation $R_\gamma$ nach der Anzahl der Objekte richtet, die gemeinsam in dieser konkreten Beziehung

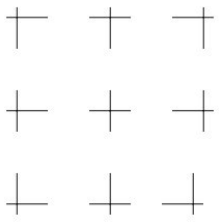


Abbildung 2.9: Eine feinere Unterteilung des +-Kreuzungstyps in neun Kreuzungstypen.

stehen. Bei den Kreuzungstypen der Architekturdomäne standen immer genau zwei Objekte in Beziehung zueinander, d.h. alle Relationen waren zweistellig.

In dieser Arbeit beschränken wir uns auch bei der allgemeinen Trefferdefinition auf zweistellige Relationen (Beziehungen), d.h.

$$R_\gamma \subseteq \mathcal{O} \times \mathcal{O}.$$

Es werden bewußt keine weiteren Anforderungen, etwa Symmetrie oder Transitivität, an die Relationen gestellt. Relationen können sowohl durch Berechnungsvorschriften, wie z.B. alle bisher alle Relationen definiert wurden, oder explizit durch eine Auflistung der Objektpaare definiert werden.

Mit $\mathcal{R}$ wird die endliche Menge aller Relationen, die Beziehungstypen realisieren, bezeichnet. Diese Menge $\mathcal{R}$ ist stark von der jeweiligen konkreten Anwendung abhängig und wird i.d.R. von einem Experten vorgegeben. Bei der Suche in Bauplänen bestand $\mathcal{R}$ aus neun Relationen, jeweils eine pro Kreuzungstyp, d.h. $\mathcal{R} = \mathcal{R}_{Kreuzungen}$.

Für den weiteren Verlauf werden folgende Bezeichnungen verwendet:
Für eine Relation $R_\gamma \subseteq \mathcal{O} \times \mathcal{O}$ und eine Menge $T \subseteq \mathcal{O}$ wird $R_\gamma(T)$ durch

$$R_\gamma(T) := R_\gamma \cap (T \times T)$$

definiert. Hiermit wird eine Relation auf eine Objektmenge $T$ eingeschränkt.

Für eine injektive Abbildung $\Psi : Q \to D$ und eine Objektmenge $T \subseteq Q$ ist $\Psi(T)$ definiert durch

$$\Psi(T) := \{\Psi(o) \mid o \in T\}.$$

Weiterhin werden für $a, b \in Q$ und für eine Relation $R \subseteq \mathcal{O} \times \mathcal{O}$ folgende Kurzschreibweisen verwendet:

$$\begin{aligned} \Psi^2((a,b)) &:= (\Psi(a), \Psi(b)) \\ \Psi^2(R) &:= \{\Psi^2((a,b)) \mid (a,b) \in R\}. \end{aligned}$$

Kommen wir nun zu einer Anforderung, die an alle Anfrage gestellt wird:

**Annahme 2.3.1.** Jede Anfrage $Q$ muß die Bedingung

$$\forall q \in Q \exists q' \in Q, R_\gamma \in \mathcal{R} : q \neq q' \wedge ((q,q') \in R_\gamma \vee (q',q) \in R_\gamma)$$

erfüllen.

Damit wird sichergestellt, daß jedes Anfrageobjekt zu mindestens einem anderen in Beziehung steht. Es werden also einzelne Objekt ausgeschlossen, die keine Anbindung an die restliche Anfrage besitzen.

Abschnitt 2.4 wird zeigen, daß die im folgenden definierten Trefferbegriffe mit Problemen aus der Graphentheorie vergleichbar sind. Dort wird die Gleichheit von Graphen durch eine Isomorphie und eine Art von Enthaltensein durch eine Subgraphenisomorphie ausgedrückt. Aus diesem Zusammenhang heraus haben die vier Trefferbegriffe ihren Namen erhalten: I-Treffer für eine **I**somorphie oder Gleichheit zwischen Anfrage und Treffer, S-Treffer für **S**ubgraphenisomorphie bzw. Enthaltensein. Zusätzlich kann eine Art Abgeschlossenheit (**C**losure) der Treffer verlangt werden, wodurch sich IC-Treffer und SC-Treffer ergeben. Diese vier Trefferbegriffe werden im folgenden definiert.

### 2.3.1 I-Treffer und S-Treffer

Sei eine Datenbasis $\mathcal{D}$ und $\mathcal{R} = \{R_1, \ldots, R_l\}$ eine Menge von Relationen $R_\gamma \subseteq \mathcal{O} \times \mathcal{O}$ gegeben. Dann lassen sich für eine Anfrage $Q$ folgende Trefferbegriffe definieren:

**Definition 2.3.2.** Ein Dokument $D \in \mathcal{D}$ wird genau dann als **I-Treffer** der Anfrage $Q$ bezeichnet, wenn eine injektive Abbildung $\Psi : Q \to D$ existiert, so daß für alle $R_\gamma \in \mathcal{R}$ gilt:

$$\boxed{\Psi^2\big(R_\gamma(Q)\big) = R_\gamma\big(\Psi(Q)\big).}$$

**Definition 2.3.3.** Ein Dokument $D \in \mathcal{D}$ wird genau dann als **S-Treffer** der Anfrage $Q$ bezeichnet, wenn eine injektive Abbildung $\Psi : Q \to D$ existiert, so daß für alle $R_\gamma \in \mathcal{R}$ gilt:

$$\boxed{\Psi^2\big(R_\gamma(Q)\big) \subseteq R_\gamma\big(\Psi(Q)\big).}$$

**Definition 2.3.4.** Sei $D$ ein Treffer bzgl. Anfrage $Q$ und injektiver Abbildung $\Psi : Q \to D$. Dann wird $\Psi$ als **Trefferabbildung** bezeichnet.

### 2.3.2 IC-Treffer und SC-Treffer

Die beiden bisher definierten Trefferbegriffe lassen sich jeweils wie folgt verfeinern:

**Definition 2.3.5.** Sei $D \in \mathcal{D}$ ein I-Treffer (S-Treffer) bzgl. der Trefferabbildung $\Psi : Q \to D$. Dann wird $D$ als **IC-Treffer** (**SC-Treffer**) bezeichnet, falls für $\Psi$ zusätzlich

$$\forall o \in \Psi(Q) \nexists o' \in D \setminus \Psi(Q), R \in \mathcal{R} : (o, o') \in R \vee (o', o) \in R$$

gilt.

Im vorherigen Abschnitt wurden Beispiele für diese Trefferbegriffe für die Suche in Bauplänen angegeben. Die dort diskutierten Treffer hängen wie folgt mit den hier beschriebenen Definitionen zusammen: Treffer bzgl. Gleichung (2.8) sind I-Treffer, bzgl. Gleichung (2.7) S-Treffer. Als IC-Treffer bzw. SC-Treffer ergeben sich in der Anwendung diejenigen, die die Gleichungen (2.8) und (2.9) bzw. (2.7) und (2.9) erfüllen.

**Beispiel 2.3.6.** Betrachten wir nochmals die in die Abbildung 2.5 dargestellte Anfrage $Q$ mit potentiellen Trefferdokumenten a) bis d) aus der Architekturdomäne. Mit $\mathcal{R}_{Kreuzungen}$ als Menge aller Relationen ergeben sich als S-Treffer die Dokumente a), b), c) und d). SC-Treffer bilden a) und c). Die Dokumente a) und b) sind I-Treffer. Nur Dokument a) stellt einen IC-Treffer dar. ∘

Wie Beispiel 2.3.6 demonstriert, besteht eine Hierarchie zwischen den einzelnen Trefferbegriffen. Ein IC-Treffer ist auch gleichzeitig I-Treffer, SC-Treffer und S-Treffer. SC-Treffer als auch I-Treffer sind gleichzeitig S-Treffer. Als Hierarchie ergibt sich:

$$\begin{array}{ccc} & \text{S-Treffer} & \\ \cup\| & & \cup\| \\ \text{SC-Treffer} & (\neq) & \text{I-Treffer} \\ \cup\| & & \cup\| \\ & \text{IC-Treffer} & \end{array}$$

Die Ungleichheit zwischen SC-Treffern und I-Treffern gilt nur für Treffer, die keinen IC-Treffer darstellen. Alle in der Hierarchie dargestellten Inklusionen ergeben sich direkt aus den Definitionen der Trefferbegriffe.

Im vorherigen Abschnitt wurden für die Beispielanwendung in der Architekturdomäne geringfügig andere Trefferdefinitionen eingeführt. Daß diese äquivalent zu den oben aufgeführten sind, zeigt folgende Bemerkung.

**Bemerkung 2.3.7.** Ein Dokument $D$ ist genau dann ein I-Treffer bzgl. einer Anfrage $Q$, wenn eine injektive Abbildung $\Psi : Q \to D$ mit

$$\forall R \in \mathcal{R}_{Kreuzungen} \forall q, q' \in Q : (q, q') \in R \Leftrightarrow \big(\Psi(q), \Psi(q')\big) \in R$$

existiert.

**Beweis** Die Behauptung folgt aus einfachen Umformungen der Trefferdefinition 2.3.2 und Gleichung (2.7). Für alle $R_i \in \mathcal{R}$ gilt:

$$\begin{aligned}
\Psi^2\big(R_i(Q)\big) &= \Psi^2\big(R_i \cap Q \times Q\big) \\
&= \Psi^2\big(\{(a,b) \in Q \times Q \mid (a,b) \in R_i\}\big) \\
&= \{(\Psi(a), \Psi(b)) \mid (a,b) \in Q \times Q, (a,b) \in R_i\} \\
&= \{(\Psi(a), \Psi(b)) \mid (a,b) \in Q \times Q, (\Psi(a), \Psi(b)) \in R_i\} \\
&= R_i(\{\Psi(a) \mid a \in Q\}) \\
&= R_i\big(\Psi(Q)\big)
\end{aligned}$$

Die Gleichheit der in der 3. und 4. Zeile definierten Mengen ist genau dann erfüllt, wenn $(a,b) \in R_i \Leftrightarrow (\Psi(a), \Psi(b)) \in R_i$ gilt. □

Für die anderen Treffertypen wird die Äquivalenz zwischen den in diesem Abschnitt und den im vorherigen Abschnitt angegebenen Definitionen analog bewiesen.

**Fazit** In diesem Abschnitt wurden verschiedene allgemeine Trefferbegriffe eingeführt, die bisher in dieser Form noch nicht untersucht wurden. Der Benutzer entscheidet sich bei einer Anfrage für einen der Trefferbegriffe. S-Treffer bilden die allgemeinste Form von Treffern, IC-Treffer die am weitesten eingeschränkte. Diese Unterteilung bietet dem Benutzer nicht nur flexible Suchmöglichkeiten, sondern hat auch praktische Gründe: die Suche nach IC-Treffern geschieht i.d.R. um ein Vielfaches schneller als die Suche nach S-Treffern (siehe Kapitel 4).

An dieser Stelle kommen wir nochmals auf die zu Beginn des Kapitels angesprochenen Beschreibungsmöglichkeiten von Anfragen zurück. Bei der exakten Suche wurde die Anfrage als Ganzes mittels einer Operation transformiert. Eine Musikanfrage durfte z.B. zeitlich verschoben werden. Dies wurde als allgemeines Konzept in Abschnitt 2.1.2 beschrieben.

Die in diesem Abschnitt eingeführten Trefferdefinitionen beschreiben eine Konstellation anhand der zwischen den Objekten geltenden Relationen. Treffer werden allein durch diese Beziehungen definiert: stehen zwei Objekte $o, o'$ der Anfrage in Relation $\gamma$, so stehen auch die entsprechenden Objekte im Treffer $(\Psi(o), \Psi(o'))$ in Relation $\gamma$. Durch die verschiedenen Trefferbegriffe kann festgelegt werden, ob neben den durch die Anfrage festgelegten Relationen noch andere Relationen zwischen Objekte im Treffer vorkommen dürfen.

Im Gegensatz zur exakten Suche wird eine Anfrage nicht als Ganzes transformiert, sondern es werden alle Transformationen berücksichtigt, die die Anfrage so modifizieren, daß die Beziehungen erhalten bleiben. Dabei ist die Transformation nebensächlich, solange die Beziehungen der Objekte untereinander noch übereinstimmen.

Eine Kombination dieser Suchmöglichkeiten, d.h. Übereinstimmung von bestimmten Objektattributen einerseits (wobei die Objekte isoliert betrachtet werden) und von Beziehungen zwischen Objekten andererseits (gleichzeitige Betrachtung mehrere Objekte untereinander), ist natürlich direkt anwendbar. Dies werden die nächsten Kapitel zeigen, in denen Verfahren zur Suche nach Treffern vorgestellt werden.

## 2.4 Trefferberechnung als Problem der Graphentheorie

In diesem Abschnitt wird ein Bezug zur Graphentheorie hergestellt. Es wird gezeigt, daß sich Dokumente als beschriftete Graphen darstellen lassen. Eine Datenbasis $\mathcal{D}$ aus Dokumenten entspricht dann einer Datenbasis aus Graphen. Die Treffersuche läßt sich dann als Isomorphieproblem zwischen Graphen ausdrücken, d.h. in allen Graphen der Datenbasis soll nach Treffern gesucht werden.

### 2.4.1 Definitionen

Im weiteren Verlauf dieser Arbeit werden die folgenden Definitionen verwendet, die sich an Bezeichnungsweisen aus [20] und [12] anlehnen:

**Definition 2.4.1.** Ein **beschrifteter Graph** $G = (V, \zeta, E, \xi)$ ist ein gerichteter Graph mit Abbildungen $\zeta : V \to W_V$ und $\xi : E \to W_E$. Die Wertebereiche $W_V$ und $W_E$ sind halbgeordnete Mengen.

Für eine halbgeordnete Menge $(W, \leq)$ gelten die drei Bedingungen $\forall w \in W : w \leq w$, $\forall a, b, c \in W : a \leq b \wedge b \leq c \Rightarrow a \leq c$ und $\forall a, b \in W : a \leq b \wedge b \leq a \Rightarrow a = b$.

$W_V$ ist die Menge aller Knotenbeschriftungen und $W_E$ ist die Menge aller Kantenbeschriftungen. Mittels der beiden Abbildungen $\zeta$ und $\xi$ erhält man für einen Knoten $v$ die Knotenbeschriftung von $v$ durch $\zeta(v)$ und für eine Kante $e$ die Kantenbeschriftung von $e$ mittels $\xi(e)$.

**Definition 2.4.2.** Ein **Subgraph** $G' = (V', \zeta', E', \xi')$ von einem Graphen $G = (V, \zeta, E, \xi)$ ist ein Graph mit $V' \subseteq V$, $E' \subseteq E \cap (V' \times V')$, $\zeta' = \zeta \downarrow V'$ und $\xi' = \xi \downarrow E'$. $G'$ ist der von $V'$ **induzierte** Subgraph von $G$ gdw. $E' = E \cap (V' \times V')$. Der zu $V'$ induzierte Subgraph von $G$ wird mit $G \downarrow V'$ bezeichnet.

**Definition 2.4.3.** Sei $G = (V, \zeta, E, \xi)$ ein Graph. Gilt $E = V \times V \setminus \Delta, \Delta := \{(v, v) | v \in V\}$, so wird $G$ als **vollständig** bezeichnet. $G$ heißt **zusammenhängend**, wenn $\forall v, w \in V$ eine Folge von Knoten $(v_i)_{1 \leq i \leq l} \in V^l$ mit $v_1 = v$, $v_l = w$ und $\forall 1 \leq i < l : (v_i, v_{i+1}) \in E \vee (v_{i+1}, v_i) \in E$ existiert.

Bei den für die Definition von zusammenhängenden Graphen verwendeten Folgen von Knoten $(v_i)_{1 \leq i \leq l}$ ist die Richtung der Kanten zwischen zwei benachbarten Knoten nicht relevant.

Beschriftete Graphen lassen sich durch die folgenden Definitionen miteinander vergleichen. Dabei seien $G = (V, \zeta : V \to W_V, E, \xi : E \to W_E)$ und $G' = (V', \zeta' : V' \to W_{V'}, E', \xi' : E' \to W_{E'})$ zwei beschriftete Graphen.

**Definition 2.4.4.** Ein Graph $G$ ist **isomorph** zu einem Graphen $G'$, $G \simeq G'$, wenn Bijektionen $\phi : V \to V'$, $\alpha : W_V \to W_{V'}$, $\beta : W_E \to W_{E'}$ existieren, so daß $\Phi_E := \phi \times \phi \downarrow E$ eine bijektive Abbildung von $E$ nach $E'$ ist und

$$\begin{aligned} \zeta &= \alpha^{-1} \circ \zeta' \circ \phi \\ \xi &= \beta^{-1} \circ \xi' \circ \Phi_E \end{aligned}$$

gilt.

Hiermit wird eine Art "Gleichheit" zwischen Graphen definiert. Eine Form von "Enthaltensein" läßt sich wie folgt beschreiben.

**Definition 2.4.5.** Ein Graph $G$ ist **subisomorph** zu einem Graphen $G'$, $G \preceq G'$, wenn Bijektionen $\phi : V \to V'$, $\alpha : W_V \to W'_V$, $\beta : W_E \to W'_E$ existieren, so daß $\Phi_E := \phi \times \phi \downarrow E$ eine bijektive Abbildung von $E$ nach $E'$ ist und

$$\begin{aligned} \zeta &\leq \alpha^{-1} \circ \zeta' \circ \phi \\ \xi &\leq \beta^{-1} \circ \xi' \circ \Phi_E \end{aligned}$$

gilt. Dabei gilt $f \leq g$ für zwei Abbildungen $f : A \to B$, $g : A \to B$ mit halbgeordneter Menge $B$, wenn $f(a) \leq g(a), \forall a \in A$.

Man beachte, daß hier beide Graphen die gleiche Anzahl an Knoten und Kanten besitzen müssen. Dies steht im Gegensatz zur Definition in [12].

Der Vergleich zweier Graphen beschränkt sich im weiteren auf Graphen, deren Wertebereiche für Knotenbeschriftungen und deren Wertebereiche für Kantenbeschriftungen gleich sind, d.h. $W_V = W_{V'}$ und $W_E = W_{E'}$. Als bijektive Abbildungen $\alpha$ und $\beta$ wird jeweils die identische Abbildung verwendet. Damit muß für die Isomorphie lediglich $\zeta = \zeta' \circ \phi$ und $\xi = \xi' \circ \Phi_E$ gelten. Für die Subisomorphie gilt Analoges bzgl. $\leq$.

Betrachtet man eine echte Teilmenge $V'$ der Knotenmenge $V$ eines Graphen $G$, so läßt sich eine Erweiterung von $V'$ bzgl. $G$ wie folgt beschreiben:

**Definition 2.4.6.** Sei $G = (V, \zeta, E, \xi)$ ein Graph und $V' \subseteq V$. Dann ist $\nu_G(V') = \nu(V') := V' \cup E(V') \cup E^{-1}(V')$ eine Erweiterung von $V'$ um alle unmittelbaren Nachbarknoten von Knoten aus $V'$.

Mit Hilfe der in diesem Abschnitt eingeführten Definitionen wird als nächstes ein Bezug zwischen Dokumenten und beschrifteten Graphen hergestellt.

### 2.4.2 Dokumente als Graphen

Nachdem im vorherigen Abschnitt beschriftete Graphen eingeführt wurden, wird nun beschrieben, wie sich jedes Dokument einer Datenbasis bzgl. $\mathcal{O}$ und $\mathcal{R}$ als Graph darstellen läßt. Dabei entspricht jedem Dokumentobjekt ein Knoten im Graphen. Zwei Knoten sind durch eine Kante verbunden, wenn die den Knoten entsprechende Objekte in Relation zueinander stehen. In der Kantenbeschriftung wird festgehalten, in welcher Relation die entsprechenden Objekte zueinander stehen. Da zwei Objekte in mehreren Relationen zueinander stehen können, wird als Wertebereich $W_E$ die Potenzmenge von $\mathcal{R}$ verwendet, d.h. eine Kantenbeschriftung ist eine Menge von Relationen bzw. Relationsbezeichnungen. Durch Knotenbeschriftungen lassen sich z.B. inhärente Objekteigenschaften repräsentieren. Hierauf wird in Abschnitt 2.5 eingegangen. Da in dieser Arbeit die Beziehungen zwischen Objekten im Vordergrund stehen, werden Knotenbeschriftungen i.d.R. nicht verwendet.

Kommen wir nun zur Darstellung von Dokumenten durch beschriftete Graphen. Hierbei wird die Objektmenge $D$ eines Dokumentes als Knotenmenge des Graphen verwendet. So entsteht eine $1:1$ Beziehung zwischen Knoten und Objekten.

**Definition 2.4.7.** Der zu einem Dokument $D$ und einer Menge $\mathcal{R} = \{R_1, \ldots, R_m\}$ von Relationen $R_\gamma$ gehörige Graph $G(D, \mathcal{R}) = (D, \zeta_D, E_D, \xi_D)$ ist definiert durch: $E_D = \big\{(v_i, v_j) \in D \times D \mid \exists R \in \mathcal{R} : (v_i, v_j) \in R\big\}$ und $\xi : E_D \to 2^{[1:m]}$ mit $\xi_D\big((v_i, v_j)\big) = \{\gamma \in [1:m] \mid (v_i, v_j) \in R_\gamma\}$. Die Knotenbeschriftungen bleiben undefiniert, da sie für die Trefferberechnung nicht verwendet werden. Der Wertebereich $W_E$ der Kantenbeschriftungen ist gleich $2^{[1:m]}$. Dieser stellt bzgl. der Mengeninklusion "$\subseteq$" eine halbgeordnete Menge dar.

Knotenbeschriftungen bzw. $\zeta$ werden im weiteren Verlauf dieser Arbeit nicht weiter betrachtet, da sie im Suchprozeß nicht verwendet werden.

**Beispiel 2.4.8.** In Abbildung 2.10 ist ein Bauplan $D$ aus der Architekturdomäne zusammen mit dem dazugehörigen Graphen $G(D, \mathcal{R}_{Kreuzungen})$ dargestellt. Der Graph ist zusammenhängend und nicht planar. ∘

Bezeichnungen, die für Graphen gelten, werde für die den Graphen entsprechenden Dokumente übernommen. Ein Dokument $D$ heißt **zusammenhängend** bzgl. einer Menge von Relationen $\mathcal{R}$, wenn der Graph $G(D, \mathcal{R})$ zusammenhängend ist.

Im folgenden Kontext wird ein Subgraph $G'$ von $G$ als **maximal** bezeichnet, wenn kein zusammenhängender Subgraph $G''$ von $G$ existiert, der $G'$ als Subgraph enthält.

In Abschnitt 2.3 wurde für jede Anfrage $Q$ gefordert, daß jedes Objekt der Anfrage mit mindestens einem anderen Objekt in einer Relation aus $\mathcal{R}$ steht (Annahme 2.3.1). Diese Forderung wird nun weiter verschärft. Im weiteren Verlauf dieser Arbeit wird von folgender Annahme ausgegangen:

**Annahme 2.4.9.** Jedes Dokument $D$ einer Datenbasis und jede Anfrage $Q$ ist zusammenhängend.

Wendet man die Definition von zusammenhängenden Dokumenten an, so bedeutet die Annahmen, daß für jedes Dokument $D$

$$\forall o \in D \exists o' \in D, R_\gamma \in \mathcal{R} : o \neq o' \wedge ((o, o') \in R_\gamma \vee (o', o) \in R_\gamma).$$

gilt. Analoges gilt für Anfragen $Q$.

Annahme 2.4.9 läßt sich leicht erfüllen. Jedes reale Dokument wird in maximal zusammenhängende Dokumente unterteilt und jedes dieser maximal zusammenhängenden Dokumente bildet ein

abstraktes Dokument der Datenbasis. In diesem Zusammenhang ist ein Teil $D'$ eines Dokumentes $D$ maximal zusammenhängend, wenn der Subgraph $G(D', \mathcal{R})$ von $G(D, \mathcal{R})$ maximal zusammenhängend ist.

Die Eigenschaft, daß alle Dokumente der Datenbasis und alle Anfragen nach Annahme 2.4.9 zusammenhängend sind, bilden eine wichtige Voraussetzung für effiziente Suchverfahren. Diese werden in späteren Kapiteln vorgestellt.

### 2.4.3 Trefferberechnung als Graphenisomorphieproblem

In Abschnitt 2.3 wurden vier Trefferbegriffe für die Konstellationssuche eingeführt. Diese lassen sich auf die Graphentheorie übertragen. Sei im folgenden $\mathcal{R}$ eine Menge von Relationen, $Q$ eine Anfrage mit $G_Q := G(Q, \mathcal{R}) = (Q, \zeta_Q, E_Q, \xi_Q)$ und $D$ ein Dokument mit $G_D := G(D, \mathcal{R}) = (D, \zeta_D, E_D, \xi_D)$. Für eine Abbildung $\phi : A \to B$ wird für eine Teilmenge $A'$ von $A$ die Menge $\phi(A')$ definiert durch $\{\phi(a) \mid a \in A'\}$.

Ist das Dokument $D$ ein I-Treffer bzgl. der Anfrage $Q$, so gilt für die entsprechenden Graphen: Es existiert eine injektive Abbildung $\phi : Q \to D$ mit $G_Q \simeq G_D \downarrow \phi(Q)$.

Falls $D$ einen IC-Treffer der Anfrage $Q$ darstellt, gilt zusätzlich zur Eigenschaft von I-Treffern: $\nu(\phi(Q)) = \phi(Q)$. Dies bedeutet, daß sich die Knotenmenge $\phi(Q)$ nicht ändert, wenn man von jedem Knoten seine direkten Nachbarknoten mit einbezieht. Da nach Annahme 2.4.9 alle Dokumente, somit auch $G_D$ und $G_Q$, zusammenhängend sind, bildet $\phi$ sogar eine bijektive Abbildung zwischen $Q$ und $D$.

Für einen S-Treffer $D$ von $Q$ existieren injektive Abbildungen $\phi : Q \to D$ und $\psi : E_Q \to E_D$, so daß $G'_D = (\phi(Q), \zeta_D \downarrow \phi(Q), \psi(E_Q), \xi_D \downarrow \psi(E_Q))$ ein Subgraph von $G_D$ ist und $G_Q \preceq G'_D$ gilt.

Falls $D$ einen SC-Treffer von $Q$ darstellt, gilt $\nu(\phi(Q)) = \phi(Q)$ zusätzlich zu den Eigenschaften von S-Treffern. Dies ist genau dann der Fall, wenn $\phi$ eine Bijektion ist.

Daraus ergibt sich der folgende Satz:

**Satz 2.4.10.** *Mit obigen Bezeichnungen für $G_D$ und $G_Q$ bzgl. eines Dokuments $D$ und einer Anfrage $Q$ gilt:*

1. *Genau dann ist $D$ ein IC-Treffer von $Q$, wenn eine bijektive Abbildung $\Psi : Q \to D$ mit $G_Q \simeq G_D$ existiert.*
2. *Genau dann ist $D$ ein I-Treffer von $Q$, wenn eine injektive Abbildung $\Psi : Q \to D$ mit $G_Q \simeq G_D \downarrow \Psi(Q)$ existiert.*
3. *Genau dann ist $D$ ein SC-Treffer von $Q$, wenn eine bijektive Abbildung $\Psi : Q \to D$ mit $G_Q \preceq G'_D$ für $G'_D = (D, \zeta_D, \Psi^2(E_Q), \xi_D \downarrow \Psi^2(E_Q))$ existiert.*
4. *Genau dann ist $D$ ein S-Treffer von $Q$, wenn eine injektive Abbildung $\Psi : Q \to D$ mit $G_Q \preceq G'_D$ für $G'_D = (\Psi(Q), \zeta_D \downarrow \Psi(Q), \Psi^2(E_Q), \xi_D \downarrow \Psi^2(E_Q))$ existiert.*

**Beweis** Als erstes wird die Aussage über I-Treffer bewiesen, die sich hauptsächlich aus der Konstruktion der Graphen für Dokumente, Definition 2.4.7, und der Trefferdefinition 2.3.2 ergibt. Danach wird die 4. Aussage bewiesen. Die restlichen Aussagen ergeben sich aus den beiden bereits bewiesenen Fällen.
(2)"⇐": Sei $\Psi : Q \to D$ eine Injektion mit $G_Q \simeq G_D \downarrow \Psi(Q)$. Dann gilt:

$$(v, w) \in E_Q \wedge \gamma \in \xi_Q((v, w)) \Leftrightarrow (\Psi(v), \Psi(w)) \in E_D \wedge \gamma \in \xi_D((\Psi(v), \Psi(w)))$$

für alle $v, w \in Q$ und $\gamma \in [1 : |\mathcal{R}|]$. Nach Konstruktion des Graphen (Definition 2.4.7) ist dies äquivalent zu

$$(v, w) \in R_\gamma \Leftrightarrow (\Psi(v), \Psi(w)) \in R_\gamma$$

für alle $v, w \in Q$ und $\gamma \in [1 : |\mathcal{R}|]$. Damit ist $\Psi$ eine Trefferabbildung bzgl. I-Treffer.
(2)"⇒": Der Beweis erfolgt analog zu obigen, da sich die für die Isomorphie benötigte Abbildung

aus der für einen I-Treffer gegebenen Trefferabbildung $\Psi$ ergibt.
(4)"$\Leftarrow$": Nach Definition 2.4.5 besteht für $G_Q \preceq G'_D$ eine Bijektion zwischen den Knotenmengen und eine Bijektion zwischen den Kantenmengen, die sich auch als Injektionen bzgl. $G_D$ darstellen lassen. O.B.d.A. seien diese durch $\Psi$ definiert. Dann ergibt sich die Inklusion

$$(v,w) \in E_Q \wedge \gamma \in \xi_Q((v,w)) \Rightarrow (\Psi(v),\Psi(w)) \in E_D \wedge \gamma \in \xi_D((\Psi(v),\Psi(w)))$$

für alle $v, w \in Q$ und $\gamma \in [1 : |\mathcal{R}|]$. Nach Konstruktion des Graphen (Definition 2.4.7) ist dies äquivalent zu

$$(v,w) \in R_\gamma \Rightarrow (\Psi(v),\Psi(w)) \in R_\gamma$$

für alle $v, w \in Q$ und $\gamma \in [1 : |\mathcal{R}|]$. Damit ist $\Psi$ eine Trefferabbildung bzgl. S-Treffer.
(4)"$\Rightarrow$" wird analog bewiesen.
$(1), (3)$ : Zusätzlich zu obigen Beweisen gilt, da alle Graphen und Dokumente zusammenhängend sind, $\Psi$ ist genau dann eine Bijektion, wenn $\nu(\Psi(Q)) = \Psi(Q)$ gilt. □

Die aufgeführten Eigenschaften veranschaulicht das folgende Beispiel.

**Beispiel 2.4.11.** In Abbildung 2.11 sind eine Anfrage $Q$ und drei Dokumente $A, B, C$ aus der Architekturdomäne dargestellt. Als Menge von Relationen $\mathcal{R}$ wurden die neun Kreuzungsrelationen aus Abschnitt 2.2.1 und eine Relation $R_{gl}$ zur Längengleichheit verwendet, die in Abschnitt 2.2.3 vorgestellt wurde, d.h. $\mathcal{R} = \mathcal{R}_{Kreuzungen} \cup \{R_{gl}\}$. Die Baupläne wurden schon im Zusammenhang mit Abbildung 2.7 besprochen, allerdings ohne die Berücksichtigung von Längengleichheiten.

Auf der rechten Seite von Abbildung 2.11 sind die den Dokumenten entsprechenden Graphen dargestellt. Für ein Dokument $X \in \{Q, A, B, C\}$ bezeichnet $G_X := G(X, \mathcal{R}) = (X, \zeta_x, E_X, \xi_X)$ den zu $X$ gehörigen Graphen bzgl. $\mathcal{R}$, der sich jeweils aus Abbildung 2.11 ergibt. Im folgenden werden die einzelnen Dokumente bzgl. der Anfrage $Q$ besprochen, wobei auch $Q$ als Dokument der Datenbasis angenommen wird und somit natürlich immer einen Treffer darstellt.

Ist $Q$ gleichzeitig Anfrage und Dokument der Datenbasis, ergibt sich eine bijektive Abbildung $\phi : Q \to Q$ mit $\phi = id$. Da $G_Q = G_Q \downarrow \phi(Q)$ gilt $G_Q \simeq G_Q \downarrow \phi(Q)$ und $\nu(\phi(Q)) = \phi(Q)$. Da $G_Q \preceq G_Q$ ist $Q$ als Datenbasisdokument bzgl. der Anfrage $Q$ eine Treffer für jeden der vier Trefferbegriffe.

Für Dokument $A$ lassen sich nur injektive Abbildungen $\phi : Q \to A$ konstruieren. Wählt man $\phi$ als Identität gilt wiederum $G_Q \simeq G_A \downarrow \phi(Q)$ und $A$ ist I-Treffer. Allerdings ist $\nu(\phi(Q)) = A \setminus \{f\} \neq A = \phi(Q)$. Somit stellt $A$ weder einen IC-Treffer noch einen SC-Treffer dar. Wählt man $\psi : E_Q \to E_A$ als identische Abbildung, so ist $G'_A = (A \setminus \{f\}, \zeta \downarrow \phi(Q), E_A \setminus \{(a,f)\}, \xi \downarrow \psi(E_Q))$ subisomorph zu $G_Q$. Dokument $A$ ist also S-Treffer und I-Treffer bzgl. Anfrage $Q$.

Im Gegensatz zu den beiden vorherigen Dokumenten ist $G_B$ nicht isomorph zu $G_Q$, da $\Phi_{E_Q} := \phi \times \phi \downarrow E_Q$ mit $\phi = id$ wegen der Kante $(b,d) \in E_B$ keine Bijektion zwischen $E_Q$ und $E_B$ ist. Weiterhin gilt $\xi_Q((c,d)) = \{\vdash\} \neq \{\vdash, gl\} = \xi_A((c,d))$. Da $\xi_Q((c,d)) \leq \xi_A((c,d))$ gilt, ist $G_Q$ subisomorph zu $G'_A = (A, \zeta_A, E_A \setminus \{(b,d)\}, \xi \downarrow E_A \setminus \{(b,d)\})$. Wegen $\nu(\phi(Q)) = \phi(B)$ bildet Dokument $B$ einen SC-Treffer und S-Treffer bzgl. der Anfrage $Q$.

Bei Dokument $C$ gilt wie bei Dokument $A$ bzgl. der Nachbarschaft $\nu(\phi(Q)) \neq \phi(C)$. Ansonsten besitzt Dokument $C$ die gleichen Eigenschaften wie Dokument $B$, womit $C$ ein S-Treffer bzgl. $Q$ darstellt.

Die Ergebnisse stimmen mit den in Abschnitt 2.2.2 dargestellten Treffern überein (vergleiche Abbildung 2.7). ∘

**Bemerkung 2.4.12.** Satz 2.4.10 hat gezeigt, daß sich die Bezeichnung "Treffer" zwischen einer Anfrage $Q$ und einem Dokument $D$ auch auf Graphen $G_Q$ und $G_D$ anwenden lassen, d.h. $G_D$ ist genau dann $T$-Treffer bzgl. $G_Q$, wenn $D$ ein $T$-Treffer bzgl. $Q$ ist für alle Trefferarten $T \in \{IC, I, SC, S\}$.

Daher wird später einfach der Graph $G_Q$ als Anfrage und der Graph $G_D$ als Dokument bezeichnet.

**Fazit** In diesem Abschnitt wurde gezeigt, wie die Trefferbegriffe, die auf Dokumenten definiert wurden, sich auf beschriftete Graphen übertragen lassen. Bei Graphen lassen sich Treffer durch Graphenisomorphie bzw. Subgraphenisomorphie ausdrücken. Man beachte, daß die in Abschnitt 2.2.2 aufgestellte Hierarchie der Trefferbegriffe auch für Graphen gilt.

### 2.4.4 Komplexität der Isomorphieprobleme

Im letzten Abschnitt wurde demonstriert, wie sich diverse Probleme der Trefferberechnung durch Probleme der Graphentheorie ausdrücken lassen. In diesem Abschnitt wird die Zeitkomplexität der Trefferberechnung mittels Graphen untersucht. Dabei wird erst auf das allgemeine Isomorphieproblem aus der Graphentheorie eingegangen. Weiterhin werden Unterschiede zwischen dem allgemeinen und dem bei der Trefferberechnung vorliegenden Isomorphieproblem aufgezeigt.

Das allgemeine Isomorphieproblem auf Basis von Definition 2.4.4 lautet: Entscheide, ob zwei gegebene Graphen isomorph zueinander sind. Bisher ist nicht bewiesen, ob dieses Problem in der Komplexitätsklasse P liegt [38]. Für planare Graphen läßt sich das Problem in linearer Zeit lösen [21]. Für Graphen, deren Knotengrad begrenzt ist, existieren Verfahren mit polynomieller Laufzeit [15].

Für die in dieser Arbeit verwendeten Form der Subgraphenisomorphie gilt die gleiche Komplexität wie für das allgemeine Isomorphieproblem. In der Literatur versteht man unter dem allgemeinen Subgraphenisomorphieproblem "Entscheide, ob für zwei Graphen $G, G'$ ein Subgraph von $G'$ existiert, der subisomorph zu $G$ ist (siehe z.B. [12]). Dieses Entscheidungsproblem ist NP-vollständig [46]. Hierbei ist die Angabe aller Subgraphen $G''$ nicht berücksichtigt. Möchte man alle Treffer auflisten oder wissen, wie viele Treffer existieren, ist mit längeren Laufzeiten zu rechnen, da exponentiell viele Treffer existieren können. An dieser Stelle sei auf das Zählproblem verwiesen [45]. In der Praxis wird die Anzahl der Treffer pro Dokument i.d.R. automatisch oder durch eine Benutzerangabe beschränkt, um einerseits die Trefferpräsentation übersichtlich zu gestalten und andererseits die Berechnungsdauer einzuschränken.

Die beiden aufgeführten allgemeinen Problemstellungen gehen davon aus, daß der Vergleich zwischen zwei Graphen stattfindet, die erst zum Anfragezeitpunkt vorliegen. Etwaige Vorverarbeitungsschritte gelten als Teil der Berechnung. Für die Zeitabschätzung wird von Graphen beliebiger Größe ausgegangen.

Die in dieser Arbeit vorgestellten Trefferberechnungen gestalten sich anders. Der Vergleich findet nicht zwischen zwei Graphen, sondern zwischen einem Anfragegraphen und einer Menge von Graphen, der Datenbasis, statt. Die Datenbasis steht zum Anfragezeitpunkt fest. Eine Vorverarbeitung der Datenbasis ist erlaubt. Sie wird nicht bei der Zeitkomplexität der Anfragebeantwortung einbezogen. Hier liegt ein großer Unterschied zu den allgemeinen Isomorphieproblemen.

Zusätzlich wird davon ausgegangen, daß die Anfrage wesentlich kleiner als die Dokumente der Datenbasis sind. Es wird also ein kleiner Graph mit vielen großen verglichen.

Weiterhin wird beim allgemeinen Isomorphieproblem davon ausgegangen, daß beide Graphen dem Vergleichsverfahren direkt zur Verfügung stehen. Dies trifft für eine Datenbank aus Dokumenten bzw. Graphen nicht zu, da die Daten auf einem externen Speichermedium liegen. Hier müssen diese erst in den Hauptspeicher transferiert werden, um dort miteinander verglichen werden zu können. Diese Transferzeiten bilden in der Praxis einen wesentlichen Anteil der Beantwortungszeit [50].

**Fazit** Die Trefferberechnungen basieren auf NP-vollständigen Problemen. Allerdings liegen bei der Trefferberechnung und bei den allgemeinen Isomorphieproblemen unterschiedliche Voraussetzungen vor. Allein schon die Verwendung einer Indexierung ermöglicht einen gezielten Zugriff auf bestimmte Knoten, wobei der Index vorab berechnet werden kann und zum Anfragezeitpunkt zur Verfügung steht. Auch die Annahme, daß Anfragegraphen wesentlich kleiner als die Graphen der Datenbasis sind, unterscheidet sich von den allgemeinen Isomorphieproblemen.

Anstatt für eine Datenbasis $\mathcal{D}$ von Graphen das Isomorphieproblem $|\mathcal{D}|$ mal zu lösen, d.h. getrennt für jedes einzelne Dokument, wird in dieser Arbeit ein Verfahren verwendet, daß alle Graphen gleichzeitig zur Bestimmung der Treffer mit einbezieht.

### 2.4.5 Diskussion der praktischen Anwendbarkeit

Der letzte Abschnitt hat gezeigt, daß die theoretische Zeitkomplexität zur Berechnung der Treffer hoch ist. Trotzdem sprechen einige Argumente für eine praktische Anwendbarkeit. Diese werden im folgenden vorgestellt.

Oft liegt es in der Natur der Daten, daß sich bestimmte Konstellationen besonders auszeichnen. In dieser Arbeit wird davon ausgegangen, daß sich solche Konstellationen durch die in $\mathcal{R}$ angegebenen Beziehungen zwischen Objekten beschreiben lassen. Dies spiegelt sich i.d.R. in "prägnanten" Knoten- bzw. Kantenbeschriftungen wider, die nur selten in Dokumenten vorkommen. Diese bilden einen idealen Startpunkt für eine weiterführende Suche [12].

Liefert eine Anfrage eine (zu) große Treffermenge, so wird diese Anfrage meistens weiter verfeinert, um so die Treffermenge übersichtlicher zu gestalten. I.d.R. wird davon ausgegangen, daß nur wenige Treffer das Ergebnis einer Anfrage darstellen. Oft bildet dies die Grundlage von guten Heuristiken, die in gewisser Weise auch von den in Kapitel 4 vorgestellten Suchverfahren angewendet werden.

In der Praxis tritt oft der Fall auf, daß bei den verwendeten Graphen der Knotengrad beschränkt ist. Dies trifft z.B. auf die Anwendung aus der Architekturdomäne (siehe Abschnitt 2.2) zu, wo ein Objekt nur mit wenigen anderen Objekten verbunden ist. Hierfür existieren Verfahren mit polynomieller Laufzeit zur Berechnung aller zu einer Anfrage isomorphen Subgraphen [15].

Wie Abschnitt 2.6 zeigen wird, werden Verfahren zur Lösung von Isomorphieproblemen in anderen Anwendungsdomänen praktisch eingesetzt. Dies sollte als Motivation angesehen werden, trotz der möglicherweise hohen theoretischen Zeitkomplexität, die hier vorgeschlagenen Trefferbegriffe in der Praxis zu untersuchen. Wie sich zeigen wird, läßt sich i.A. durch die bereits angesprochene Möglichkeit der Vorverarbeitung der Datenbasisdokumente eine geringe Beantwortungszeit von Anfragen erzielen.

In der Praxis werden zur Berechnung von Graphenisomorphien bewußt Verfahren eingesetzt, die im schlechtesten Fall nicht polynomielles Laufzeitverhalten haben. Dies geschieht selbst in Spezialfällen, für die das Graphenisomorphieproblem in Polynomialzeit lösbar ist, wie z.B. im Fall planarer Graphen, für die das Problem in Linearzeit lösbar ist. Gründe hierfür sind hohe konstante Faktoren bei den linearen Algorithmen [21], [30].

## 2.5 Vergleich der Trefferdefinitionen

In Abschnitt 2.1.2 wurden Treffer mittels $G$-Mengen, sogenannte $G$-Treffer, definiert. Diese Art von Treffern lassen sich auch als verallgemeinerte Form von I-Treffern ausdrücken. Hierauf wird in Abschnitt 2.5.1 eingegangen. Im darauffolgenden Abschnitt werden Erweiterungen der Darstellung durch Graphen motiviert.

### 2.5.1 $G$-Treffer und I-Treffer

Die in Abschnitt 2.1.2 beschriebenen $G$-Treffer lassen sich nicht direkt durch die in Abschnitt 2.3 vorgestellten Trefferbegriffe erfassen. Dafür ist folgende Verallgemeinerung von I-Treffern notwendig:

**Definition 2.5.1.** Dokument $D$ ist ein **verallgemeinerter I-Treffer** einer Anfrage $Q$ bzgl. $\mathcal{R}$, wenn eine injektive Abbildung $\Psi : Q \to D$ und eine Bijektion $\pi \in S_m$ existiert mit:

$$\forall \gamma \in [1 : |\mathcal{R}|] \forall q, q' \in Q : (q, q') \in R_\gamma \Leftrightarrow (\Psi(q), \Psi(q') \in R_{\pi(\gamma)}.$$

Die Äquivalenz zwischen $G$-Treffern und verallgemeinerten I-Treffern ergibt sich wie folgt: Für eine $G$-Menge $\mathcal{O}$ und ein $g \in G$ definiere

$$R_g := \{(a, ga) | a \in \mathcal{O}\}$$

und $\mathcal{R} := \{R_g | g \in G\}$.

Ein $G$-Treffer $gQ \subseteq D$ ist dann ein verallgemeinerter I-Treffer. Dies bedeutet, daß eine injektive Abbildung $\Psi_g : Q \to D, q \mapsto gq$ und eine Bijektion $\pi : x \mapsto gxg^{-1}$ existiert mit:

$$\forall x \in G \; \forall q, q' \in Q : \; (q, q') \in R_x \Rightarrow (gq, gq') \in R_{gxg^{-1}}.$$

### 2.5.2 Kombination der Trefferarten

Die im letzten Abschnitt dargestellte Realisierung von $G$-Treffern durch I-Treffer ist für die Praxis unpraktikabel. I-Treffer stellen im Vergleich zu $G$-Treffern ein viel allgemeineres Trefferkonzept dar. Der in Abschnitt 2.1.2 vorgestellte Ansatz gilt für die Berechnung von $G$-Treffern als sehr effizient [5].

Ein anderer Ansatz zur Integration von $G$-Treffern bietet sich bei den in Abschnitt 2.4 eingeführten beschrifteten Graphen. Hier lassen sich die Knotenbeschriftungen ausnutzen, um objektspezifische Informationen zu speichern. Knotenbeschriftungen wurden bei den bisherigen Trefferbesprechungen nicht verwendet. In der Graphendarstellung ist $W_V = \mathcal{O}$, d.h. der dem Objekt $o \in D$ entsprechende Knoten $v_o$ erhält als Knotenbeschriftung $\zeta(v_o) = o$.

Die Isomorphie zwischen Graphen kann an $G$-Treffer angepaßt werden, indem man für die Übereinstimmung der Knotenbeschriftungen, statt $\zeta_Q(v) = \zeta_D(\phi(v))$, $\forall v \in Q$, wie in Definition 2.4.4 nun $\exists g \in G : \zeta_Q(v) = g\zeta_D(\phi(v))$, $\forall v \in Q$, fordert.

Durch die Verwendung von Knotenbeschriftungen läßt sich auch eine Fokussierung für die Architekturanwendung realisieren. Setzt sich etwa die Menge der Objekte aus Zu- und Abluftleitungen zusammen, so läßt sich für jedes Objekt in der Knotenbeschriftung erfassen, ob es eine Zu- oder Abluftleitung darstellt. Führt man weiterhin eine Knotenbeschriftung "egal" ein und definiert $W_V = \{Abluft, Zuluft, egal\}$ mit $egal \leq Abluft$ und $egal \leq Zuluft$, so läßt sich durch die Knotenbeschriftung der Anfrage und die Angabe der Trefferart, I-Treffer oder S-Treffer, entscheiden, ob im Treffer die Anfrageobjekte auf Zuluft, Abluft oder beliebigen Objekten der Dokumente abgebildet werden können.

Man sieht also, daß sich Knotenbeschriftungen vielseitig nutzen lassen. Der Schwerpunkt dieser Arbeit liegt aber bei den in Abschnitt 2.3 definierten Trefferbegriffen. Die später beschriebenen Verfahren zur Trefferberechnung lassen sich leicht erweitern, so daß Knotenbeschriftungen mitberücksichtigt werden.

## 2.6 Weitere Anwendungsgebiete

Bisher wurden die in Abschnitt 2.3 definierten Trefferbegriffe für die Suche in Bauplänen aus der Architekturdomäne verwendet. In diesem Abschnitt wird exemplarisch gezeigt, daß sich die vorgestellten Trefferbegriffe direkt in anderen Anwendungsgebieten verwenden lassen.

### 2.6.1 Geoinformatik

Ein Aspekt der Geoinformatik ist die Suche in Landkarten [2]. In stark abstrahierter Form sind sie mit Bauplänen vergleichbar. Straßen lassen sich als Linienzüge darstellen. Eine Konstellation von mehreren Straßen wird wesentlich durch die Straßenkreuzungen charakterisiert. Dabei existieren mehrere Typen von Straßenkreuzungen, z.B. T-Kreuzung, Y-Kreuzung, Kreisverkehr. Hier sind die Parallelen zur vorgestellten Suche in Bauplänen sehr deutlich. Sieht man jede Straße als ein Objekt an und erstellt für jeden Kreuzungstyp eine Relation, so lassen sich damit Treffer finden, die skalierungsunabhängig sind. Wie bei Bauplänen wird eine Konstellation, also eine Menge von sich kreuzenden Straßen, durch ihre Kreuzungstypen beschrieben. Somit läßt sich eine Suche in großen Straßenkarten- und Stadtplanarchiven realisieren. Die Unterscheidung der einzelnen Trefferbegriffe erfolgt analog zum Architekturbeispiel aus Abbildung 2.5.

Auch in anderen Teilgebieten der Geoinformatik läßt sich das hier beschriebene Trefferkonzept anwenden. Etwa wenn Konstellationen aus Grundstücken und Gebäuden bestehen und solche Konstellationen in einer großen Datenbank aus Landkarten gesucht werden sollen. Zur Beschreibung

solcher Konstellationen können z.B. Beziehungen wie “grenzt an” oder “ist (etwa) gleich groß wie” verwendet werden.

### 2.6.2 Chemische Strukturformeln

In der organischen Chemie wächst die Zahl an neuen Verbindungen täglich. Oft muß überprüft werden, ob diese Verbindung schon entdeckt wurde. Dies geschieht meistens durch eine Suche in Datenbanken mit chemischen Strukturformeln. Bei der Vorhersage von Reaktionen gilt ein ähnliches Szenario. Auch hier wird mittels Strukturformeln in Datenbanken gesucht.

Im Kontext dieser Aufgabenstellung, der Suche in großen Datenbanken mit Strukturformeln, existieren einige Arbeiten (z.B. [7], [49] und [29]). Als Datenstruktur zur Darstellung von Strukturformeln werden i.d.R. ungerichtete gewichtete Graphen verwendet. Knoten sind Atome und Kanten sind die Bindungen zwischen den entsprechenden Atomen.

Treffer einer Anfrage sind Graphen der Datenbank, die isomorph zum Graphen der Anfrage sind. Dies entspricht einem IC-Treffer aus Abschnitt 2.3.

Da Knoten und Kantenanzahl bei einer Isomorphie zwischen Anfrage- und Treffergraphen übereinstimmen, wird ein kompletter Graph durch eine eindeutige Zahl kodiert [29]. Die eigentliche Arbeit steckt im aufwendigen Kodierungsschritt. Die Suche auf den Kodierungen kann z.B. durch eine binäre Suche erledigt werden.

Vergleichbar mit S-Treffern ist die Suche nach Substrukturen. Hierfür werden in der Literatur nur Verfahren angegeben, die zwei Graphen miteinander vergleichen. Ein Suche in großen Molekül-Datenbanken erfolgt in zwei Schritten: erst wird die Anfrage in stark vereinfachter bzw. vergröberter Form mit ebenfalls stark vergröberten Datenbankeinträgen verglichen. Dadurch werden viele Datanbankeinträge für den zweiten Suchschritt eliminiert. In diesem wird die Anfrage detailliert mit den übriggebliebenen Datenbankeinträgen sequentiell verglichen [7], [49]. In Abschnitt 3.1.2 wird auf diese Verfahren näher eingegangen.

Für die Anwendung im Kontext dieser Arbeit bietet sich eine detaillierte Darstellung der Strukturformeln an. Ein Objekt ist ein Atom, d.h. $\mathcal{O}$ ist die Menge aller Atome. Eine Bindung zwischen zwei Atomen läßt sich im wesentlichen durch die Bindungsordnung $p$ , den Bindungswinkel $\alpha$ und die Bindungslänge $d$ beschreiben [49]. Für jede mögliche Kombination dieser drei Eigenschaften wird eine eigene Relation $R_{(p,\alpha,d)}$ definiert. Zwei Objekte stehen in solch einer Relation, wenn sie mit der entsprechenden Bindung verbunden sind. Für die Suche nach Substrukturen lassen sich die Trefferbegriffe S-Treffer und SC-Treffer bzw. die in Kapitel 4 beschriebenen Suchverfahren direkt anwenden.

### 2.6.3 Wissensrepräsentation und Bilddatenbanken

Im Bereich der künstlichen Intelligenz befinden sich viele Gebiete, in denen sich die Trefferbegriffe zur Suche in Datenbanken anwenden lassen (z.B. [16]). Stellvertretend sei hier das Gebiet der Wissensrepräsentation kurz vorgestellt. Eine Möglichkeit der Beschreibung von Szenen aus der realen Welt ist die Extraktion von Objekten. Viele Objekte setzen sich aus kleineren Objekten zusammen. Ein Stuhl besteht zum Beispiel aus vier Stuhlbeinen, einer Sitzfläche und einer Lehne. Bei der Beschreibung ist das Zusammenspiel der Objekte wichtig. Dieses wird durch Prädikate (Funktionen, Rollen) [16] beschrieben, etwa “steht auf”, “ist angeschraubt an” oder “hängt an”. Diese Prädikate lassen sich i.d.R. als zweistellige Relationen auf der Menge aller Objekte auffassen. Damit sind die für diese Arbeit verwendeten Mengen $\mathcal{O}$ und $\mathcal{R}$ definiert. Durch die Auswahl einer der vier Trefferbegriffe wird die Suche präzisiert: Für genaue Übereinstimmung werden I-Treffer verwendet. Mittels S-Treffer lassen sich bei der obigen Beschreibung eines Stuhls z.B. auch solche erkennen, die Querstreben zwischen den Stuhlbeinen besitzen.

In vielen Fällen wird das Wissen durch Ausdrücke formuliert und gespeichert, die auf der Prädikatenlogik basieren [3]. Teilweise werden die Informationen auch direkt in beschriftete Graphen gespeichert [12]. In beiden Szenarien wird davon ausgegangen, daß bei der Suche alle Daten im Hauptspeicher vorliegen. Auf die Problematiken, die sich bei großen Datenbanken, wo der externe Datenzugriff gering gehalten werden soll, wird nicht eingegangen.

Bei der inhaltsbasierten Suche in Bilddatenbanken gilt teilweise ein ähnliches Szenario. In Bildern erkannte Objekte werden in Relation zueinander gesetzt, z.B. Sonne “über” Palme oder Buch “auf” Tisch. Hier kann die Konstellationssuche ebenfalls angewendet werden, wenn in der Anfrage die Relationen zwischen den Objekten angegeben wird.

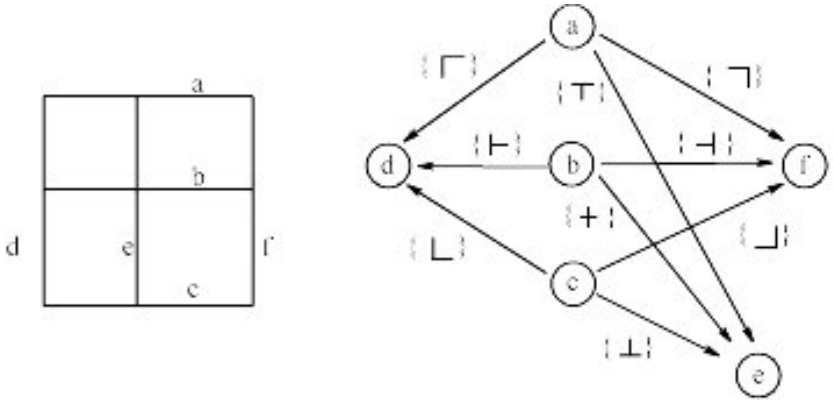



Abbildung 2.10: Links ein Bauplan $D$, rechts der Graph $G(D, \mathcal{R}_{Kreuzungen})$.

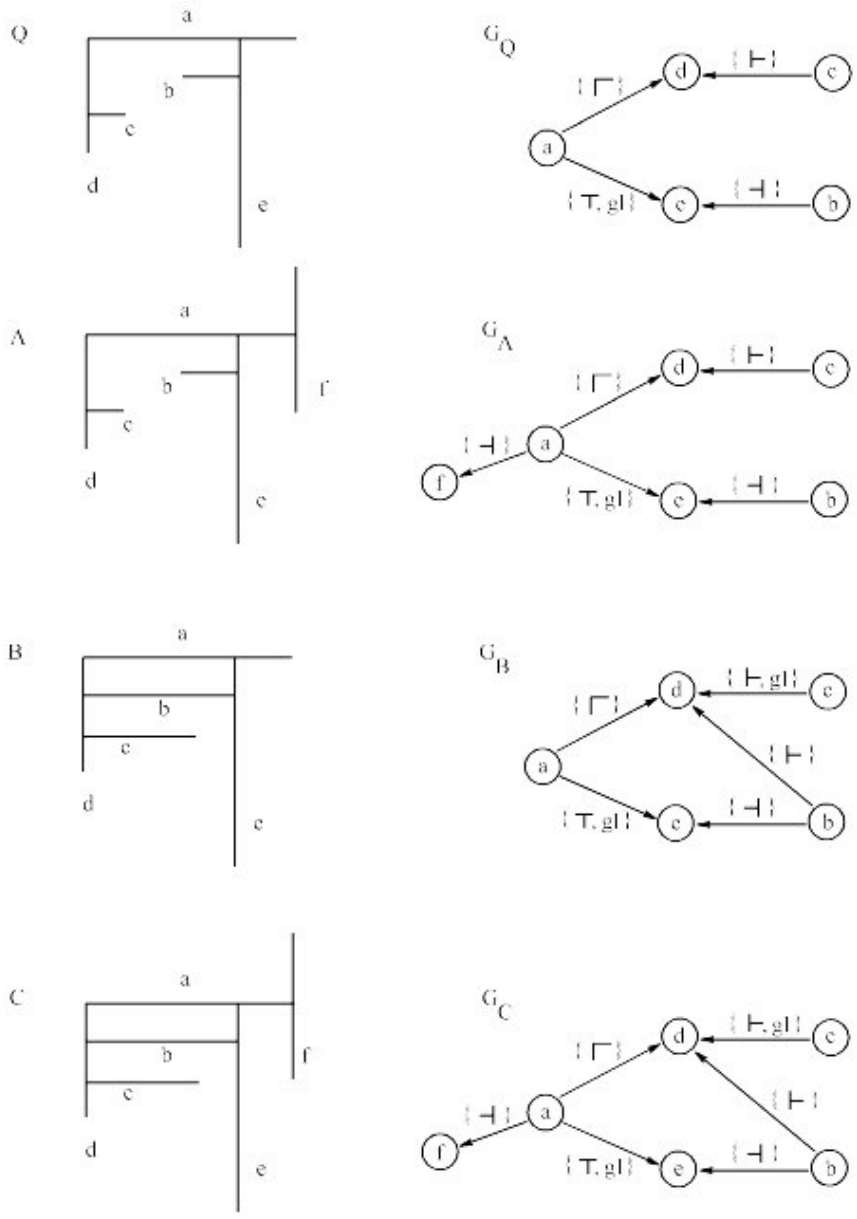



Abbildung 2.11: Auf der linken Seite sind 4 Dokumente und auf der rechten Seite die entsprechenden Graphen dargestellt. Objekte und Knoten sind mit Kleinbuchstaben beschriftet. Knotenbeschriftungen sind nicht angegeben. Eine Kantenbeschriftung $\xi(e)$ ist eine Teilmenge von $\mathcal{R}$, also ein Menge von Relationen. Neben den neun Kreuzungsrelationen wird auch eine Relation $R_{gl}$ bzgl. der Längengleichheit von Objekten verwendet.

# Kapitel 3

# Trefferberechnung

Nachdem im vorherigen Kapitel durch die vier Trefferarten definiert wurde, "was" gesucht wird, geht es in diesem Kapitel um das Problem "wie" effizient gesucht werden kann.

Das Kapitel ist so aufgebaut, das erst die Rahmenbedingungen der Suche, also das Ausgangsszenario vorgestellt wird. Anschließend werden bekannte Ansätze zur Lösung von Graphenisomorphieproblemen beschrieben, die sich bedingt zur Trefferberechnung eignen. Danach wird auf das allgemeine Konzept der invertierten Listen eingegangen. Die letzten beiden Abschnitte bilden den Kern dieser Arbeit. Erst wird die Suche nach Treffern unter Verwendung von invertierten Listen theoretisch beschrieben, bevor vier Trefferberechnungsverfahren vorgestellt werden, die auf den theoretischen Ergebnissen aufbauen. Im nächsten Kapitel werden die Suchverfahren auf ihre Praxistauglichkeit hin untersucht.

## 3.1 Einleitung

Als Vorbereitung auf die in den nächsten Abschnitten vorgestellten Verfahren zur Trefferberechnung werden in diesem Abschnitt bereits bekannte Ansätze beschrieben (Abschnitt 3.1.2). Dabei wird auch auf die unterschiedlichen Voraussetzungen bzw. Anwendungsaspekte, die beim Algorithmenentwurf im Vordergrund standen, eingegangen. Wie sich herausstellen wird, unterscheidet sich das Ausgangsszenario bei den bisherigen Ansätzen von dem in dieser Arbeit betrachteten Ansatz. Auf das Ausgangsszenario dieser Arbeit wird in Abschnitt 3.1.1 eingegangen.

### 3.1.1 Ausgangsszenario

Alle in dieser Arbeit besprochenen Suchverfahren sollen auf große Datenbestände angewendet werden. Diese sind auf einem externem Medium (Massenspeicher, Sekundärmedium) gespeichert. Es wird davon ausgegangen, daß die Größe der Datenbasis die Größe des Hauptspeichers um ein Vielfaches übertrifft. Typische Werte für das Größenverhältnis zwischen Hauptspeicher und Datenbasis sind z.Z. etwa 512 MB Hauptspeicher gegenüber einem Datenvolumen von mehreren Hundert Gigabyte.

Nach [50] werden für extern gespeicherte Datenbasen besondere Verfahren zur Suche in den Daten benötigt. Begründet wird dies mit der benötigten Transferzeit, um Daten vom Sekundärmedium in den Hauptspeicher zu transferieren. Bei der Suche wird davon ausgegangen, daß der eigentliche Vergleich zwischen Anfrage- und Dokumentdaten im Hauptspeicher stattfindet. Beim Datentransfer ist weiterhin zu beachten, daß wahlfreie Zugriffe erheblich zeitintensiver als sequentielle Zugriffe sind.

In [50] wird die wahlfreie Zugriffszeit $t_s$ mit $10 \times 10^{-3}$ Sekunden und die Transferzeit $t_r$ für ein Byte mit $0.5 \times 10^{-6}$ Sekunden angegeben.

Für eine Datenbasis mit 10.000 Dokumenten und einem Datenvolumen von einem Gigabyte liegt die Gesamtdauer für den Datentransfer bei über 10 Minuten (ca. $10^9 \times t_r + 10^4 \times t_s = 600$

Sekunden). Allein schon die Zeit für den Datentransfer ist als Antwortzeit nicht akzeptabel.

Dieses Beispiel belegt, daß der Datentransfer einen wesentlichen Einfluß auf die Beantwortungszeit von Anfragen ausübt. Auf dieser Basis werden neue Suchverfahren vorgestellt, die einen möglichst geringen Datentransfer und eine geringe Anzahl an wahlfreien Zugriffen gewährleisten.

Weiterhin wird in dem hier betrachteten Anwendungsszenario davon ausgegangen, daß die Datenbasis vorverarbeitet werden kann (siehe auch [50]). Somit können Indexe oder andere Hilfsstrukturen vorab berechnet werden, mit deren Hilfe zum Anfragezeitpunkt der Suchraum stark eingeschränkt wird.

Bei der Verwendung von solchen Zusatzstrukturen sind der dafür benötigte zusätzliche Speicherplatz und die Berechnungszeit zu berücksichtigen. Bei der Berechnungszeit wird zwischen einer neuen Erstellung der Hilfsstruktur und einer Anpassung einer bereits berechneten Hilfsstruktur an Änderungen der Datenbasis unterschieden.

In [50] werden mehrere Indexstrukturen untersucht. Im Abschnitt 3.2 wird die Indexierung mit invertierten Listen (inverted file index) vorgestellt, auf die die Suchalgorithmen aus Abschnitt 4 aufbauen.

### 3.1.2 Bekannte Ansätze

Abschnitt 2.4 hat gezeigt, daß sich die Trefferbegriffe aus Abschnitt 2.3 durch Probleme der Graphenisomorphie ausdrücken lassen.

Für den Vergleich zweier Graphen auf (Sub-)Graphenisomorphie lassen sich einfache Varianten der Breiten- oder Tiefensuche einsetzen [44]. In Kombination mit Heuristiken lassen sich so in vielen Fällen Subgraphenisomorphien schnell bestimmen [49].

Abschnitt 3.1.1 hat gezeigt, daß ein kompletter Durchlauf aller Dokumente einer großen Datenbank aus Graphen, um die Dokumente sequentiell z.B. mittels Breiten- oder Tiefensuche mit einem Anfragegraphen zu vergleichen, allein schon wegen der Transferzeiten sehr zeitintensiv ist. Hier werden andere Ansätze benötigt.

Verfahren zur Lösung von Graphenisomorphieproblemen werden z.B. bei Konzeptgraphen oder in der Chemie eingesetzt, wo u.a. in Molekülдatenbanken gesucht wird. Hierauf wurde bereits in Abschnitt 2.6 eingegangen.

Die verwendeten Ansätze für eine Suche in großen Datenbanken lassen sich grob in zwei Strategien unterteilen:

1. Filterung: Die Reduktion des Suchraums durch grobe Vergleiche, um so große Teile des Datenbestandes, die bzgl. der Anfrage auf keinen Fall einen Treffer darstellen können, von einem zweiten, detaillierten Vergleichsschritt ausschließen zu können.

2. Kodierung: Anfragen und alle Graphen der Datenbasis werden kodiert. Die Suche findet auf den Kodierungen statt.

Diese Strategien werden im folgenden genauer vorgestellt. Wie schon erwähnt, steht die Berechnung von Isomorphien und Subisomorphien zwischen einem Anfragegraphen und einer Menge von Graphen, der Datenbasis, im Vordergrund.

**Filterung** Diese Suchstrategie unterteilt sich in zwei Schritte. Im ersten Schritt wird die Anfrage vergröbert und mit ebenfalls vergröberten Dokumenten der Datenbank verglichen. Dabei sind die Vergröberungen der Dokumente persistent gespeichert. Weiterhin werden schnelle, meist approximative, Vergleichsverfahren angewendet. Nur Dokumente, deren Vergröberung mit der vergröberten Anfrage übereinstimmen (Isomorphie) bzw. in deren Vergröberung die vergröberte Anfrage enthalten ist (Subisomorphie), werden im zweiten Suchschritt auf detaillierte Übereinstimmung zur Anfrage untersucht. Dabei wird davon ausgegangen, daß die groben Vergleiche im ersten Schritt um ein Vielfaches schneller zu berechnen sind als die detaillierten Berechnungen, d.h. (Sub-)Graphenisomorphien, im zweiten Schritt. Ziel dieser unterteilten Suche ist die Reduktion des Suchraumes durch den ersten Schritt, um so viele Dokumente für den zweiten, i.d.R.

zeitintensiveren, Schritt ausschließen zu können, die bzgl. der Anfrage keinen Treffer darstellen können.

Für diese Suchstrategie existieren viele Varianten, die oft an anwendungsspezifische Eigenschaften angepaßt werden. Die Suche in Datenbanken aus chemischen Strukturformeln, dargestellt in Form von beschrifteten Graphen, wurde in Abschnitt 2.6.2 motiviert. Eine Möglichkeit der Vergröberung dieser Graphen ist die Unterscheidung innerhalb der Strukturformeln in Ringe und Ketten, die jeweils durch einen Knoten repräsentiert werden. Hierdurch ergeben sich erheblich kleinere Graphen. Anfrage und Dokumente lassen sich in dieser vergröberten Form schneller miteinander vergleichen [7].

Eine andere Vergröberungsform erhält man, indem sukzessive alle Knoten mit Grad 2 eliminiert werden, wobei deren Nachbarknoten durch eine Kante verbunden werden (homeomorphic reduction) [7]. Ein Beispiel ist in Abbildung 3.1 dargestellt.

In den vorherigen Beispielen wurde für jedes Dokument der Datenbank eine Vergröberung gespeichert. Teilweise lassen sich Vergröberungen berechnen, die jeweils als Repräsentant für mehrere Dokumente stehen. In [11] wird eine Vergröberung von beschrifteten Graphen bzw. Konzeptgraphen zu sogenannten “skeleton” Graphen beschrieben, indem alle Beschriftungen ausgeblendet werden und nur eine sehr grobe Struktur (Skelett) eines Graphen verwendet wird. Bei der Suche steht ein solches Skelett für mehrere Graphen, die dieses Skelett enthalten. Dadurch reduziert sich die Anzahl der Vergleiche im ersten Suchschritt, da nicht mehr für jedes Dokument eine eigene Vergröberung abgespeichert wird.

**Kodierung** Bei dieser Strategie werden die Graphen nicht direkt miteinander verglichen. Statt dessen werden Kodierungen von Graphen berechnet und diese miteinander verglichen. Die Kodierungen aller Graphen der Datenbank werden vorab berechnet und gespeichert. Eine Anfrage wird ebenfalls kodiert und dann mit den persistenten Kodierungen verglichen. Dabei wird von der Annahme ausgegangen, daß der Vergleich von Kodierungen effizienter ist als die Berechnungen von (Sub-)Graphenisomorphien.

Kodierungen von Graphen lassen sich in zwei Klassen unterteilen. Für die erste gilt, daß zwei Graphen genau dann isomorph zueinander sind, wenn die Kodierungen der Graphen gleich sind. Es besteht also eine Äquivalenz zwischen Gleichheit von Kodierungen und Graphenisomorphie. Bei der zweiten Klasse gilt nur eine Inklusion: Sind zwei Graphen isomorph zueinander, dann stimmen auch die Kodierungen überein. Eine Gleichheit der Kodierung besagt hier nur, daß die Graphen isomorph sein können, aber nicht müssen. Hier muß die Isomorphie noch nachgeprüft werden. Andererseits sind Graphen, deren Kodierungen verschieden sind, auf jeden Fall nicht isomorph. Solche Kodierungen können auch als Filterung angesehen werden, worauf z.B. in [11] eingegangen wird.

Eine Kodierung, die zur ersten Klasse gehört, erzeugt das Verfahren “nauty” [29]. Für einen gegebenen beschrifteten Graphen berechnet “nauty” einen kanonischen Graphen unter Berücksichtigung aller Automorphismen. Der kanonische Graph wird dann kodiert. Isomorphe Graphen besitzen gleiche kanonische Graphen und somit auch gleiche Kodierungen [30].

Die Berechnung einer solchen Kodierung benötigt im schlechtesten Fall mehr als polynomielle Laufzeit. Das Verfahren konnte bisher nicht zur Bestimmung von Subgraphenisomorphien erweitert werden.

Zur zweiten Klasse zählt ein Verfahren aus [11]. Hier werden für einen (ungerichteten) Graphen die Knoten untersucht. Für jeden Knoten wird der Grad berechnet. Die Folge aller Knotengrade wird sortiert und die sortierte Folge als Kodierung abgespeichert. Beispiele dieser Kodierung sind in Abbildung 3.2 dargestellt. Mit zunehmender Größe von Graphen wird die Kodierung ungenauer, d.h. die Kodierung ist gleich, obwohl die Graphen nicht isomorph sind.

Nach [11] läßt sich dieser Ansatz erweitern, indem Graphen in kleine Subgraphen zerlegt werden, deren Kodierungen nach Knotengraden in eine Hierarchie eingeordnet werden und sich somit eine Kodierung der ersten Klasse ergibt. Allerdings steigt die Länge dieser Kodierung exponentiell zur Graphengröße an und ist somit für Anwendungen mit größeren Graphen ungeeignet.

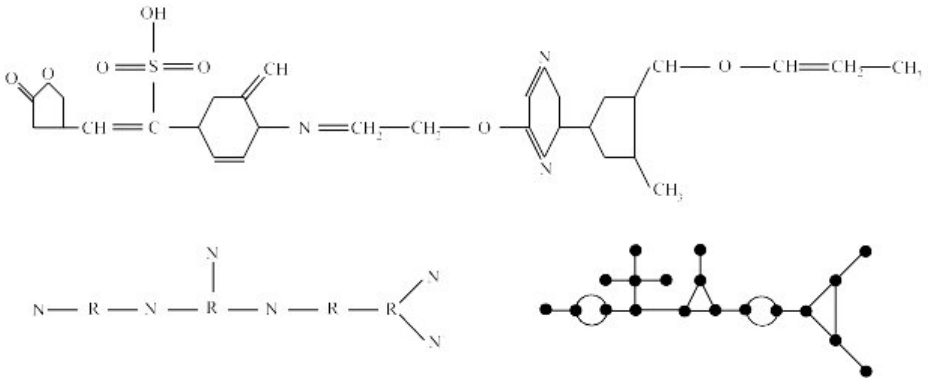


Abbildung 3.1: Eine Strukturformel (oben) zusammen mit zwei Vergröberungen aus [7]. Bei der linken Vergröberung wird nur noch zwischen Ringen (cyclic) und Nicht-Ringen (acyclic) unterschieden (Knotenbeschriftungen $N$ und $R$). Bei der rechten Vergröberung wurden alle Knoten mit Grad 2 entfernt.

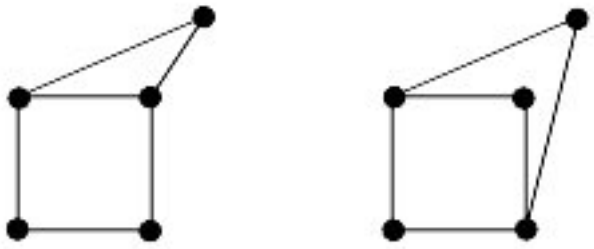


Abbildung 3.2: Zwei Graphen mit der Kodierung $2, 2, 2, 3, 3$ nach [11]. Sie ergibt sich aus der aufsteigend sortierten Folge der Knotengrade. Obwohl die Kodierung für beide Graphen gleich ist, sind die Graphen nicht isomorph.

Beide Kodierungsverfahren eigenen sich nach [11] zur Bestimmung von Subgraphenisomorphien. Laufzeiten werden weder für die Berechnung der Kodierung noch für den Vergleich von Kodierungen angegeben.

Filterung und Kodierung lassen sich kombinieren. Beispielsweise werden Graphen erst vergröbert und die Vergröberungen dann kodiert. Vergleiche finden erst zwischen Kodierungen statt. Falls diese erfolgreich waren, werden die entsprechenden Graphen verglichen. In [11] wird dieses Vorgehen ansatzweise besprochen.

### 3.1.3 Fazit

Das Verfahren "nauty" wird in vielen Anwendungen eingesetzt und gilt, da in praktischen Anwendungen der Worst Case zumeist nicht eintritt, als effizientes Verfahren zur Berechnung von Isomorphien zwischen Graphen. Es kann für die Berechnung von IC-Treffern, die in dieser Arbeit eingeführt wurden, verwendet werden. Allerdings eignet sich "nauty" nicht zur Berechnung von Subgraphenisomorphien. Es kann also nicht zur Berechnung der anderen drei Trefferarten aus Abschnitt 2.3 eingesetzt werden.

Die anderen vorgestellten Kodierungsverfahren eignen sich eher für kleine Graphen. Für größere Graphen gelten die Verfahren als unpraktikabel [11]. Weiterhin lassen sich die Verfahren nur bedingt zur Berechnung von Subgraphenisomorphien einsetzen.

Die Filterung wird in vielen Bereichen der Chemie eingesetzt. Hier werden die Filter (Screens) für eine Anwendung durch Experten bestimmt. Auch hier gilt die Berechnung von Subgraphenisomorphien, falls sie überhaupt möglich ist, als zeitintensiv.

Die Verfahren basieren auf dem Vergleich von zwei Graphen miteinander, d.h. die Datenbasis wird sequentiell durchlaufen und mit einer Anfrage verglichen. Dabei wird der Suchbereich durch Filterung bzw. Vergröberung eingeschränkt.

Bei allen Verfahren bleiben die speziellen Anforderungen an Datenbanken in Hinblick auf eine Minimierung von wahlfreien Zugriffen auf das Sekundärmedium und das Datentransfervolumen unberücksichtigt. Bei beiden Strategien ist davon auszugehen, daß alle Vergröberungen bzw. Kodierungen der Datenbankdokumente nicht in den Hauptspeicher passen. Spezielle Indexstrukturen werden hierfür nicht besprochen.

Mit keinem der bisherigen Verfahren lassen sich alle vier Trefferbegriffe effizient berechnen. Neue, speziell auf die Anforderungen an große Datenbanken aus Abschnitt 3.1.1 abgestimmte Verfahren werden benötigt.

## 3.2 Invertierte Listen

Dieser Abschnitt dient zur Motivation von invertierten Listen. Ursprünglich wurden sie als Indexstruktur für die Volltextsuche eingesetzt [50]. In [5] wurde dieser Ansatz verallgemeinert. Dies wird in den folgenden Abschnitten zusammengefaßt.

### 3.2.1 Das allgemeine Konzept

Invertierte Listen treten nicht nur in der Informatik auf. Das Schlagwortverzeichnis in Sachbüchern ist eine Form von invertierten Listen. Würde das Schlagwortverzeichnis eines Buches aus allen Wörtern bestehen, die in dem Buch vorkommen, erhält man einen Index, der für eine Volltextsuche verwendet werden könnte.

Im Zusammenhang mit invertierten Listen werden die Begriffe Vokabular und Treffermenge verwendet. Diese hängen von der jeweiligen Anwendung ab und werden im folgenden für einige bereits besprochene Anwendungen motiviert. Grob formuliert ist in diesem Zusammenhang das Vokabular eine Menge $\mathcal{V}$, wo jedem Element $v \in \mathcal{V}$ eine invertierte Liste $L(v)$ zugewiesen wird. Für die in diesem Abschnitt besprochenen Anwendungen ergibt sich das Vokabular aus den Objekten, die in mindestens einem Dokument einer Datenbasis vorkommen. Eine invertierte Liste $L_{\mathcal{D}}(w), w \in$

$\mathcal{V}$, besteht im wesentlichen aus einer Menge von Verweisen auf diejenigen Dokumente aus $\mathcal{D}$, in denen $w$ "vorkommt". Möglicherweise werden zusätzlich zu einem Dokumentverweis noch weitere Informationen über $w \in D_j$ gespeichert. Als eindeutiger Verweis auf ein Dokument $D_j \in \mathcal{D}$ wird einfach die Position $j$ innerhalb der Folge $(D_1, \ldots D_N) = \mathcal{D}$ verwendet. Die Treffermenge ergibt sich als die Schnittmenge mehrerer invertierter Listen, wobei i.d.R. jedes Anfrageobjekt einer Anfrage $Q$ eine invertierte Liste bestimmt, die Teil der Schnittmengenbildung ist.

Das vorgestellte Konzept wird im folgenden an einigen bereits vorgestellten Anwendungen verdeutlicht. Es wird später für die in Abschnitt 2.3 definierten Trefferbegriffe erweitert.

**Volltextsuche** Die Volltextsuche in Textsammlungen wurde in Abschnitt 2.1.1 beschrieben. Im folgenden gehen wird von der dort beschriebenen abstrakten Sichtweise aus. Sei $\mathcal{D} = (D_1, \ldots, D_N)$ eine Datenbasis aus $N$ Dokumenten.

Das Vokabular $\mathcal{V}$ besteht aus der Menge der Wörter, die mindestens in einem Dokument der Datenbasis vorkommen:

$$\mathcal{V}_\mathcal{D} := D_1 \cup \ldots \cup D_N.$$

Für jedes Wort $w \in \mathcal{V}$ werden in einer invertierten Liste $L_\mathcal{D}(w)$ die Referenzen auf alle Dokumente $D_j$ der Datenbasis $\mathcal{D}$ gespeichert, die $w$ enthalten:

$$L_\mathcal{D}(w) := \{j \in [1:N] | w \in D_j\}.$$

Hier wird der Vergleich zu einem Schlagwortverzeichnis deutlich, wo zu einem Schlagwort Seitenzahlen angegeben werden, d.h. Referenzen auf die Seiten eines Buches, auf denen das Schlagwort steht.

Die Treffermenge $\mathcal{M}_\mathcal{D}(Q)$ einer Anfrage $Q = \{w_1, \ldots, w_m\}$ berechnet sich aus den invertierten Listen durch

$$\mathcal{M}_\mathcal{D}(Q) = \cap_{k=1}^{m} L_\mathcal{D}(w_k). \tag{3.1}$$

Es ist offensichtlich, daß $\mathcal{M}_\mathcal{D}(Q)$ genau diejenigen Referenzen $j$ auf Dokumente der Datenbasis enthält, die einen Treffer $Q \subseteq D_j$ nach Gleichung (2.1) darstellen.

**Konvention** Im folgenden wird statt $\mathcal{V}_\mathcal{D}$ nur noch $\mathcal{V}$ geschrieben, da klar ist, daß das Vokabular bzgl. einer Datenbasis $\mathcal{D}$ definiert ist. Analog werden invertierte Listen nur noch mit $L(w), w \in \mathcal{V}$, bezeichnet.

**Suche in Musikdatenbanken** Neben der Volltextsuche wurde in Abschnitt 2.1.1 auch auf eine Suche in Musikdatenbanken eingegangen. Diese dient als zweites Beispiel zur Motivation von invertierten Listen.

Dokumente, hier also Musikstücke, setzen sich in dieser Anwendung aus Noten $(p, t) \in \mathbb{N} \times \mathbb{N}$ mit Einsatzzeit $t$ und Tonhöhe $p$ zusammen. Treffer bzgl. einer Anfrage $Q$ sind nach Gleichung (2.2) diejenigen Dokumente, die $Q$ in einer zeitlich verschobenen Form, d.h. zu einem beliebigen Zeitpunkt, enthalten.

Für eine Menge $M \subset \mathbb{N} \times \mathbb{N}$ sei $\pi_1(M)$ als die elementweise Projektion auf den ersten Tupeleintrag definiert, d.h.

$$\pi_1(M) := \{p \mid (p, t) \in M\}.$$

Als Vokabular wird die Menge aller Tonhöhen der Noten aus den Musikstücken der Datenbank verwendet:

$$\mathcal{V} := \pi_1\Big(D_1 \cup \ldots \cup D_N\Big).$$

Bei der Volltextsuche besteht ein Eintrag in einer invertierten Liste aus einer Referenz auf ein Dokument. Bei der Suche in Musikstücken wird in einem Listeneintrag zusätzlich zur Dokumentreferenz noch eine Einsatzzeit $t$ gespeichert. Für eine Tonhöhe $p \in \mathcal{V}$ ergibt sich als invertierte Liste $L(p)$ :

$$L(p) := \{(j, t) \in [1:N] \times \mathbb{N} \mid (p, t) \in D_j\}.$$

Für eine invertierte Liste $L(p), p \in \mathcal{V}$, und $z \in \mathbb{Z}$ wird durch $L(p)-(0,z)$ eine zeitliche Verschiebung aller Einsatzzeiten in $L(p)$ um $z$ bezeichnet, d.h.

$$L(p) - (0,z) := \{(j, t-z) \mid (j,t) \in L(p)\}.$$

Diese zeitlichen Verschiebungen werden bei der Schnittmengenbildung der Trefferberechnung mitberücksichtigt. Für eine Anfrage $Q = \{(p_1,t_1),\ldots,(p_m,t_m)\}$ aus $m$ Noten mit $t_1 = 0 \leq t_2 \leq \ldots \leq t_m$ ergibt sich die Treffermenge $\mathcal{M}(Q)$ durch

$$\mathcal{M}(Q) = \cap_{k=1}^{m} \Big( L(p_k) - (0,t_k) \Big)$$

**Beispiel 3.2.1.** Sei $D_1 = \{(63,1),(64,6),(66,11),(65,17)\}$ das einzige Dokument einer Datenbasis $\mathcal{D}$. Als Vokabular $\mathcal{V}$ ergibt sich $\mathcal{V} = \{63,64,65,66\}$ und als invertierte Listen $L(63) = \{(1,1)\}$, $L(64) = \{(1,6)\}$, $L(65) = \{(1,17)\}$ und $L(66) = \{(1,11)\}$. Für die Anfrage $Q = \{(64,0),(66,5),(65,11)\}$ ergibt sich mit $t_1 = 0$ die Treffermenge $\mathcal{M}(Q) = \{(1,6)\}$ als Durchschnitt von $L(64) - (0,0) = \{(1,6)\}$, $L(66) - (0,5) = \{(1,11-5)\}$ und $L(65) - (0,11) = \{(1,17-11)\}$. Also liegt wegen $Q + (0,6) \subseteq D_1$ ein Treffer zum Zeitpunkt $t = 6$ vor. Man sieht, wie die einzelnen invertierten Listen in Abhängigkeit der Anfrage vor der Schnittbildung modifiziert werden müssen. ◦

**G-Treffer** Die Volltextsuche und die Suche in Musikstücken sind Sonderfälle der Suche nach $G$-Treffern in $G$-Mengen. In Abschnitt 2.1.2 wurde diese Mustersuche aus gruppentheoretischer Sicht beschrieben. Dort wurde auch dargestellt, wie sich $G$-invertierte Listen $G_{\mathcal{D}}(m)$ zusammensetzen. Als Vokabular $\mathcal{V}$ ergibt sich ein Repräsentantensystem $R$ der $G$-Bahnen der Objektmenge $\mathcal{O}$.

Gleichung (2.3) zeigt, wie sich die Treffermenge $\mathcal{M}(Q) = G_{\mathcal{D}}(Q)$ als Schnittmenge von $G$-invertierten Listen ergibt.

### 3.2.2 Fuzzy-Suche

Bei den oben besprochenen Anwendungen wurde immer "exakt" mittels invertierter Listen gesucht. In Abschnitt 2.1.3 wurde auf Varianten der unscharfen Suche eingegangen. Die erste dort vorgestellte Variante läßt sich mit invertierten Listen realisieren. Hierfür ist eine Anfrage $\hat{Q}$ eine Menge von nichtleeren Objektmengen $\hat{q}$.

Bisher hat jedes Objekt $q$ einer Anfrage eine invertierte Liste definiert, die Teil der Schnittmengenbildung zur Trefferberechnung war. Für $|Q| = m$ wurden $m$ invertierte Listen geschnitten.

Bei der unscharfen Suche ist dies anders. Hier ist ein "Objekt" der Anfrage eine Objektmenge $\hat{q}$, d.h. $\hat{Q} = \{\hat{q}_1,\ldots,\hat{q}_m\}$. Für jedes $\hat{q} \in \hat{Q}$, wird erst die Vereinigung aller invertierten Listen $L(q)$ mit $q \in \hat{q}$ berechnet, bevor diese geschnitten werden.

Die Suche in Musikstücken hat gezeigt, daß das Vokabular keine Teilmenge von $\mathcal{O}$ sein muß. Dort, sowie bei der Suche nach $G$-Treffern, wurde bei der Schnittmengenbildung die invertierten Listen modifiziert: $L(p) - (0, t_k - t_1)$ bzw. $G_{\mathcal{D}}(r_m)g_m^{-1}$.

Zur vereinfachten Darstellung sei $\mathcal{V} \subseteq \mathcal{O}$. Dann ergibt sich, ohne Berücksichtigung der Modifikation von invertierten Listen, für die Trefferberechnung bei einer unscharfen Suche:

$$\mathcal{M}(\hat{Q}) = \cap_{\hat{q}\in\hat{Q}} \big( \cup_{q\in\hat{q}} L(q) \big).$$

Beispielsweise ergibt sich dieser Bezug in der Musikanwendung durch

$$\mathcal{M}(\hat{Q}) = \cup_{(p,t)\in\hat{q}_1} L(p) \cap \bigcap_{k\in[2:m]} \cup_{(p',t')\in\hat{q}_k} \big( L(p') - (0, t'-t) \big).$$

Die zweite in Abschnitt 2.1.3 vorgestellte Variante der unscharfen Suche läßt sich i.d.R. nicht mittels invertierter Listen realisieren. Hier wird ein Distanzmaß auf Notenfolgen verwendet. Da aber die Noten einer Notenfolge in unterschiedliche invertierte Listen zerteilt werden, eignet sich das Konzept der invertierten Listen nicht, um Notenfolgen als Ganzes zu vergleichen.

### 3.2.3 Zusammenfassung und Fazit

Invertierte Listen (inverted file index) wurden ursprünglich zur Volltextsuche verwendet [50]. In den letzten Jahre wurden sie auch in anderen Anwendungsgebieten eingesetzt, z.B. für eine Suche in Musikstücken [4]. Das Konzept der invertierten Listen läßt sich auf $G$-Mengen verallgemeinern [5].

Die Objekte der Dokumente einer Datenbasis $\mathcal{D}$ definieren das Vokabular $\mathcal{V}$. Für jede Vokabel $v \in \mathcal{V}$ wird eine invertierte Liste $L(v)$ erzeugt, die hauptsächlich aus Referenzen auf diejenigen Dokumente bestehen, die $v$ enthalten. Für eine Anfrage $Q$ berechnet sich die Treffermenge durch Schnittmengenbildung von $|Q|$ invertierten Listen. Dies ist ein Grund, warum invertierte Listen bei der Volltextsuche in großen Datenbanken als effizient gelten.

Wie das Ausgangsszenario (Abschnitt 3.1.1) motiviert hat, werden große Datenbasen auf externen Medien gespeichert. Die Beantwortungszeit einer Anfrage hängt stark von dem Datentransfervolumen vom Sekundärmedium in den Hauptspeicher und von der dafür benötigten Anzahl von wahlfreien Zugriffen auf das Sekundärmedium ab.

Die Einträge einer invertierten Liste werden zusammenhängend auf dem Sekundärmedium gespeichert, d.h. für den Transfer einer invertierten Liste in den Hauptspeicher ist *ein* wahlfreier Zugriff nötig. Es ergeben sich für eine Anfrage $Q$ insgesamt $|Q|$ wahlfreie Zugriffe. Dies stellt bei kleinen Anfragen gegenüber großen Datenbasen einen erheblichen Vorteil dar. Weiterhin sind die Einträge jeder invertierten Listen sortiert nach der Dokumentreferenz abgespeichert. Dadurch läßt sich der Durchschnitt zweier Listen effizient berechnen.

Das Datentransfervolumen hängt von der Länge der invertierten Listen ab, die für eine Anfragebeantwortung benötigt werden. Somit sind kurze invertierte Listen wünschenswert. Bei der Volltextsuche ist dies i.d.R. der Fall. Sehr häufig vorkommende Wörter wie “und”, “oder” sowie Pronomen bilden eine Ausnahme und werden i.d.R. als Teil der Anfrage ignoriert.

Für beliebige Anwendungen lassen sich kurze invertierte Listen nicht immer realisieren. Für die Suche nach $G$-Treffern besteht ein Repräsentantensystem im Extremfall nur aus einem Element. Somit existiert nur eine invertierte Liste, in der alle Daten enthalten sind. In diesem schlechtesten Fall müssen für jedes Anfrageobjekt die kompletten Daten der Datenbasis abgearbeitet werden.

Auch kann die Beantwortungszeit unterschiedlicher Anfragen gleicher Länge verschieden sein, da die benötigten invertierten Listen i.d.R. unterschiedliche Längen besitzen.

Bei einer unscharfen Suche, wo eine Anfrage $\hat{Q}$ aus einer Folge von Objektmengen $\hat{q}$ besteht, werden pro $\hat{q}$ nun $|\hat{q}|$ invertierte Listen benötigt. Somit steigt die Anzahl der wahlfreien Zugriffe auf maximal $\sum_{\hat{q}\in\hat{Q}} |\hat{q}|$.

Eine Indexierung mittels invertierter Listen bietet sich nicht für jede Anwendung an. Hier muß überprüft werden, ob sich eine Aufteilung der Daten in “kurze” invertierte Listen ergibt, da diese das für eine Anfragebeantwortung benötigte Datentransfervolumen bestimmen. Für eine ausführliche Besprechung sei auf [50] verwiesen.

## 3.3 Trefferberechnung

### 3.3.1 Motivation

Bei den in Abschnitt 3.1.2 beschriebenen Verfahren wurde eine Datenbasis von Graphen sequentiell durchlaufen, um so jeden Graphen mit einer Anfrage auf (Sub-)Graphenisomorphie zu vergleichen. In den folgenden Abschnitten werden Verfahren vorgestellt, die alle Dokumente der Datenbank gleichzeitig betrachten.

Grundlage vieler Suchverfahren sind Invarianten bzgl. (Sub-)graphenisomorphie. Hiermit lassen sich Eigenschaften festlegen, die ein Treffer erfüllen muß. Von fundamentaler Bedeutung ist die Betrachtung des Grades $Grad(v)$ eines Knoten $v$. Hiermit ist für gerichtete Graphen die Kardinalität der Menge aller Nachbarknoten gemeint. Als direkte Folgerung aus Definition 2.4.4 ergibt sich für zwei isomorphe Digraphen $G, G'$ mit Bijektion $\phi : V_G \to V_{G'}$:

| Treffer-art | Knotengrad $\forall v \in Q$ | Kantenbeschriftung $\forall (v,w) \in E_Q$ | #Knoten |
|---|---|---|---|
| IC | $Grad(v) = Grad(\Psi(v))$ | $\xi_Q((v,w)) = \xi_D((\Psi(v), \Psi(w))$ | $\|Q\| = \|D\|$ |
| I | $Grad(v) \leq Grad(\Psi(v))$ | $\xi_Q((v,w)) = \xi_D((\Psi(v), \Psi(w))$ | $\|Q\| \leq \|D\|$ |
| SC | $Grad(v) \leq Grad(\Psi(v))$ | $\xi_Q((v,w)) \leq \xi_D((\Psi(v), \Psi(w))$ | $\|Q\| = \|D\|$ |
| S | $Grad(v) \leq Grad(\Psi(v))$ | $\xi_Q((v,w)) \leq \xi_D((\Psi(v), \Psi(w))$ | $\|Q\| \leq \|D\|$ |

Tabelle 3.1: Beziehungen zwischen Graphen $G_Q$ (Anfrage) und $G_D$ (Dokument) in Abhängigkeit von der Trefferart bei zugrundegelegter Trefferabbildung $\Psi$.

- Beide Graphen besitzen die gleich Knotenanzahl und Kantenanzahl, d.h. $|V_G| = |V_{G'}|$ und $|E_G| = |E_{G'}|$.
- Die Knotengrade stimmen überein, d.h. $Grad(v) = Grad(\phi(v)),\ \forall v \in V_G$.
- Die Knoten- und Kantenbeschriftungen stimmen überein, d.h. $\zeta_G(v) = \zeta_{G'}(\phi(v)),\ \forall v \in V_G$ und $\xi_G((v,w)) = \xi_{G'}((\phi(v), \phi(w))),\ \forall (v,w) \in E_G$.

Für das allgemeine Subgraphenisomorphieproblem zwischen zwei Graphen $G, G'$ ergibt sich für eine injektive Abbildung $\phi : V_G \to V_{G'}$:

- $|V_G| \leq |V_{G'}|$ und $|E_G| \leq |E_{G'}|$,
- $Grad(v) \leq Grad(\phi(v)),\ \forall v \in V_G$,
- $\zeta_G(v) = \zeta_{G'}(\phi(v)),\ \forall v \in V_G$ und $\xi_G((v,w)) = \xi_{G'}((\phi(v), \phi(w))),\ \forall (v,w) \in E_G$.

Für die Trefferbegriffe aus Abschnitt 2.3 gelten ähnliche Beziehungen. Ausgehend von der Graphendarstellung für eine Anfrage $Q$ und ein Dokument $D$ ergeben sich für $G_Q$ und $G_D$ nach Satz 2.4.10 die in Tabelle 3.1 dargestellten Beziehungen. Die dort aufgeführten Beziehungen zwischen der Knotenanzahl (rechte Spalte) gelten, außer bei SC-Treffern, auch für die Kantenanzahl, d.h. $|E_Q| = |E_D|$ bzw. $|E_Q| \leq |E_D|$. Für SC-Treffer gilt $|E_Q| \leq |E_D|$, da hier keine Bijektion zwischen den Kanten der Anfrage und den Kanten des Dokuments gefordert wird.

Alle aufgeführten Beziehungen werden bei der Indexierung bzw. bei der Suche ausgenutzt.

### 3.3.2 Vorkommen

Als Motivation für den verwendeten Index bzw. die Suchverfahren wird in diesem Abschnitt auf einen zentralen Aspekt eingegangen. Wegen ihrer anschaulichen Darstellungsform wird im folgenden die graphentheoretische Sicht bevorzugt, d.h. Anfragen $Q$ und Dokumente $D$ werden als beschriftete Graphen $G_Q$ bzw. $G_D$ angesehen. Der erste Teil dieses Abschnittes bezieht sich auf alle vier Trefferbegriffe aus Abschnitt 2.3. Später wird auf Besonderheiten der einzelnen Trefferbegriffe im Zusammenhang mit den Ergebnissen des letzten Abschnitts eingegangen.

Ein zentraler Schritt der Suchverfahren ist für ein Objekt $q$ einer Anfrage $Q$ und einem Dokument $D$ die Beantwortung der Frage: Welche Objekte $d \in D$ kommen als Bild $\Psi(q)$ in Frage, falls Dokument $D$ einen Treffer mit Trefferabbildung $\Psi$ darstellt?

Um festzustellen, ob $D$ ein Treffer bzgl. $Q$ ist, werden Trefferabbildungen $\Psi : Q \to D$ sukzessive konstruiert. Hierfür werden nacheinander alle Anfrageobjekte betrachtet. Später soll die Suche nicht nur auf ein Dokument beschränkt werden, sondern gleichzeitig auf alle Dokumente einer Datenbasis. Doch starten wir die Konstruktion erstmal nur bzgl. eines Anfrageobjektes $q \in Q$ und einem Dokument $D$.

Im folgenden wird $q$ und dessen direkte Nachbarn im Vergleich zu $\Psi(q)$ und dessen direkte Nachbarn untersucht. Für diese lokale Sichtweise werden folgende Definitionen verwendet.

**Definition 3.3.1.** Sei $G = (V, \zeta, E, \xi)$ ein gerichteter beschrifteter Graph. Für einen Knoten $v \in V$ ist ein **Vorgänger** von $v$ ein Knoten $u$ mit einer Kante von $u$ nach $v$, d.h. $(u,v) \in E$. Mit

$\mathtt{V}(v, G)$ wird die Menge aller Vorgänger von $v$ in $G$ bezeichnet: $\mathtt{V}(v, G) := \{u \in V | (u, v) \in E\}$. Entsprechend ist ein **Nachfolger** von $v$ ein Knoten $w \in V$ mit einer Kante $(v, w) \in E$ und $\mathtt{N}(v, G) := \{w \in V | (v, w) \in E\}$. Mit der Nachbarschaft oder **Umgebung** $\mathtt{U}(v, G)$ von $v$ sind alle Vorgänger und Nachfolger von $v$ gemeint, d.h. $\mathtt{U}(v, G) := \mathtt{V}(v, G) \cup \mathtt{N}(v, G)$.

Analog lassen sich die Begriffe für Dokumente $D \subset \mathcal{O}$ und eine Menge $\mathcal{R}$ von Relationen definieren. Für $o \in D$ ist $\mathtt{V}(o, D, \mathcal{R}) := \{p \in D | \exists R \in \mathcal{R} : (p, o) \in R\}$, $\mathtt{N}(o, D, \mathcal{R}) := \{p \in D | \exists R \in \mathcal{R} : (o, p) \in R\}$ und $\mathtt{U}(o, D, \mathcal{R}) := \mathtt{N}(o, D, \mathcal{R}) \cup \mathtt{V}(o, D, \mathcal{R})$. Da $\mathcal{R}$ für eine Anwendung fest gewählt ist, wird oft nur kurz $\mathtt{V}(o, D)$ geschrieben.

Im letzten Abschnitt wurden einige Anforderungen gestellt, die zwischen $q$ und $\Psi(q)$ gelten müssen (siehe Tabelle 3.1). Dies betrifft den Knotengrad von $q$ und $\Psi(q)$ sowie die Beschriftung der eingehenden und ausgehenden Kanten von $q$ von $\Psi(q)$.

Betrachtet man $q$ und seine Umgebung in $G_Q$, so läßt sich diese lokale Sichtweise durch einen Subgraphen $G_Q^q$ darstellen, der nur aus mit $q$ verbundenen Kanten besteht und nur Knoten aus der Umgebung von $q$ enthält.

**Definition 3.3.2.** Sei $G_Q = (Q, \zeta_Q, E_Q, \xi_Q)$ ein beschrifteter Graph und $q \in Q$. Dann ist der Graph $G_Q^q = (Q^q, \zeta_Q^q, E_Q^q, \xi_Q^q)$ definiert durch $Q^q := \mathtt{U}(q, G_Q) \cup \{q\}$, $\zeta_Q^q = \zeta_Q \downarrow Q^q$, $E_Q^q := \big(E_Q \cap (\{q\} \times Q^q)\big) \cup \big(E_Q \cap (Q^q \times \{q\})\big)$, $\xi_Q^q := \xi_Q \downarrow E_Q^q$.

In Abbildung 3.3 ist ein Beispiel dargestellt. Im folgenden wird immer davon ausgegangen, daß mit $G_Q$ ein beschrifteter Graph $G_Q = (Q, \zeta_Q, E_Q, \xi_Q)$, mit $G_D$ ein beschrifteter Graph $G_D = (D, \zeta_D, E_Q, \xi_D)$, mit $G_Q^q$ der Graph $G_Q^q = (Q^q, \zeta_Q^q, E_Q^q, \xi_Q^q)$ nach Definition 3.3.2 für ein $q \in Q$ und mit $G_D^d$ der Graph $G_D^d = (D^d, \zeta_D^d, E_D^d, \xi_D^d)$, ebenfalls nach Definition 3.3.2 für ein $d \in D$ gemeint ist, ohne jeweils immer das entsprechende Viertupel mit anzugeben.

**Bemerkung 3.3.3.** Der Graph $G_Q^q$ ist bipartit mit der Bipartition $\{q\}$ und $Q^q \setminus \{q\}$, da nach Konstruktion nur Kanten von oder nach $q$ in $E_Q^q$ enthalten sind.

Analog zur Gegenüberstellung von Dokumenten und Graphen aus Abschnitt 2.4.2 ergibt sich das $G_Q^q$ entsprechende Dokument $Q^q$ bzgl. $\mathcal{R}$ als $Q^q := \{q\} \cup \mathtt{U}(q, G_Q)$. Weiterhin wird $\mathcal{R}$ eingeschränkt auf $\mathcal{R}^q := \{\big(R \cap (\{q\} \times \mathtt{U}(q))\big) \cup \big(R \cap (\mathtt{U}(q) \times \{q\})\big) | R \in \mathcal{R}\}$. Pro Relation $R$ in $\mathcal{R}$ werden hier nur die Tupel betrachtet, bei denen ein Eintrag gleich $q$ ist.

Mit Hilfe dieser lokalen Sicht auf ein Anfrageobjekt $q$ und dessen Umgebung wird nun in Dokumenten $D$ nach "passenden" Objekten $d \in D$ bzgl. einer Trefferabbildung $\Psi : Q^q \to D^d$ gesucht. Diese hängen vom jeweiligen Trefferbegriff ab.

Um festzustellen, ob Dokument $D$ einen Treffer bzgl. $Q^q$ und $\mathcal{R}^q$ darstellt, läßt sich Satz 2.4.10 anwenden. Im folgenden wird auf die Berechnung der Subgraphen von $G_D$ aus Satz 2.4.10

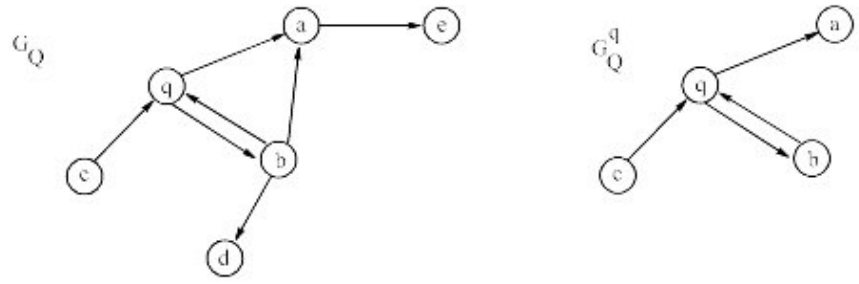



Abbildung 3.3: Ein Graph $G_Q$ und bzgl. $q$ der Graph $G_Q^q$. In letzterem sind nur $q$ und dessen Nachbarknoten sowie Kanten von und nach $q$ enthalten. Z.B. ist die Kante $(b, a)$ nicht in $E_Q^q$. Kantenbeschriftungen wurden bei der Darstellung vernachlässigt.

eingegangen, um somit alle Treffer zu erhalten. Ein Subgraph wird durch eine Abbildung $\Psi : Q^q \to D$ definiert, die je nach Trefferart injektiv oder bijektiv ist. Für S- und SC-Treffer wird zusätzlich eine injektive Abbildung $\psi$ auf den Kantenmengen definiert, d.h. $\psi : E_Q^q \to E_D$ (siehe Satz 2.4.10).

Für die lokale Sicht auf $G_Q^q$ werden die Objekte $d$ eines Dokuments $D$ als Vorkommen von $q$ bezeichnet, für die eine Abbildung $\Psi$ bzgl. Satz 2.4.10 existiert mit $\Psi(q) = d$. Pro Trefferart wird eine "Vorkommenart" definiert, um so den unterschiedlichen Eigenschaften der Trefferarten gerecht zu werden.

Im folgenden wird für einen Treffer $G_D$ einer Anfrage $G_Q$ unterschieden zwischen einer globalen Sichtweise bzgl. einer globalen Trefferabbildung $\Psi : Q \to D$, die sich auf eine komplette Anfrage $Q$ bezieht, und einer lokalen Sichtweise, die sich auf ein Anfrageobjekt $q \in Q$ mit einer lokalen Trefferabbildung $\psi^{q,d} : Q^q \to D^d$ bezieht.

**Definition 3.3.4.** Seien $G_Q$ und $G_D$ zwei gerichtete beschriftete Graphen und $q \in Q$. Ein Knoten $d \in D$ heißt **S-Vorkommen** von $q$ genau dann, wenn eine injektive Abbildung $\psi^{q,d} : \mathtt{U}(q, G_Q) \to \mathtt{U}(d, G_D)$ existiert mit

1. $\psi^{q,d}(\mathtt{N}(q, G_Q)) \subseteq \mathtt{N}(d, G_D)$,
2. $\psi^{q,d}(\mathtt{V}(q, G_Q)) \subseteq \mathtt{V}(d, G_D)$,
3. $\psi^{q,d}(\mathtt{N}(q, G_Q)) \cap \mathtt{V}(q, G_Q)) \subseteq \mathtt{N}(d, G_D) \cap \mathtt{V}(d, G_D)$,
4. $\forall v \in \mathtt{V}(q, G_Q) : \xi_Q((v, q)) \leq \xi_D((\psi^{q,d}(v), d))$ und
5. $\forall w \in \mathtt{N}(q, G_Q) : \xi_Q((q, w)) \leq \xi_D(d, \psi^{q,d}(w))$.

Ein Knoten $d \in D$ heißt **I-Vorkommen** von $q$ genau dann, wenn $d$ ein S-Vorkommen von $q$ ist und zusätzlich gilt:

- $\psi^{q,d}(\mathtt{N}(q, G_Q) \setminus \mathtt{V}(q, G_Q)) \subseteq \mathtt{N}(d, G_D) \setminus \mathtt{V}(d, G_D)$,
- $\psi^{q,d}(\mathtt{V}(q, G_Q) \setminus \mathtt{N}(q, G_Q)) \subseteq \mathtt{V}(d, G_D) \setminus \mathtt{N}(d, G_D)$,
- $\forall v \in \mathtt{V}(q, G_Q) : \xi_Q((v, q)) = \xi_D((\psi^{q,d}(v), d))$ und
- $\forall w \in \mathtt{N}(q, G_Q) : \xi_Q((q, w)) = \xi_D(d, \psi^{q,d}(w))$.

Ein Knoten $d \in D$ heißt **IC-Vorkommen** von $q$ genau dann, wenn $d$ ein I-Vorkommen von $q$ ist und zusätzlich $\psi^{q,d}$ eine Bijektion ist. Es gilt also:

- $\psi^{q,d}(\mathtt{N}(q, G_Q) \setminus \mathtt{V}(q, G_Q)) = \mathtt{N}(d, G_D) \setminus \mathtt{V}(d, G_D)$,
- $\psi^{q,d}(\mathtt{V}(q, G_Q) \setminus \mathtt{N}(q, G_Q)) = \mathtt{V}(d, G_D) \setminus \mathtt{N}(d, G_D)$ und
- $\psi^{q,d}(\mathtt{N} \cap \mathtt{V}(q, G_Q)) = \mathtt{N}(d, G_D) \cap \mathtt{V}(d, G_D)$.

Wir sagen auch genauer für eine Vorkommenart $T \in \{S, I, IC\}$: $d$ ist $T$-Vorkommen von $q$ bzgl. $\psi^{q,d}$.

S-Vorkommen sind die allgemeinste Art von Vorkommen. Die erste Bedingung garantiert für die injektive Abbildung $\psi^{q,d}$, daß Nachfolger von $q$ auf Nachfolger von $d$ abgebildet werden. Analoges gilt für die nächsten beiden Bedingungen bzgl. Vorgänger und Nachfolger, die auch Vorgänger sind. Durch die letzten beiden Bedingungen wird sichergestellt, daß die Beschriftungen der Kanten von bzw. nach $q$ und den mittels $\psi^{q,d}$ entsprechenden Kanten von bzw. nach $d$ "passend" bzgl. der Ordnung "$\leq$" sind.

Diese Bedingungen werden für I-Vorkommen weiter eingeschränkt. "Nur" Nachfolger von $q$ werden auf "nur" Nachfolger von $d$ abgebildet. Entsprechendes gilt für "nur" Vorgänger. Dadurch wird sichergestellt, daß zwei Knoten genau dann in der Anfrage mit einer Kante verbunden sind, wenn dies auch im Dokument gilt. Dies wird in Abbildung 3.4 exemplarisch veranschaulicht. Weiterhin wird eine Gleichheit der Kantenbeschriftungen gefordert.

Während bei I-Vorkommen von einer injektiven Abbildung $\psi^{q,d}$ ausgegangen wird, wird bei IC-Vorkommen eine Bijektion gefordert.

Eine Unterscheidung in SC-Vorkommen und S-Vorkommen entfällt, da bei den entsprechenden Trefferbegriffen nur der globale Unterschied $|Q| = |D|$ bzw. $|Q| \leq |D|$ gilt. Für die lokale Sichtweise auf die Umgebung von $q$ hat dies keine Auswirkung.

Analog zur Hierarchie auf den vier Trefferarten aus Abschnitt 2.3 existiert offensichtlich eine Hierarchie auf den drei Arten von Vorkommen:

$$\text{IC-Vorkommen} \Rightarrow \text{I-Vorkommen} \Rightarrow \text{S-Vorkommen}.$$

In Abbildung 3.5 sind Beispiele von Vorkommen dargestellt.

Einen Zusammenhang zwischen der Trefferberechnung bzgl. $Q^q$ bzw. $G_Q^q$ und Vorkommen von $q$ ergibt sich durch folgenden Satz, wobei hier eine zu $q$ analoge lokale Sichtweise auf Dokumentebene angewendet wird.

**Satz 3.3.5.** *Seien $G_Q$ und $G_D$ zwei beschriftete Graphen, $q$ ein Knoten von $G_Q$ und $d$ ein Knoten aus $G_D$.*

1. *$G_D^d$ ist IC-Treffer bzgl. $G_Q^q \Leftrightarrow$ $d$ ist IC-Vorkommen von $q$.*
2. *$G_D^d$ ist I-Treffer bzgl. $G_Q^q \Leftrightarrow$ $d$ ist I-Vorkommen von $q$.*
3. *$G_D^d$ ist SC-Treffer bzgl. $G_Q^q \Leftrightarrow$ $d$ ist S-Vorkommen von $q$ mit $|\mathtt{U}(q, G_Q)| = |\mathtt{U}(d, G_D)|$.*
4. *$G_D^d$ ist S-Treffer bzgl. $G_Q^q \Leftrightarrow$ $d$ ist S-Vorkommen von $q$.*

**Beweis** Die Aussagen folgen sofort aus Definition 3.3.4 und Satz 2.4.10, wenn man $\psi^{q,d} : \mathtt{U}(q, G_Q) \to \mathtt{U}(d, G_D)$ als Einschränkung von $\Psi : Q^q \to D^d$ ansieht, da $Q^q = \{q\} \cup \mathtt{U}(q, G_Q)$. □

Im folgenden wird $\psi^{q,d}$ aus Definition 3.3.4 auch als Abbildung $\psi^{q,d} : \{q\} \cup \mathtt{U}(q, G_Q) \to \{d\} \cup \mathtt{U}(d, G_D)$ mit $\psi^{q,d}(q) = d$ angesehen.

Der letzte Satz spiegelt eine lokale Treffersicht innerhalb $G_Q$ auf die Umgebung von $q$, den Subgraphen $G_Q^q$ von $G_Q$, wider. Für Treffer bzgl. des kompletten Anfragegraphen $G_Q$ ergibt sich aus der lokalen Sicht:

**Satz 3.3.6.** *Seien $G_Q$ und $G_D$ zwei beschriftete Graphen, $q$ ein Knoten von $G_Q$ und $d$ ein Knoten aus $G_D$.*

1. *Ist $G_D$ ein IC-Treffer bzgl. $G_Q$ mit Trefferabbildung $\Psi$, dann ist $\Psi(q)$ IC-Vorkommen von $q$ für alle $q \in Q$.*
2. *Ist $G_D$ ein I-Treffer bzgl. $G_Q$ mit Trefferabbildung $\Psi$, dann ist $\Psi(q)$ I-Vorkommen von $q$ für alle $q \in Q$.*
3. *Ist $G_D$ ein SC-Treffer oder ein S-Treffer bzgl. $G_Q$ mit Trefferabbildung $\Psi$, dann ist $\Psi(q)$ S-Vorkommen von $q$ für alle $q \in Q$.*

**Beweis** Der Satz folgt direkt aus Satz 3.3.5. □

Beispiele für beide Sätze befinden sich in Abbildung 3.5. Der dort dargestellte Graph $G_4$ zeigt, daß die Umkehrung von Satz 3.3.6 nicht gilt: Obwohl für jedes $q$ der Anfrage ein Vorkommen in $G_4$ enthalten ist, bildet $G_4$ keinen Treffer bzgl. $G_Q$.

Bisher wurde nicht auf eine effiziente Berechnung von Vorkommen bei gegebenem $q \in Q$ und $D$ eingegangen. Im Abschnitt 3.3.5 wird eine Indexstruktur definiert, mit der sich alle Vorkommen von $q \in Q$ nicht nur in einem einzelnen Dokument $D$, sondern in allen Dokumenten einer Datenbasis berechnen lassen. Im darauffolgenden Abschnitt wird die Berechnung der Vorkommen unter Verwendung der Indexstruktur beschrieben. Im nächsten Abschnitt wird gezeigt, wie sich Vorkommen zu Treffern kombinieren lassen.

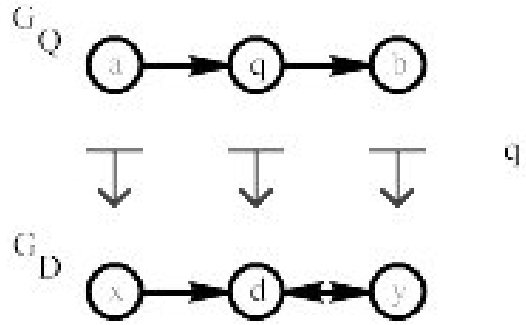



Abbildung 3.4: $d$ ist S-Vorkommen mit Trefferabbildung $\psi^{q,d} : Q \to D$, $a \mapsto x, q \mapsto d$ und $b \mapsto y$. Allerdings stellt $d$ kein I-Vorkommen von $q$ dar, da $\psi^{q,d}(\mathtt{N}(q, G_Q) \setminus \mathtt{V}(q, G_Q)) \subseteq \mathtt{N}(d, G_D) \setminus \mathtt{V}(d, G_D)$ wegen der Kante $(y, d) \in E_D$ nicht gilt.

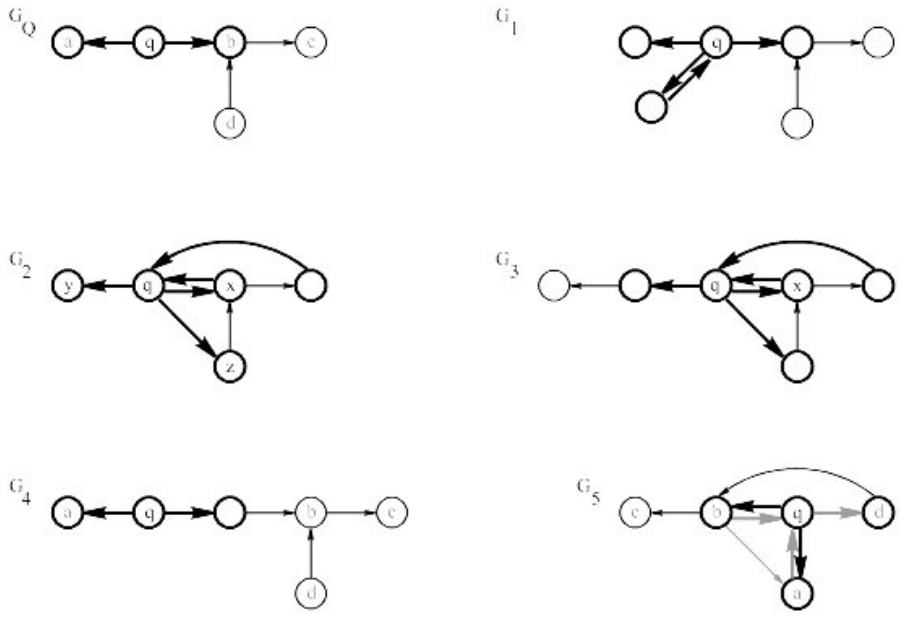



Abbildung 3.5: Sechs beschriftete Graphen, in denen die Umgebung von $q$ fett markiert ist. $G_Q$ dient als Anfrage. Der Übersicht halber wurden keine Kantenbeschriftungen dargestellt. Es wird davon ausgegangen, daß diese für alle Kanten gleich sind. Alle Graphen enthalten ein I-Vorkommen von $q$ aus $G_Q$, ebenfalls mit $q$ in den Graphen markiert, z.B. $y \leftarrow q \rightarrow z$ in $G_2$. In $G_4$ stellt das Vorkommen sogar ein IC-Vorkommen dar. $G_2, G_3$ und $G_5$ besitzen zusätzlich ein S-Vorkommen von $q$, in $G_2$ und $G_3$ mit $x$ markiert. Ein Vorkommen garantiert noch keinen Treffer von $G_Q$, was $G_4$ demonstriert. $G_1$ ist I-Treffer, $G_2$ SC-Treffer und $G_3$ S-Treffer von $G_Q$, jeweils mit genau einer Trefferabbildung. $G_5$, eine Erweiterung von $G_2$, ist SC-Treffer mit zwei Trefferabbildungen, eine analog zu $G_2$ und eine bzgl. der Markierungen von $G_Q$.

### 3.3.3 Treffer aus Vorkommen

Der vorherige Abschnitt hat gezeigt, daß sich Treffer aus Vorkommen zusammensetzen, d.h. für jedes Objekt $q$ einer Anfrage $Q$ gilt für eine Trefferabbildung $\Psi$ mit Treffer $D$, daß $\Psi(q)$ ein Vorkommen von $q$ darstellt (Satz 3.3.6). Allerdings bildet eine Dokument, das für jedes Anfrageobjekt ein Vorkommen enthält, nicht immer einen Treffer. Dies wurde in Abbildung 3.5 durch $G_4$ gezeigt.

Im folgenden wird untersucht, wann ein Dokument, das für jedes Anfrageobjekt ein Vorkommen enthält, einen Treffer darstellt. Dabei werden globale Trefferabbildungen aus den lokalen Informationen über Vorkommen konstruiert.

Wegen der Injektion bzw. Bijektion zwischen Anfrage- und Dokumentobjekten dürfen keine zwei Anfrageobjekte auf dasselbe Objekt im Dokument abgebildet werden. Bzgl. einer Anfrage und eines Treffers müssen also die Vorkommen der Anfrageobjekte alle paarweise verschieden sein.

Weiterhin müssen die Nachbarschaften erhalten bleiben. Falls $q$ und $q'$ in der Anfrage benachbart sind, so muß dies auch für das Vorkommen von $q$ und das Vorkommen von $q'$ gelten. In Abbildung 3.5 sind in der Anfrage $G_Q$ die Knoten $b$ und $q$ benachbart, in $G_4$ gilt dies für die Vorkommen von $b$ und $q$ nicht.

Um die lokalen Informationen der Vorkommen beschreiben zu können, benötigen wir Abbildungen, die von Vorkommen eines Anfrageobjektes induziert werden.

**Definition 3.3.7.** Seien $G_Q$ und $G_D$ zwei beschriftete Graphen, $q$ ein Knoten von $G_Q$ und $d$ ein Knoten aus $G_D$. Weiterhin sei $G_D^d$ ein Treffer von $G_Q^q$. Dann bezeichnet $\psi^{q,d}$ eine **lokale** Trefferabbildung $\psi^{q,d} : Q^q \to D^d$ mit $\psi^{q,d}(q) = d$. Weiterhin wird mit $\Psi^{q,d} := \{\psi^{q,d} | \psi^{q,d}$ ist lokale Trefferabbildung $Q^q \to D^d\}$ die Menge aller lokalen Trefferabbildungen bzgl. $q$ und $d$ bezeichnet.

Nach Satz 3.3.5 ist $\Psi^{q,d} \neq \emptyset$ für jedes Vorkommen $d$ von $q$. Dies folgt aus dem Beweis von Satz 3.3.5 ("$\Leftarrow$"). Für IC-Treffer und SC-Treffer sind alle Elemente aus $\Psi^{q,d}$ bijektiv, für I-Treffer und S-Treffer injektiv.

Bzgl. $G_Q, G_Q^q, G_D$ und $G_D^d$ können mehrere Abbildungen $\psi^{q,d}$ existieren. Beispielsweise besitzt das mit $q$ markierte S-Vorkommen in $G_2$ aus Abbildung 3.5 insgesamt sechs verschiedene lokale Trefferabbildungen.

Bei gegebenem Vorkommen $d$ von $q$ mit Trefferabbildung $\psi^{q,d}$ ergibt sich ein Bezug zu Treffern durch folgenden Satz.

**Satz 3.3.8.** *Seien $G_Q$ und $G_D$ beschriftete Graphen und für jedes $q \in Q$ sei $d_q \in D$ ein Vorkommen von $q$ zusammen mit einer lokalen Trefferabbildung $\psi^{q,d_q}$ mit $\psi^{q,d_q}(q) = d_q$. Weiterhin gelte*

$$|Q| = \left|\{\psi^{q,d_q}(q) | q \in Q\}\right| \tag{3.2}$$

*und*

$$\forall a, b \in Q \; \forall p \in Q^a \cap Q^b : \psi^{a,d_a}(p) = \psi^{b,d_b}(p) \tag{3.3}$$

*Dann gilt:*

1. *$G_D$ ist IC-Treffer von $G_Q \Leftrightarrow$*
   *$|Q| = |D|$ und $\forall q \in Q : \psi^{q,d_q}(q)$ ist IC-Vorkommen von $q$.*

2. *$G_D$ ist I-Treffer von $G_Q \Leftrightarrow$*
   *$\forall q \in Q : \psi^{q,d_q}(q)$ ist I-Vorkommen von $q$.*

3. *$G_D$ ist SC-Treffer von $G_Q \Leftrightarrow$*
   *$|Q| = |D|$ und $\forall q \in Q : \psi^{q,d_q}(q)$ ist S-Vorkommen von $q$.*

4. *$G_D$ ist S-Treffer von $G_Q \Leftrightarrow$*
   *$\forall q \in Q : \psi^{q,d_q}(q)$ ist S-Vorkommen von $q$.*

**Beweis** "⇒": Die Implikation ergibt sich aus Satz 3.3.6.
"⇐": Für die Rückrichtung wird gezeigt, wie sich aus den einzelnen Abbildungen $\psi^{q,d_q}$, die jeweils für die Umgebung von $q$ definiert sind, eine globale Trefferabbildung $\Psi$ konstruieren läßt. Die Trefferabbildung $\Psi : Q \to D$ ergibt sich für alle Trefferarten durch:

$$\Psi(q) := \psi^{q,d_q}(q) \tag{3.4}$$

für alle $q \in Q$. Wegen $|Q| = \left|\{\psi^{q,d_q}(q) | q \in Q\}\right|$ ist $\Psi$ injektiv.

Pro Trefferart werden nun die an einen Treffer gestellten Bedingungen nachgeprüft.
(1) IC-Treffer: Wegen $|Q| = |D|$ ist $\Psi$ eine Bijektion. Die Bijektion zwischen den Kanten $E_Q$ und $E_D$ ergibt sich durch:

$$\begin{aligned}
(a,b) \in E_Q &\Leftrightarrow (a,b) \in E_Q^a \wedge (a,b) \in E_Q^b && (3.5)\\
&\Leftrightarrow (\psi^{a,d_a}(a), \psi^{a,d_a}(b)) \in E_D^{\psi^{a,d_a}(a)} \wedge (\psi^{b,d_b}(a), \psi^{b,d_b}(b)) \in E_D^{\psi^{b,d_b}(b)} && (3.6)\\
&\Leftrightarrow (\Psi(a), \Psi(b)) \in E_D^{\psi^{a,d_a}(a)} \wedge (\Psi(a), \Psi(b)) \in E_D^{\psi^{b,d_b}(b)} && (3.7)\\
&\Leftrightarrow (\Psi(a), \Psi(b)) \in E_D. && (3.8)
\end{aligned}$$

Die Äquivalenz in (3.5) sowie zwischen (3.7) und (3.8) ergibt sich aus der Konstruktion von $G_Q^a$, der einen Teilgraphen von $G_Q$ darstellt (für $G_D$ analog). Aus den IC-Vorkommen folgt wegen $G_Q^a \simeq G_D^{\psi^{a,d_a}(a)}$ aus Satz 3.3.5 die Äquivalenz zwischen (3.5) und (3.6). Die nachfolgende Äquivalenz ergibt sich aus der Voraussetzung $\psi^{a,d_a}(a) = \psi^{b,d_b}(a)$ sowie $\psi^{a,d_a}(b) = \psi^{b,d_b}(b)$ und der Konstruktion von $\Psi$ nach (3.4). Zusätzlich zur Bijektion zwischen den Kanten in $G_Q$ und den Kanten in $G_D$ ergibt sich aus den einzelnen IC-Vorkommen auch die Gleichheit der Kantenbeschriftungen.
(2) I-Treffer: Der Beweis erfolgt im wesentlichen analog zum Beweis bzgl. IC-Treffer. $\Psi$ ist allerdings nur eine injektive Abbildung. Trotzdem gelten alle oben aufgeführten Äquivalenzen, da $G_Q^a \simeq G_D^{\psi^{a,d_a}(a)} \downarrow \psi^{a,d_a}(Q)$ gilt. Es können zwar weitere Kanten $(\psi^{a,d_a}(a), c) \in E_D$ existieren. Für den Fall, daß ein $q' \in Q$ mit $c \in \psi^{q',d_{q'}}(Q^{q'})$ existiert, gilt (3.3). Ansonsten ist $c \notin \Psi(Q)$ und somit kein Teil des Treffers.
(3), (4) SC-Treffer und S-Treffer: Hier muß nur $(a,b) \in E_Q \Rightarrow (\Psi(a), \Psi(b)) \in E_D$ bewiesen werden. Diese Implikation ergibt sich aus der Konstruktion der S-Vorkommen, die auch die Beziehung zwischen den Kantenbeschriftungen sicherstellen. Im Gegensatz zu den beiden vorherigen Treffern besteht hier keine Bijektion zwischen Kanten in $E_Q$ und Kanten in $E_D \cap (\Psi(Q) \times \Psi(Q))$. Eigenschaft (3.3) garantiert allerdings die erforderliche injektive Abbildung zwischen $E_Q$ und $E_D$. □

Der Beweis zeigt, daß (3.3) eine wichtige Eigenschaft ist, die für alle Trefferarten gilt. Sie bezieht sich nicht lokal auf *ein* Vorkommen, sondern stellt den Zusammenhang zwischen *allen* Vorkommen sicher. Abbildung 3.6 demonstriert dies.

Für die Konstruktion von globalen Trefferabbildungen nach Satz 3.3.8 Formel (3.4) wurde für jedes Objekt $q$ einer Anfrage $Q$ ein Vorkommen benötigt. Diese Anzahl läßt sich teilweise einschränken, so daß nur für eine Teilmenge aller Anfrageobjekte Vorkommen berechnet werden müssen.

Sei $G = (V, E)$ ein zusammenhängender gerichteter Graph und $V'$ eine nichtleere Teilmenge von $V$. Im folgenden wird an $V'$ die Bedingung

$$\forall (a,b) \in E : \; a \in V' \vee b \in V' \tag{3.9}$$

gestellt. Hierdurch wird für einen zusammenhängenden Graphen gewährleistet, daß jeder Knoten $v \in V \setminus V'$ einen Nachbarknoten in $V'$ besitzt. Zusätzlich findet eine Art "Kantenüberdeckung" statt: Jede Kante ist mit mindestens einem Knoten aus $V'$ verbunden. Bipartite Graphen bzgl. der Unterteilung $V = V_1 \sqcup V_2$ sind hierfür ein Sonderfall.

In der Literatur werden Knotenmengen, die Gleichung (3.9) erfüllen, als trennende Knotenmengen bezeichnet [48]. Da zwei Verfahren zur Trefferberechnung auf solchen $V'$ basieren, wird

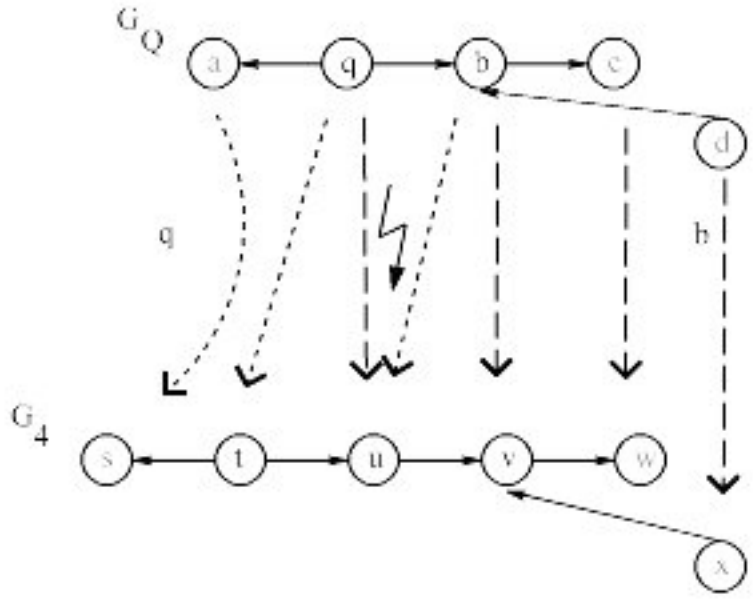



Abbildung 3.6: Oben die Anfrage $G_Q$, unten Dokument $G_4$ aus Abbildung 3.5. Seien $\psi^{q,t}$ : $\{a, q, b\} \rightarrow \{s, t, u\}$ durch $a \mapsto s$, $q \mapsto t$ sowie $b \mapsto u$ und $\psi^{b,v} : \{q, b, c, d\} \rightarrow \{u, v, w, x\}$ durch $q \mapsto u$, $b \mapsto v$, $c \mapsto w$ und $d \mapsto x$ definiert. Dann sind $\psi^{q,t}$ und $\psi^{b,v}$ lokale Trefferabbildungen aus $G_Q$ nach $G_D$. In obiger Abbildung wird $\psi^{q,t}$ durch die drei linken gepunkteten Pfeile und $\psi^{b,v}$ durch die vier rechten gestrichelten Pfeile veranschaulicht. Hier wird deutlich, daß $G_4$ kein Treffer ist, obwohl für jedes Anfrageobjekt ein Vorkommen in $G_4$ enthalten ist, da $\psi^{q,t}(q) \neq \psi^{b,v}(q)$ und $\psi^{q,t}(b) \neq \psi^{b,v}(b)$.

eine Teilmenge $V'$ einer Knotenmege $V$ eines Graphen $G$, die Gleichung 3.9 erfüllt, als **Basis** von $V$ (und $G$) bezeichnet.

Auf eine Anfrage $G_Q$ bezogen existiert für jede Basis $Q'$ von $G_Q$, die also bzgl. $G_Q$ Bedingung (3.9) erfüllt, mindestens eine Abbildung $\alpha : V \to V'$ mit $\alpha(q) \in \mathtt{U}(q, G) \cup \{q\}$. Hierdurch wird die oben beschriebene Eigenschaft festgehalten, daß jeder Knoten in $Q \setminus Q'$ mindestens einen Nachbarknoten in $Q'$ besitzt. Zusätzlich wird davon ausgegangen, daß $\alpha \downarrow Q' = id_{Q'}$ gilt.

Analog zu Satz 3.3.8 ergibt sich der folgende Satz, der anschließend durch ein Beispiel veranschaulicht wird.

**Satz 3.3.9.** *Seien $G_Q$ und $G_D$ beschriftete Graphen und $Q' \subseteq Q$ eine Basis von $G_Q$ Für jedes $q \in Q'$ sei $d_q \in D$ ein Vorkommen von $q$ zusammen mit einer lokalen Trefferabbildung $\psi^{q,d_q}$ mit $\psi^{q,d_q}(q) = d_q$. Weiterhin sei $\alpha : Q \to Q'$ eine totale Abbildung mit $\forall q \in Q : \alpha(q) \in \mathtt{U}(q) \cup \{q\}$ und es gelte*

$$|Q| = \left|\cup_{q \in Q'} \psi^{q,d_q}(Q^q)\right| \tag{3.10}$$

*sowie*

$$\forall a, b \in Q' \; \forall p \in Q^a \cap Q^b : \psi^{a,d_a}(p) = \psi^{b,d_b}(p). \tag{3.11}$$

*Dann gilt für Vorkommen bzgl. $Q'$ (statt bzgl. $Q$ wie in Satz 3.3.8):*

1. *$G_D$ ist IC-Treffer von $G_Q \Leftrightarrow$*
   *$|Q| = |D|$, $\forall q \in Q' : \psi^{q,d_q}(q)$ ist IC-Vorkommen von $q$ und*
   *$\forall q \in Q \setminus Q' : |\mathtt{U}(q, G_Q)| = \left|\mathtt{U}(\psi^{\alpha(q),d_q}(q), G_D)\right|$.*

2. *$G_D$ ist I-Treffer von $G_Q \Leftrightarrow$*
   *$\forall q \in Q' : \psi^{q,d_q}(q)$ ist I-Vorkommen von $q$ und*
   *$\forall q \in Q \setminus Q' : |\mathtt{U}(q, G_Q)| = \left|\mathtt{U}(\psi^{\alpha(q),d_q}(q), G_D) \cap \cup_{q' \in Q'} \psi^{q',d_{q'}}(Q^{q'})\right|$.*

3. *$G_D$ ist SC-Treffer von $G_Q \Leftrightarrow$*
   *$|Q| = |D|$ und $\forall q \in Q' : \psi^{q,d_q}(q)$ ist S-Vorkommen von $q$.*

4. *$G_D$ ist S-Treffer von $G_Q \Leftrightarrow$*
   *$\forall q \in Q' : \psi^{q,d_q}(q)$ ist S-Vorkommen von $q$.*

**Beweis** Der Beweis erfolgt analog zum Beweis von Satz 3.3.8, allerdings unter Berücksichtigung der Einschränkung von $Q$ nach $Q'$. Hierdurch verringert sich die Anzahl der gegebenen Vorkommen und Abbildungen $\psi^{q,d_q}$.
"⇒": Sei $\Psi$ eine Trefferabbildung $\Psi : Q \to D$. Definiere für jedes $q' \in Q'$ die Abbildung $\psi^{q',d_{q'}} : \mathtt{U}(q', G_Q) \cup \{q'\} \to \mathtt{U}(\Psi(q'), G_D) \cup \{\Psi(q')\}$ durch $\psi^{q',d_{q'}} := \Psi \downarrow \mathtt{U}(q', G_Q) \cup \{q'\}$. Dann ergeben sich die aufgeführten Bedingungen aus den Eigenschaften der Trefferabbildung $\Psi$ (Satz 3.3.5).
"⇐": Die Trefferabbildung $\Psi : Q \to D$ ergibt sich durch

$$\Psi(q) = \psi^{\alpha(q),d_q}(q). \tag{3.12}$$

Diese Abbildung ist wegen (3.11) wohldefiniert und ist bzgl. $G_Q$ und $Q'$ unabhängig von einem konkreten $\alpha$. Durch $\alpha$ wird nur ausgedrückt, durch welches $\psi^{q',d_{q'}}$, $q' \in Q'$, das Bild eines Anfrageobjektes $q \in Q$ definiert ist, falls sich hierfür mehrere $q' \in Q'$ anbieten. Gleichung (3.11) garantiert, daß hierfür alle Bilder $\psi^{q',d_{q'}}(q)$ gleich sind. Für I-Treffer und IC-Treffer gilt weiterhin die Äquivalenz zwischen $(a, b) \in E_Q$ und $(\Psi(a), \Psi(b)) \in E_D$ bzgl. $\psi^{\alpha(x),d_x}(x)$ und $E_D^{\psi^{\alpha(x),d_x}(x)}$ für $x \in \{a, b\}$. Hierbei stellen die Zusatzbedingungen bzgl. der Nachbarschaften für $q \in Q \setminus Q'$ sicher, daß die Rückrichtung gilt. Da $Q'$ so gewählt wurde, daß jede Kante aus $E_Q$ mit mindestens einem Knoten aus $Q'$ verbunden ist, gilt dies auch für Kanten in $\Psi(Q)$, da die Nachbarschaften entsprechend eingeschränkt wurden. Eine Kante zwischen zwei Knoten aus $\Psi(Q \setminus Q')$ wird somit ausgeschlossen. Abbildung 3.7 demonstriert dies. Für IC-Treffer gilt dies für den kompletten Graphen, für I-Treffer eingeschränkt auf $\Psi(Q)$.

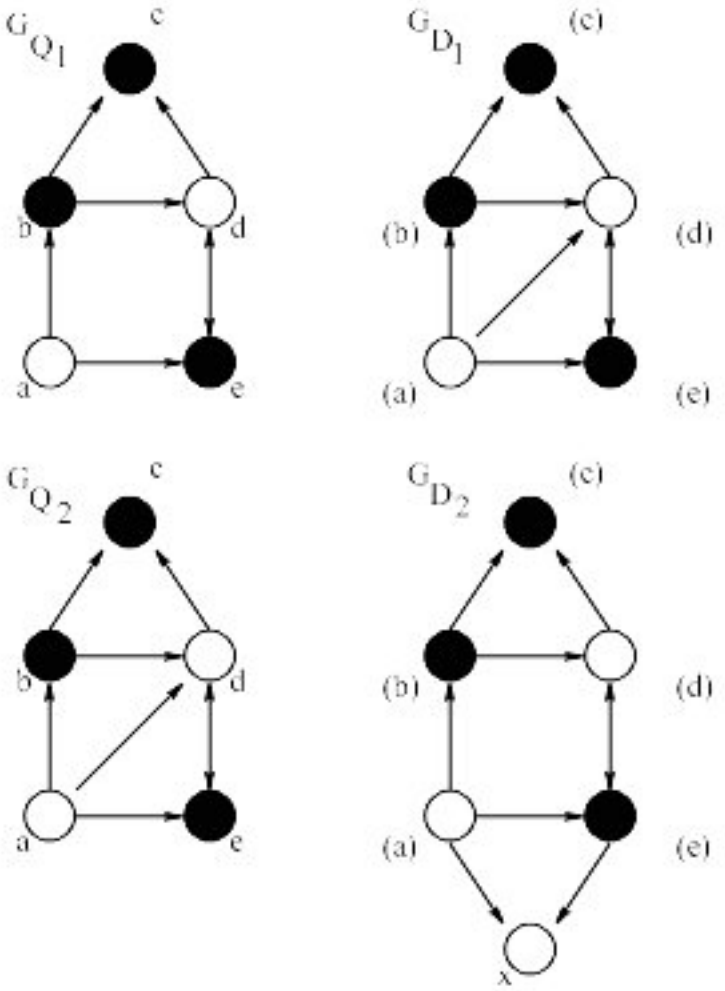



Abbildung 3.7: Zwei Anfragen und zwei Dokumente, die in Beispiel 3.3.10 besprochen werden. Es wird davon ausgegangen, daß alle Kantenbeschriftungen gleich sind. Ausgefüllte Punkte markieren die Knotenmenge $Q' = \{b, c, e\}$

.

Da für SC-Treffer und S-Treffer nur $(a,b) \in E_Q \Rightarrow (\Psi(a),\Psi(b)) \in E_D$ gelten muß, ändert sich die Beweisführung aus Satz 3.3.8 bis auf die oben angesprochenen Änderungen nicht. □

Das folgende Beispiel veranschaulicht die Aussagen des vorangegangen Satzes.

**Beispiel 3.3.10.** In Abbildung 3.7 sind zwei Anfragen $G_{Q_1}$ und $G_{Q_2}$ und zwei Dokumente $G_{D_1}$ und $G_{D_2}$ dargestellt. Für $G_{Q_1}$ wurde als Basis $Q' = \{b,c,e\}$ gewählt. $G_{D_1}$ bildet einen SC-Treffer, aber wegen der Kante $(\Psi(a),\Psi(d))$ in $G_{D_1}$ zwischen den beiden Knoten aus $D_1 \setminus \Psi(V')$ keinen IC-Treffer oder I-Treffer. Dies wird durch die Bedingung $\forall q \in Q_1 \setminus Q' : |\mathtt{U}(q, G_{Q_1})| = \left|\mathtt{U}(\psi^{\alpha(q),d_q}(q), G_{D_1}) \cap \cup_{q' \in Q'} \psi^{q',d_{q'}}(Q^{q'})\right|$ sichergestellt. Für $a$ mit $\alpha(a) = b$ ergibt sich $\mathtt{U}(a, G_{Q_1}) = \{b,e\}$ und $\mathtt{U}(\psi^{b,d_b}(a)) = \{\Psi(b),\Psi(d),\Psi(e)\}$, d.h. die Kardinalitäten beider Mengen sind verschieden.

$G_{D_2}$ ist I-Treffer bzgl. $G_{Q_1}$, da für I-Treffer die Kardinalitäten der Umgebungen nur bezogen auf $\Psi(Q_1)$ und nicht auf das komplette $D_2$ betrachtet werden. Damit wird der mit $x$ markierte Knoten nicht berücksichtigt, d.h. $x \notin \Psi(Q_1)$.

$G_{Q_2}$ ist eine Erweiterung von $G_{Q_1}$ um die Kante $(a,d)$. Allerdings erfüllt hier $Q'$ nicht mehr Bedingung (3.9), d.h. $Q'$ ist keine Basis von $G_{Q_2}$. ∘

Die konstruktiven Beweise der letzten beiden Sätze liefern ein erstes Vorgehen zur Berechnung aller Treffer. Bei Satz 3.3.8 wird für jedes Anfrageobjekt ein Vorkommen benötigt. Bei einer Anwendung von Satz 3.3.9 genügt es, wenn für eine Basis $Q'$ von $G_Q$ alle Vorkommen gegeben sind.

Nun betrachtet man jede Kombination von Vorkommen bzgl. Knoten aus $Q$ bzw. $Q'$. Für diese überprüft man die Bedingungen aus Satz 3.3.8 bzw. Satz 3.3.9 und erhält mittels (3.4) bzw. (3.12) alle Trefferabbildungen.

Bisher wurde nicht auf eine Berechnung der Vorkommen eingegangen. Dies wird in den nächsten Abschnitten besprochen. Dabei wird nicht nur von einem Dokument ausgegangen, in dem alle Treffer berechnet werden sollen, sondern von einer Datenbank mit vielen Dokumenten, in denen alle Treffer gesucht sind.

### 3.3.4 Vokabular

In den nächsten beiden Abschnitten wird eine Indexstruktur definiert, mit dessen Hilfe zu einem gegebenen Objekt $q$ einer Anfrage $Q$ alle Vorkommen von $q$ in allen Dokumenten einer Datenbank berechnet werden. Zusätzlich legt ein Benutzer für seine Anfrage fest, bzgl. welcher Trefferart die Treffer berechnet werden sollen. Der Index soll also für alle vier Trefferarten bzw. drei Arten von Vorkommen anwendbar sein.

Als Indexstruktur wird eine Variante bzw. Erweiterung von invertierten Listen verwendet, deren allgemeines Konzept in Abschnitt 3.2.1 motiviert wurde. In diesem Abschnitt wird das Vokabular, im nächsten Abschnitt werden invertierte Listen definiert. Jedes Objekt $d$ eines Dokumentes erzeugt einen Eintrag in genau einer invertierten Liste. In welcher invertierten Liste der Eintrag gespeichert wird, entscheidet die zu $d$ gehörige Vokabel.

Die Kombination aus Vokabel und Listeneintrag kodiert die Umgebung (Nachbarschaft) von $d \in D$ in $G_D$, d.h. welche Nachbarknoten von $d$ wie mit $d$ verbunden sind.

Dies wird an dem in Abbildung 3.8 dargestellten einfachen Beispiel $a$) motiviert. Dort besteht die Vokabel von $d \in D$ aus den Informationen

1. $d$ besitzt einen Nachfolger, der kein Vorgänger von $d$ ist, wobei die dazugehörige von $d$ ausgehende Kante mit $\alpha$ beschriftet ist.

2. $d$ besitzt einen Vorgänger, der kein Nachfolger von $d$ ist. Die dazugehörige in $d$ eingehende Kante ist mit $\beta$ beschriftet.

3. $d$ besitzt einen Nachfolger, der auch gleichzeitig Vorgänger von $d$ ist. Die dazugehörige von $d$ ausgehende Kante ist mit $\gamma$ und die in $d$ eingehende Kante ist mit $\delta$ beschriftet.

In dem zu $d$ gehörigen Eintrag in einer invertierten Liste wird abgespeichert, welche Nachbarknoten den oben aufgelisteten Kanten entsprechen, d.h. $a$ ist Nachfolger aber kein Vorgänger von $d$ mit Kantenbeschriftung $\alpha$. Analog wird abgespeichert, daß $b$ Vorgänger aber nicht Nachfolger von $d$ mit Kantenbeschriftung $\beta$ ist und $c$ sowohl Nachfolger als auch Vorgänger von $d$ mit den Kantenbeschriftungen $\xi_D((d,c)) = \gamma$ und $\xi_D((c,d)) = \delta$ ist.

Zur Bestimmung der Vokabeln und Listeneinträge werden die Nachbarknoten eines Objektes $d \in D$ in "nur" Nachfolgern von $d$ in $D$, d.h. $\mathtt{N}(d, G_D) \setminus \mathtt{V}(d, G_D)$, den "nur" Vorgängern von $d$ in $D$, d.h. $\mathtt{V}(d, G_D) \setminus \mathtt{N}(d, G_D)$, und den Nachfolgern, die auch Vorgänger von $d$ sind, d.h. $\mathtt{N}(d, G_D) \cap \mathtt{V}(d, G_D)$, klassifiziert.

Für jede dieser Klasse von Nachbarknoten wird in der Vokabel $\mathtt{v}(d, G_D)$ bzgl. $d$ und den entsprechenden eingehenden und ausgehenden Kanten festgehalten, welche Kantenbeschriftung wie oft vorkommt. Für das in Abbildung 3.8 abgebildete Beispiel $b$) ergibt sich, daß $d$ nur Nachfolger besitzt und die Kantenbeschriftung $\alpha$ zweimal und die Kantenbeschriftung $\beta$ einmal vorkommt.

**Definition 3.3.11.** Für einen Knoten $d \in D$ und eine Kantenbeschriftung $\alpha$ bezeichnet $\mathtt{V}(d, G_D, \alpha)$ die Menge der Vorgänger $c$ von $d$, deren Kante $(c, d)$ mit $\alpha$ beschriftet ist, d.h.

$$\mathtt{V}(d, G_D, \alpha) := \{c \in \mathtt{V}(d, G_D) | \xi_D((c,d)) = \alpha\}.$$

Diese Menge wird weiter auf diejenigen Vorgänger von $d$ eingeschränkt, die keine Nachfolger von $d$ sind, d.h.

$$\mathtt{V}^{\setminus \mathtt{N}}(d, G_D, \alpha) := \mathtt{V}(d, G_D, \alpha) \setminus \mathtt{N}(d, G_D).$$

Analog ergibt sich für Nachfolger bzw. Nachfolger, die keine Vorgänger sind,

$$\mathtt{N}(d, G_D, \alpha) := \{e \in \mathtt{N}(d, G_D) | \xi_D((d,e)) = \alpha\},$$
$$\mathtt{N}^{\setminus \mathtt{V}}(d, G_D, \alpha) := \mathtt{N}(d, G_D, \alpha) \setminus \mathtt{V}(d, G_D)$$

und für Nachfolger, die gleichzeitig Vorgänger sind,

$$\mathtt{NV}(d, G_D, \gamma, \delta) := \{a \in \mathtt{N}(d, G_D) \cap \mathtt{V}(d, G_D) | \xi_D((d,a)) = \gamma \wedge \xi_D((a,d)) = \delta\}.$$

Die von einem Objekt (Knoten) $d \in D$ erzeugte Vokabel ergibt sich durch die drei im folgenden definierten Mengen, die sich bzgl. den "nur" Nachfolgern, den "nur" Vorgängern und den Knoten, die gleichzeitig Vorgänger und Nachfolger sind, ergeben.
Für $d$ wird die Menge aller Kantenbeschriftungen $\xi_D((d,e))$ der ausgehenden Kanten festgehalten, wobei $e$ ein Nachfolger aus $\mathtt{N}(d, G_D) \setminus \mathtt{V}(d, G_D)$ ist. Zusätzlich wird für jede Kantenbeschriftung $\alpha$ die Anzahl $x_\alpha$ der mit $\alpha$ beschrifteten Kanten gespeichert. Insgesamt ergibt sich eine Menge von Paaren $(\alpha, x_\alpha) \in W_E \times \mathbb{N}$, wobei $W_E$ die Menge aller möglichen Kantenbeschriftungen ist. Als Schreibweise ergibt sich:

$$\mathtt{v}^{\mathtt{N} \setminus \mathtt{V}}(d, G_D) := \Big\{ \big(\alpha, \big|\mathtt{N}^{\setminus \mathtt{V}}(d, G_D, \alpha)\big| \big) \;\Big|\; \mathtt{N}^{\setminus \mathtt{V}}(d, G_D, \alpha) \neq \emptyset,\ \alpha \in W_E \Big\}. \tag{3.13}$$

Für alle Vorgänger $c \in \mathtt{V}(d, G_D) \setminus \mathtt{N}(d, G_D)$ geschieht dies analog:

$$\mathtt{v}^{\mathtt{V} \setminus \mathtt{N}}(d, G_D) := \Big\{ \big(\beta, \big|\mathtt{V}^{\setminus \mathtt{N}}(d, G_D, \beta)\big| \big) \;\Big|\; \mathtt{V}^{\setminus \mathtt{N}}(d, G_D, \beta) \neq \emptyset,\ \beta \in W_E \Big\}. \tag{3.14}$$

Schließlich werden für alle Knoten, die sowohl Vorgänger als auch Nachfolger von $d$ sind, eine Menge von Tripeln generiert. Hier werden zwei Tupeleinträge für Kantenbeschriftungen benötigt: jeweils einer pro Kante.

$$\mathtt{v}^{\mathtt{N} \cap \mathtt{V}}(d, G_D) := \Big\{ \big(\gamma, \delta, |\mathtt{NV}(d, G_D, \gamma, \delta)| \big) \;\Big|\; \mathtt{NV}(d, G_D, \gamma, \delta) \neq \emptyset,\ \gamma, \delta \in W_E \Big\}. \tag{3.15}$$

Die oben definierten drei Mengen werden in den entsprechenden Vokabeln zusammengefaßt.

**Definition 3.3.12.** Die **Vokabel** $\mathtt{v}(d, G_D)$ eines Objektes $d \in D$ wird definiert durch

$$\mathtt{v}(d, G_D) := \big(\mathtt{v}^{\mathtt{N} \setminus \mathtt{V}}(d, G_D), \mathtt{v}^{\mathtt{V} \setminus \mathtt{N}}(d, G_D), \mathtt{v}^{\mathtt{N} \cap \mathtt{V}}(d, G_D)\big). \tag{3.16}$$

Eine Vokabel ist also ein Tupel aus drei Mengen, d.h. $\mathtt{v}(d, G_D) \in 2^{W_E \times \mathbb{N}} \times 2^{W_E \times \mathbb{N}} \times 2^{W_E \times W_E \times \mathbb{N}}$. Da alle Dokumente bzw. Graphen zusammenhängend sind, ist mindestens eine der drei Mengen nicht leer. Weiterhin gilt nach Konstruktion:

$$\forall(\alpha, x), (\alpha', x') \in \pi_1(\mathtt{v}(d, G_D)) : \alpha = \alpha' \Rightarrow x = x'.$$

Für eine Kantenbeschriftung $\alpha$ existiert maximal ein Tupel in $\pi_1(\mathtt{v}(d, G_D))$. Analoges gilt für $\pi_2(\mathtt{v}(d, G_D))$. Für alle $(\gamma, \delta, z), (\gamma', \delta', z') \in \pi_3(\mathtt{v}(d, G_D))$ ergibt sich: $\gamma = \gamma' \wedge \delta = \delta' \Rightarrow z = z'$.

**Beispiel 3.3.13.** In Abbildung 3.9 ist ein Graph $G_D$ mit $D = \{a, b, c, d, e, f, g\}$ abgebildet. Kantenbeschriftungen sind durch griechische Buchstaben dargestellt. Im folgenden werden die Vokabeln der einzelnen Elemente von $D$ besprochen.

$a$ besitzt einen Vorgänger $d$ mit der Kantenbeschriftung $\xi_D((d, a)) = \alpha$. Dies führt zum einzigen Eintrag in der Menge $\mathtt{v}^{\mathtt{V}\setminus\mathtt{N}}(a, G_D) = \{(\alpha, 1)\}$, d.h. $a$ besitzt genau einen Vorgänger, der mit einer Kante mit der Beschriftung $\alpha$ verbunden ist und gleichzeitig kein Nachfolger ist. Als Vokabel ergibt sich, da sowohl $\mathtt{v}^{\mathtt{N}\setminus\mathtt{V}}(a, G_D)$ als auch $\mathtt{v}^{\mathtt{N}\cap\mathtt{V}}(a, G_D)$ leer sind, $\mathtt{v}(a, G_D) = \big(\emptyset, \{(\alpha, 1)\}, \emptyset\big)$.

$g$ besitzt zwei Nachfolger, $d$ und $f$, mit jeweils unterschiedlichen Kantenbeschriftungen, $\alpha$ bzw. $\gamma$. Daraus ergibt sich $\mathtt{v}(g, G_D) = \big(\{(\alpha, 1), (\gamma, 1)\}, \emptyset, \emptyset\big)$. Jeder Nachfolger hat einen Eintrag in $\mathtt{v}^{\mathtt{N}\setminus\mathtt{V}}(g, G_D)$ erzeugt.

$c$ hat zwei Vorgänger, $d$ und $e$, deren Kanten die gleiche Beschriftung $\alpha$ aufweisen. Dadurch besteht die Menge $\mathtt{v}^{\mathtt{V}\setminus\mathtt{N}}(c, G_D)$ aus dem Eintrag $(\alpha, 2)$. Dieser besagt, daß zwei Vorgänger mit der gleichen Kantenbeschriftung $\alpha$ existieren. $\mathtt{v}(c, G_D)$ ist gleich $\big(\emptyset, \{(\alpha, 2)\}, \emptyset\big)$.

$e$ besitzt einen Vorgänger $f$ mit $\xi_D((f, e)) = \gamma$ und einen Nachfolger $c$ mit $\xi_D((e, c)) = \alpha$. Somit ergibt sich $\mathtt{v}^{\mathtt{N}\setminus\mathtt{V}}(e, G_D) = \{(\alpha, 1)\}$ und $\mathtt{v}^{\mathtt{V}\setminus\mathtt{N}}(e, G_D) = \{(\gamma, 1)\}$. Die Vokabel $\mathtt{v}(e, G_D)$ ist damit gleich $\big(\{(\alpha, 1)\}, \{(\gamma, 1)\}, \emptyset\big)$.

$b$ besitzt einen Vorgänger $d$, der auch gleichzeitig der einzige Nachfolger von $b$ ist. Beide Kantenbeschriftungen werden im Tripel $(\alpha, \beta, 1)$ eingetragen, das das einzige Element der Menge $\mathtt{v}^{\mathtt{N}\cap\mathtt{V}}(b, G_D)$ ist. $\mathtt{v}(b, G_D)$ ist gleich $\big(\emptyset, \emptyset, \{(\alpha, \beta, 1)\}\big)$.

Schließlich ergeben sich für $f$ bzw. $d$ die Vokabeln $\mathtt{v}(f, G_D) = \big(\{(\gamma, 1)\}, \{(\alpha, 1)\}, \{(\beta, \beta, 1)\}\big)$ und $\mathtt{v}(d, G_D) = \big(\{(\alpha, 2)\}, \{(\gamma, 1)\}, \{(\beta, \alpha, 1), (\beta, \beta, 1)\}\big)$. ∘

**Definition 3.3.14.** Die Menge aller Vokabeln bzgl. aller Objekte aus Dokumenten einer Datenbasis $\mathcal{D}$ wird als **Vokabular** $\mathcal{V}$ bezeichnet:

$$\mathcal{V} := \{\mathtt{v}(d, G_D) | d \in D \wedge D \in \mathcal{D}\}.$$

Im folgenden wird die oben getroffene Wahl der Vokabeln motiviert. Sicherlich existieren noch alternative Möglichkeiten, die Vokabel für ein Objekt $d \in D$ zu definieren. Wie sich herausstellen wird, besitzen die Vokabeln nach Definition 3.3.12 eine Eigenschaft, die speziell auf S-Vorkommen abgestimmt ist. Auch für die anderen Vorkommenarten bringt diese Eigenschaft praktische Vorteile. Dies wird später bei der Trefferberechnung deutlich, wenn für ein Anfrageobjekt $q \in Q$ abhängig von der Trefferart im Vokabular nach "passenden" Vokabeln $v \in \mathcal{V}$ gesucht wird. Hierauf wird in Abschnitt 3.3.6 eingegangen.

In den letzten Abschnitten wurde auf die lokale Sichtweise in einem Dokument $G_D$ für ein Objekt $d \in D$ im Zusammenhang mit Vorkommen eingegangen. Hierbei wurde für $d \in D$ der Teilgraph $G_D^d$ von $G_D$ definiert, der nur aus Nachbarknoten von $d$ und den entsprechenden Kanten von und nach $d$ besteht. Dies wird durch ein Paar $(d, G_D^d)$ ausgedrückt, wobei $G_D$ ein Dokument und $d \in D$ ist.

Speziell für den Vergleich von Vorkommen wird die Relation $\leq_T$ auf der Menge der möglichen Paare $(d, G_D^d)$ für jede Vorkommenart $T \in \{IC, I, S\}$ eingeführt.

**Definition 3.3.15.** Seien $G_D$ und $G_Q$ zwei Dokumente und $q \in Q$ sowie $d \in D$ zwei Knoten. Dann gilt $(q, G_Q^q) \leq_T (d, G_D^d)$ genau dann, wenn $d$ ein $T$-Vorkommen von $q$ ist.

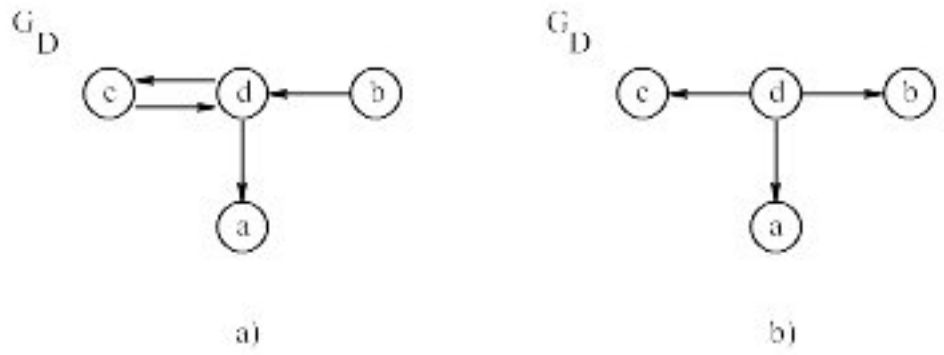



Abbildung 3.8: Zwei einfache Beispiele als Motivation für die Definition der Vokabeln. Im linken Beispiel ergibt sich die Vokabel für $d$ als $\mathtt{v}(d, D) = \big(\{(\alpha, 1)\}, \{(\beta, 1)\}, \{(\gamma, \delta, 1)\}\big)$, im rechten Beispiel ist $\mathtt{v}(d, D) = \big(\{(\alpha, 2), (\beta, 1)\}, \emptyset, \emptyset\big)$.

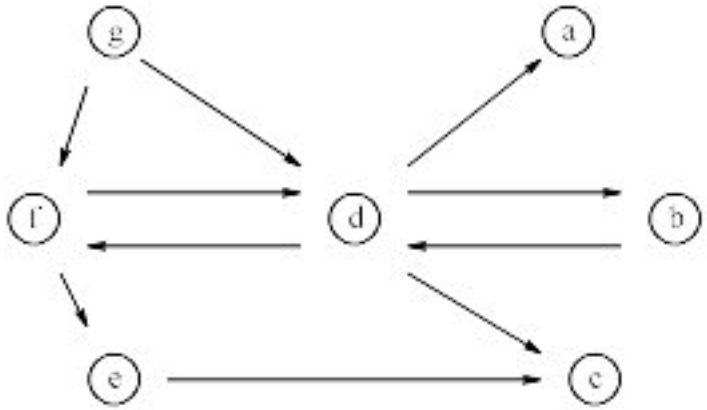



Abbildung 3.9: Beispiel zur Berechnung von Vokabeln. Für die einzelnen Objekte ergeben sich: $\mathtt{v}(a, G_D) = \big(\emptyset, \{(\alpha, 1)\}, \emptyset\big)$, $\mathtt{v}(b, G_D) = \big(\emptyset, \emptyset, \{(\alpha, \beta, 1)\}\big)$, $\mathtt{v}(c, G_D) = \big(\emptyset, \{(\alpha, 2)\}, \emptyset\big)$, $\mathtt{v}(d, G_D) = \big(\{(\alpha, 2)\}, \{(\gamma, 1)\}, \{(\beta, \alpha, 1), (\beta, \beta, 1)\}\big)$, $\mathtt{v}(e, G_D) = \big(\{(\alpha, 1)\}, \{(\gamma, 1)\}, \emptyset\big)$, $\mathtt{v}(f, G_D) = \big(\{(\gamma, 1)\}, \{(\alpha, 1)\}, \{(\beta, \beta, 1)\}\big)$ und $\mathtt{v}(g, G_D) = \big(\{(\alpha, 1), (\gamma, 1)\}, \emptyset, \emptyset\big)$.

Sei $T \in \{IC, I, S\}$. Die Relation $\leq_T$ ist offensichtlich reflexiv. Sie ist auch transitiv, da man die entsprechenden Trefferabbildungen einfach verknüpfen kann. Allerdings ist die Relation $\leq_T$ *nicht* antisymmetrisch. So gilt z.B. für zwei isomorphe Graphen $G_D$ und $G_Q$ mit den entsprechenden Knoten $q \in Q$, $d \in D$: $(q, G_Q^q) \leq_T (d, G_D^d)$ und $(d, G_D^d) \leq_T (q, G_Q^q)$.

**Beispiel 3.3.16.** Betrachten wir die Dokumente aus Abbildung 3.10, so gilt $(q, G_{Q_1}^q) \leq_S (d, G_{D_1}^d)$ für $q \in Q_1$ und $d \in D_1$, da $d$ ein S-Vorkommen von $q$ ist. $(d, G_{D_1}^d) \leq_S (q, G_{Q_1}^q)$ gilt nicht.

Für die Dokumente $G_{Q_2}$ und $G_{D_2}$ gilt sowohl $(t, G_{Q_2}^t) \leq_S (a, G_{D_2}^a)$ als auch $(a, G_{D_2}^a) \leq_S (t, G_{Q_2}^t)$.

Untersucht man die beiden Dokumente $G_{Q_1}$ und $G_{D_2}$ aus Abbildung 3.10, so gilt für $T = I$: $(q, G_{Q_1}^q) \leq_T (a, G_{D_2}^a)$. Allerdings gilt $(a, G_{D_2}^a) \leq_T (q, G_{Q_1}^q)$ nicht. ∘

Keine der Relationen $\leq_T$ bildet eine Partialordnung. Allerdings läßt sich eine Äquivalenzrelation $\sim$ definieren, die auf den Relationen $\leq_T$ aufbaut, und die schließlich zu Partialordnungen auf der Menge der Äquivalenzklassen führen.

**Definition 3.3.17.** Seien $G_D$ und $G_Q$ zwei Dokumente mit $q \in Q$ und $d \in D$. Weiterhin sei die Relation $\leq_S$ definiert nach Definition 3.3.15. Dann gilt $(q, G_Q^q) \sim (d, G_D^d)$ genau dann, wenn $(q, G_Q^q) \leq_S (d, G_D^d)$ und $(d, G_D^d) \leq_S (q, G_Q^q)$ ist.

Zwei Paare $(q, G_Q^q)$ und $(d, G_D^d)$ sind also $\sim$-äquivalent, wenn $q$ ein S-Vorkommen von $d$ und $d$ ein S-Vorkommen von $q$ ist. Als Bezeichnung für die Äquivalenzklassen wird statt $[(d, G_D^d)]_\sim$ nur $[d, G_D^d]$ geschrieben.

Die letzte Definition ist unabhängig von der gewählten Vorkommenart. Definiert man $\sim$ für $\leq_{IC}$ oder $\leq_I$ statt für $\leq_S$, so ergeben sich die gleichen Äquivalenzklassen. Der nächste Satz verdeutlicht dies.

**Satz 3.3.18.** *Für jede $\sim$-Äquivalenzklasse $[d, G_D^d]$ gilt: $(q, G_Q^q)$ und $(d, G_D^d)$ sind genau dann zwei Elemente von $[d, G_D^d]$, wenn die Vokabeln von $q$ und $d$ gleich sind, d.h. $\mathtt{v}(d, G_D^d) = \mathtt{v}(q, G_Q^q)$.*

**Beweis** Der Satz ergibt sich als direkte Folgerung aus Definition 3.3.15, Definition 3.3.17 und der Definition der Vokabeln. □

Dabei entspricht für $d \in D$ die "lokale" Vokabel bzgl. $G_D^d$ der "globalen" Vokabel bzgl. $G_D$, denn $\mathtt{v}(d, G_D) = \mathtt{v}(d, G_D^d)$.

Der letzte Satz dient als Motivation für die gewählte Definition der Vokabeln. Sie ist speziell an die Bedürfnisse der Trefferberechnung angepaßt.

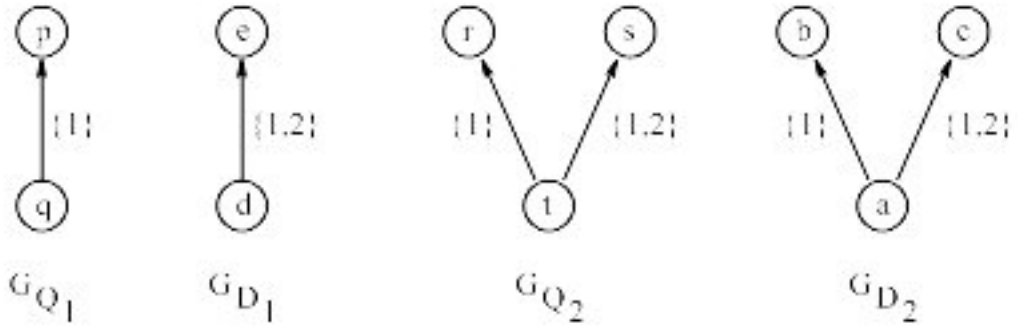



Abbildung 3.10: Vier beschriftete Digraphen zur Demonstration von partiellen Ordnungen.

**Beispiel 3.3.19.** Betrachten wir nochmals die Dokumente aus Abbildung 3.10, auf die bereits in Beispiel 3.3.16 eingegangen wurde. Für die ersten beiden Dokumente gilt $(q, G^q_{Q_1}) \sim (d, G^d_{D_1})$ nicht. Die Objekte $d$ und $q$ besitzen unterschiedliche Vokabeln, denn $\mathtt{v}(q, G^q_{Q_1}) = \big(\{(\{1\}, 1)\}, \emptyset, \emptyset\big)$, aber $\mathtt{v}(d, G^d_{D_1}) = \big(\{(\{1, 2\}, 1)\}, \emptyset, \emptyset\big)$.

Für die nächsten beiden Dokumente gilt $(t, G^t_{Q_2}) \sim (a, G^a_{D_2})$, denn $\mathtt{v}(t, G^t_{Q_2}) = \mathtt{v}(a, G^a_{D_2}) = \big(\{(\{1\}, 1), (\{1, 2\}, 1)\}, \emptyset, \emptyset\big)$. ∘

Das letzte Beispiel demonstriert auch die Unabhängigkeit der Definition von $\sim$ bzgl. einer bestimmten Vorkommenart $T \in \{IC, I, S\}$ und $\leq_T$.

Die Menge der $\sim$-Äquivalenzklassen wird also charakterisiert durch die Menge der Vokabeln. Für jede Vorkommenart wird nun eine partielle Ordnung auf dem Vokabular definiert. Diese wird für eine spätere Suche im Vokabular ausgenutzt.

**Definition 3.3.20.** (1) Seien $[d, G^d_D]$ und $[q, G^q_Q]$ zwei $\sim$-Äquivalenzklassen und $T \in \{IC, I, S\}$ eine Vorkommenart. Dann gilt $[q, G^q_Q] \sqsubseteq_T [d, G^d_D]$ genau dann, wenn $(q, G^q_Q) \leq_T (d, G^d_D)$ ist.
(2) Sei $\mathcal{V}$ ein Vokabular einer Datenbasis $\mathcal{D}$. Dann gilt $v \sqsubseteq^{\mathcal{V}}_T w$ für zwei Vokabeln $v, w \in \mathcal{V}$ genau dann, wenn $[q, G^q_Q] \sqsubseteq_T [d, G^d_D]$ für zwei Paare $(q, G^q_Q)$, $(d, G^d_D)$ mit $v = \mathtt{v}(q, G^q_Q)$ und $w = \mathtt{v}(d, G^d_D)$ gilt.

Die partielle Ordnung $\sqsubseteq_T$ ist wohldefiniert, da sie bzgl. $\leq_T$ unabhängig von den gewählten Repräsentanten der Äquivalenzklassen ist. Mit der partiellen Ordnung $\sqsubseteq^{\mathcal{V}}_T$ lassen sich die Vokabeln hierarchisch anordnen.

Im Kontext der letzten Beispiele bzgl. Abbildung 3.10 ergibt sich für $T = S$

$$\begin{array}{rrrl} [q, G^q_{Q_1}] & \sqsubseteq_S & [d, G^d_{D_1}] & \sqsubseteq_S [a, G^a_{D_2}] \text{ und} \\ \mathtt{v}(q, G^q_{Q_1}) & \sqsubseteq^{\mathcal{V}}_S & \mathtt{v}(d, G^d_{D_1}) & \sqsubseteq^{\mathcal{V}}_S \mathtt{v}(a, G^a_{D_2}), \end{array}$$

sowie für $T = I$

$$\begin{array}{rrl} [d, G^d_{D_1}] & \sqsubseteq_I & [a, G^a_{D_2}] \text{ und} \\ \mathtt{v}(d, G^d_{D_1}) & \sqsubseteq^{\mathcal{V}}_I & \mathtt{v}(a, G^a_{D_2}). \end{array}$$

Für $T = I$ gilt $\mathtt{v}(q, G^q_{Q_1}) \sqsubseteq^{\mathcal{V}}_T \mathtt{v}(d, G^d_{D_1})$ nicht. Dies demonstriert den Unterschied zwischen den partiellen Ordnungen $\sqsubseteq_I$ und $\sqsubseteq_S$.

Für $T = IC$ gilt $[q, G^q_Q] \sqsubseteq_{IC} [d, G^d_D]$ genau dann, wenn $[q, G^q_Q] = [d, G^d_D]$ ist.

Betrachten wir nun den konkreten Zusammenhang zwischen den partiellen Ordnungen auf dem Vokabular und Vorkommen.

**Satz 3.3.21.** *Seien $G_Q, G_D$ zwei Dokumente, $q \in Q, d \in D$ und $T \in \{IC, I, SC\}$. Genau dann ist $d$ ein $T$-Vorkommen von $q$, wenn $\mathtt{v}(q, G_Q) \sqsubseteq^{\mathcal{V}}_T \mathtt{v}(d, G_D)$ gilt.*

**Beweis** Dies ergibt sich als direkte Folgerung aus der Definition der partiellen Ordnung $\sqsubseteq^{\mathcal{V}}_T$. □

Die letzten Beispiele demonstrieren den Satz: $d \in D_1$ ist S-Vorkommen von $q \in Q_1$ und es gilt $[q, G^q_{Q_1}] \sqsubseteq_S [d, G^d_{D_1}]$.

Vergleicht man die drei partiellen Ordnungen auf dem Vokabular untereinander, so ergibt sich eine Inklusionsbeziehung wie bei den entsprechenden Trefferbegriffen.

**Satz 3.3.22.** *Seien $G_Q, G_D$ zwei Dokumente und $q \in Q, d \in D$. Dann gilt*

$$\mathtt{v}(q, G_Q) \sqsubseteq^{\mathcal{V}}_{IC} \mathtt{v}(d, G_D) \;\Rightarrow\; \mathtt{v}(q, G_Q) \sqsubseteq^{\mathcal{V}}_I \mathtt{v}(d, G_D) \;\Rightarrow\; \mathtt{v}(q, G_Q) \sqsubseteq^{\mathcal{V}}_S \mathtt{v}(d, G_D).$$

**Beweis** Die Aussage folgt aus

$$[q, G_Q^q] \sqsubseteq_{IC} [d, G_D^d] \;\Rightarrow\; [q, G_Q^q] \sqsubseteq_I [d, G_D^d] \;\Rightarrow\; [q, G_Q^q] \sqsubseteq_S [d, G_D^d]$$

und den entsprechenden Inklusionen bzgl. der Vorkommenarten. □

Für die Beispiele aus Abbildung 3.10 gilt

$$\mathtt{v}(q, G_{Q_1}) \sqsubseteq_T^{\mathcal{V}} \mathtt{v}(a, G_{D_2}) \;\Rightarrow\; \mathtt{v}(q, G_{Q_1}) \sqsubseteq_S^{\mathcal{V}} \mathtt{v}(a, G_{D_2}).$$

### 3.3.5 Invertierte Listen

Nachdem im letzten Abschnitt die Vokabel für ein Objekt $d$ aus einem Dokument $D$ definiert wurde, wird nun auf invertierte Listen eingegangen. Dabei erzeugt jedes Objekt $d$ genau einen Eintrag in genau einer invertierten Liste. Die invertierte Liste wird durch die Vokabel von $d$ bestimmt.

Während in der Vokabel zu $d$ Informationen über die ein- und ausgehenden Kanten von $d$ gespeichert sind, werden in dem Listeneintrag von $d$ die dazugehörigen Nachbarobjekte von $d$ gespeichert.

Für Beispiel $a$) aus Abbildung 3.8 werden in dem Listeneintrag von $d$ folgende Objekte abgespeichert: Objekt $a$ als Nachfolger von $d$ mit der Kantenbeschriftung $\alpha$, Objekt $b$ als Vorgänger von $d$ mit Kantenbeschriftung $\beta$ und Objekt $c$, wobei $c$ Nachfolger und Vorgänger von $d$ mit den Kantenbeschriftungen $\gamma$ und $\delta$ ist.

**Definition 3.3.23.** Für ein Objekt $d \in D$ ist der **Listeneintrag** (kurz: Eintrag) $E(d, G_D)$ ein Fünftupel definiert durch:

$$\begin{aligned}[ \quad & d, \;\; D, \\ & \{\mathtt{N}^{\backslash \mathtt{V}}(d, G_D, \alpha) | (\alpha, x) \in \pi_1(\mathtt{v}(d, G_D))\}, \\ & \{\mathtt{V}^{\backslash \mathtt{N}}(d, G_D, \beta) | (\beta, y) \in \pi_2(\mathtt{v}(d, G_D))\}, \\ & \{\mathtt{NV}(d, G_D, \gamma, \delta) | (\gamma, \delta, z) \in \pi_3(\mathtt{v}(d, G_D))\} \quad ].\end{aligned}$$

Ein Listeneintrag besteht aus einem Objekt $d$, aus einem Dokument $G_D$ mit $d \in D$, und aus drei Mengen von Objektmengen. Hierbei sind $d$ und $D$ als Referenzen zu verstehen. Es wird nicht das komplette Objekt und nicht das komplette Dokument in einem Eintrag eingetragen. Vielmehr wird für $j \in [1 : |\mathcal{D}|]$ und $d_i \in D_j$ die Referenz $i$ in $\pi_1(E(d_i, G_{D_j}))$ und $j$ in $\pi_2(E(d_i, G_{D_j}))$ abgespeichert. Analoges gilt für die Elemente der Objektmengen aus $\pi_3(E(d_i, G_{D_j}))$, $\pi_4(E(d_i, G_{D_j}))$ und $\pi_5(E(d_i, G_{D_j}))$. Die in der Definition gewählte Schreibweise vereinfacht die Darstellung in vielen Fällen.

**Annahme 3.3.24.** O.B.d.A. wird bei einer Vokabel $\mathtt{v}(d, G_D)$ und einem Listeneintrag $E(d, G_D)$ davon ausgegangen, daß für $\pi_1(\mathtt{v}(d, G_D)) = \{(\alpha_1, x_1), \ldots, (\alpha_l, x_l)\}$ und $\pi_3(E(d, G_D)) = \{O_1, \ldots, O_l\}$

$$O_i = \mathtt{N}^{\backslash \mathtt{V}}(g, G_D, \alpha_i)$$

gilt. Analoges wird für $\pi_2(\mathtt{v}(d, G_D))$ und $\pi_4(E(d, G_D))$ sowie für $\pi_3(\mathtt{v}(d, G_D))$ und $\pi_5(E(d, G_D))$ angenommen. Hierdurch wird eine Zuordnung von Teilen der Vokabel $v$, z.B. $(\alpha, x) \in \pi_1(v)$, und Objektmengen im Eintrag, z.B. $O \in \pi_3(E), E \in L(v)$, sichergestellt.

Die Listeneinträge der einzelnen Objekte werden in Abhängigkeit von ihrer Vokabel in invertierte Listen eingetragen.

**Definition 3.3.25.** Für eine Datenbasis $\mathcal{D}$ ergibt sich für jede Vokabel $v \in \mathcal{V}$ die **invertierte Liste** $L_{\mathcal{D}}(v)$, kurz $L(v)$, durch

$$L(v) = \{E(d, G_D) | G_D \in \mathcal{D}, d \in D, v = \mathtt{v}(d, G_D)\}. \tag{3.17}$$

**Satz 3.3.26.** *Für einen Listeneintrag $E(d, D)$, $d \in D$, gilt:*

$$D^d \setminus \{d\} = \sqcup_{M \in \pi_i(E(d,D)), i \in [3:5]} M.$$

**Beweis** Diese Eigenschaft folgt direkt aus der Konstruktion (Definition 3.3.25 und Definition 3.3.23), da jedes Objekt $a \in D^d \setminus \{d\}$ entweder nur Nachfolger, nur Vorgänger oder Nachfolger und Vorgänger von $d$ ist und dementsprechend in genau einer der Mengen von Objektmengen $\pi_3(E(d, G_D))$, $\pi_4(E(d, G_D))$ oder $\pi_5(E(d, G_D))$ enthalten ist.

**Beispiel 3.3.27.** Für eine Datenbasis, die nur aus dem in Abbildung 3.8 b) dargestellten Dokument besteht, ergeben sich folgende invertierte Listen:

$$\begin{array}{lllllll}
L((\emptyset, \{(\alpha,1)\}, \emptyset)) = \{ & [a, & D, & \emptyset, & \{\{d\}\}, & \emptyset], & \\
 & [c, & D, & \emptyset, & \{\{d\}\}, & \emptyset] & \} \\
L((\emptyset, \{(\beta,1)\}, \emptyset)) = \{ & [b, & D, & \emptyset, & \{\{d\}\}, & \emptyset] & \} \\
L((\{(\alpha,2),(\beta,1)\}, \emptyset, \emptyset)) = \{ & [d, & D, & \{\{a,c\},\{b\}\}, & \emptyset, & \emptyset] & \}
\end{array}$$

Da $\mathtt{v}(a, G_D) = \mathtt{v}(c, G_D)$ ist, werden die Listeneinträge $E(a, G_D)$ und $E(c, G_D)$ in der gleichen invertierten Liste gespeichert. ∘

**Bemerkung 3.3.28.** Betrachtet man bzgl. einer Datenbasis alle invertierten Listen und ihre Einträge, so ist die Anzahl aller Listeneinträge gleich $\sum_{G_D \in \mathcal{D}} |D|$, da jedes Objekt genau einen Eintrag erzeugt. Insgesamt enthalten die Einträge aller invertierten Listen $2\sum_{D \in \mathcal{D}} |E_D|$ Objektreferenzen und $\sum_{G_D \in \mathcal{D}} |D|$ Dokumentreferenzen. $E_D$ ist die Kantenmenge des Graphen $G_D$.

### 3.3.6 Berechnung von Vorkommen

In diesem Abschnitt wird auf die Berechnung von Vorkommen mit Hilfe der im letzten Abschnitt vorgestellten Indexstruktur eingegangen. Zur Berechnung aller Vorkommen von $q \in Q$ wird als erstes die Vokabel von $q$ berechnet. Diese definiert die invertierten Listen, mit denen sich die lokalen Trefferabbildungen $\psi^{q,d_q}$ (Definition 3.3.7) bzgl. eines Vorkommen $d_q$ von $q$ angeben lassen.

Die Berechnung hängt neben der Anfrage von der gewählten Trefferart ab (vgl. Satz 3.3.5). Für IC-Vorkommen gestaltet sich die Berechnung einfach: Es muß in Abhängigkeit der Vokabel nur eine invertierte Liste betrachtet werden. Bei I-Vorkommen müssen mehrere invertierte Listen betrachtet werden. Noch komplexer ist dies bei S-Vorkommen.

Der folgende Satz zeigt für IC-Vorkommen und I-Vorkommen, welche invertierten Listen betrachtet werden müssen. Der Beweis des Satzes erfolgt konstruktiv und liefert alle möglichen Trefferabbildungen $\psi^{q,d_q}$. S-Vorkommen werden an späterer Stelle untersucht.

**Satz 3.3.29.** *Sei $G_Q$ eine Anfrage und $G_D$ ein Dokument. Dann gilt für alle $q$ aus $Q$ und $d$ aus $D$ mit den Vokabeln $v := \mathtt{v}(d, G_D)$ und $v^q := \mathtt{v}(q, G_Q)$:*

*(1) $d$ ist IC-Vorkommen von $q \Leftrightarrow v = v^q$.*

*(2) $d$ ist I-Vorkommen von $q \Leftrightarrow$*

$$\forall(\alpha, x^q) \in \pi_1(v^q) \quad \exists(\alpha, x^d) \in \pi_1(v) : \quad x^q \leq x^d \tag{3.18}$$

$$\forall(\beta, y^q) \in \pi_2(v^q) \quad \exists(\beta, y^d) \in \pi_2(v) : \quad y^q \leq y^d \tag{3.19}$$

$$\forall(\gamma, \delta, z^q) \in \pi_3(v^q) \quad \exists(\gamma, \delta, z^d) \in \pi_3(v) : \quad z^q \leq z^d. \tag{3.20}$$

Der letzte Satz zeigt, für welche $v \in \mathcal{V}$ die invertierten Listen $L(v)$ betrachtet werden müssen: Für IC-Vorkommen ist dies die invertierte Liste $L(\mathtt{v}(q, G_Q))$, falls $\mathtt{v}(q, G_Q) \in \mathcal{V}$. Jeder Eintrag in dieser Liste definiert ein IC-Vorkommen.

Für I-Vorkommen werden alle invertierten Listen $L(v)$ betrachtet, für die $v \in \mathcal{V}$ gilt und die (3.18), (3.19) und (3.20) erfüllen.

Für beide Arten von Vorkommen von $q$ gilt: Existiert kein $v \in \mathcal{V}$, das Satz 3.3.29 bzgl. $\mathtt{v}(q, G_Q)$ erfüllt, existiert kein Vorkommen von $q$ in Dokumenten der Datenbasis.

Die Aussagen des Satzes lassen sich auch im Zusammenhang mit den Relationen $\leq_I$ und $\sqsubseteq_I^{\mathcal{V}}$ verwenden. Hierauf wird in Abschnitt 4.1 eingegangen.

Teil des nun folgenden Beweises von Satz 3.3.29 ist die Konstruktion aller Trefferabbildungen $\psi^{q,d_q}$ für Vorkommen $d_q$ von $q$.

**Beweis Satz 3.3.29 (1)** "⇒": Betrachten wir die Graphen $G_Q^q$ und $G_D^d$ nach Definition 3.3.2. Für diese ist eine IC-Trefferabbildung $\Psi$ nach Satz 3.3.5 eine Bijektion zwischen $Q^q$ und $D^d$, die nach der Trefferdefinition auch eine Bijektion zwischen den Kanten impliziert (bei gleichen Kantenbeschriftungen). Dann sind nach Konstruktion (Definition 3.3.12) die Vokabeln $\mathtt{v}(q, G_Q)$ und $\mathtt{v}(d, G_D)$ gleich.
"⇐": Im folgenden wird eine Trefferabbildung $\Psi : Q^q \to D^d$ aus den Mengen $\{\mathtt{N}^{\setminus\mathtt{V}}(d, G_D, \alpha) | (\alpha, x) \in \pi_1(\mathtt{v}(d, G_D))\}$, $\{\mathtt{V}^{\setminus\mathtt{N}}(d, G_D, \beta) | (\beta, y) \in \pi_2(\mathtt{v}(d, G_D))\}$, und $\{\mathtt{NV}(d, G_D, \gamma, \delta) | (\gamma, \delta, z) \in \pi_3(\mathtt{v}(d, G_D))\}$ sowie den entsprechenden Mengen für $q$ bzgl. $\mathtt{v}(q, G_Q)$ konstruiert. Das sind genau die Mengen, die in den Listeneinträgen $E(q, G_Q)$ und $E(d, G_D)$ gespeichert sind. Genauer handelt es sich bei diesen drei Mengen um Mengen von Objektmengen. In diesen Mengen sind alle Nachbarn von $d$ bzw. $q$ gespeichert. Z.B. ist $\mathtt{N}^{\setminus\mathtt{V}}(d, G_D, \alpha)$ die Menge der "nur" Nachfolger von $d$, die mit einer Kante mit der Beschriftung $\alpha$ verbunden sind.

Betrachten wir zunächst den Fall, daß $\pi_2(\mathtt{v}(q, G_Q)) = \pi_3(\mathtt{v}(q, G_Q)) = \emptyset$. Hier besitzt $q$ sowohl in $G_Q$ als auch in $G_Q^q$ nur ausgehende Kanten. Da die Vokabeln von $q$ und $d$ gleich sind, gilt nach Definition 3.3.12: $\forall(\alpha, x) \in W_E \times \mathbb{N} : (\alpha, x) \in \pi_1(\mathtt{v}(d, G_D)) \Leftrightarrow (\alpha, x) \in \pi_1(\mathtt{v}(q, G_Q))$. Für ein solches $(\alpha, x) \in \pi_1(\mathtt{v}(d, G_D))$ sei $\mathtt{N}^{\setminus\mathtt{V}}(d, G_D, \alpha) = \{c_1, \ldots, c_x\}$ und $\mathtt{N}^{\setminus\mathtt{V}}(q, G_Q, \alpha) = \{p_1, \ldots, p_x\}$. Dann definiere simultan für alle solchen $(\alpha, x)$ die Abbildung $\Psi : Q^q \to D^d$ mittels $\Psi(p_i) = c_i$ für $i \in [1 : x]$ und $\Psi(q) = d$. $\Psi$ ist wegen 3.3.26 wohldefiniert. Es gilt $(q, c) \in E_Q \wedge \xi_Q((q, c)) = \alpha \Leftrightarrow (\Psi(q), \Psi(c)) \in E_D \wedge \xi_D(\Psi(q), \Psi(c)) = \alpha$. $\Psi$ ist also nach Konstruktion (Definition 3.3.12 und Definition 3.3.25) eine Trefferabbildung, d.h. $G_D^d$ ist IC-Treffer von $G_Q^q$. Nach Satz 3.3.5 bedeutet dies, daß $d$ ein IC-Vorkommen von $q$ ist.

Der allgemeine Fall ergibt sich aus dem Sonderfall, indem man $\Psi$ analog für $\pi_2(\mathtt{v}(q, G_Q))$ und $\pi_3(\mathtt{v}(q, G_Q))$ definiert. Es wird eine Bijektion $\Psi : Q^q \to D^d$ aus den folgenden drei injektiven Abbildungen konstruiert:

$$
\begin{array}{lcl}
\Psi_1 & : & \mathtt{N}(q, G_Q) \setminus \mathtt{V}(q, G_Q) \to \mathtt{N}(d, G_D) \setminus \mathtt{V}(d, G_D) \text{ mit } \Psi_1(p) \in \mathtt{N}^{\setminus\mathtt{V}}(d, G_D, \xi_Q((q, p))), \\
\Psi_2 & : & \mathtt{V}(q, G_Q) \setminus \mathtt{N}(q, G_Q) \to \mathtt{V}(d, G_D) \setminus \mathtt{N}(d, G_D) \text{ mit } \Psi_2(p) \in \mathtt{V}^{\setminus\mathtt{N}}(d, G_D, \xi_Q((q, p))), \\
\Psi_3 & : & \mathtt{N}(q, G_Q) \cap \mathtt{V}(q, G_Q) \to \mathtt{N}(d, G_D) \cap \mathtt{V}(d, G_D) \text{ mit} \\
 & & \Psi_3(p) \in \mathtt{NV}(d, G_D, \xi_Q((q, p)), \xi_Q((p, q))).
\end{array}
$$

Damit ergibt sich $\Psi : Q^q \to D^d$ als

$$
\Psi(p) := \begin{cases} d & \text{falls } p = q \\ \Psi_1(p) & \text{falls } p \in \mathtt{N}(q, G_Q) \setminus \mathtt{V}(q, G_Q) \\ \Psi_2(p) & \text{falls } p \in \mathtt{V}(q, G_Q) \setminus \mathtt{N}(q, G_Q) \\ \Psi_3(p) & \text{falls } p \in \mathtt{N}(q, G_Q) \cap \mathtt{V}(q, G_Q) \end{cases} \tag{3.21}
$$

Vor dem Beweis von (2) kommentieren wir die gerade bewiesene Aussage. Die Anzahl der möglichen Trefferabbildungen, die in obigen Beweis konstruiert wurden, ergibt sich aus:

**Bemerkung 3.3.30.** Für zwei Objekte $d \in D$ und $q \in Q$ mit gleichen Vokabeln $\mathtt{v}(q, G_Q) = \mathtt{v}(d, G_D)$ existieren genau

$$
\prod_{(\alpha,x) \in \pi_1(\mathtt{v}(q,G_Q))} x! \cdot \prod_{(\beta,y) \in \pi_2(\mathtt{v}(q,G_Q))} y! \cdot \prod_{(\gamma,\delta,z) \in \pi_3(\mathtt{v}(q,G_Q))} z!
$$

unterschiedliche Trefferabbildungen $\Psi : Q^q \to D^d$. Diese ergeben sich aus allen Kombinationsmöglichkeiten der Zuordnungen zwischen $\mathtt{N}^{\setminus\mathtt{V}}(q, G_Q, \alpha)$ und $\mathtt{N}^{\setminus\mathtt{V}}(d, G_D, \alpha)$, zwischen $\mathtt{V}^{\setminus\mathtt{N}}(q, G_Q, \beta)$ und $\mathtt{V}^{\setminus\mathtt{N}}(d, G_D, \beta)$ sowie zwischen den Mengen $\mathtt{NV}(q, G_Q, \gamma, \delta)$ und $\mathtt{NV}(d, G_D, \gamma, \delta)$; siehe den Beweis der Rückrichtung von Satz 3.3.29 (1).

**Beispiel 3.3.31.** Für die in Abbildung 3.11 dargestellte Anfrage $G_Q$ und das Dokument $G_D$ ergeben sich die Vokabeln für $q \in Q$ und $d \in D$ als $\mathtt{v}(q, G_Q) = \mathtt{v}(d, G_D) = \big(\big\{(\{1\}, 2), (\{1, 2\}, 1)\big\}, \emptyset, \emptyset\big)$. Für die Konstruktion von Trefferabbildungen $\psi^{q,d}$ ergibt sich nach (3.21) für $t \in \mathtt{N}(q, Q) \setminus \mathtt{V}(q, Q)$,

daß $t$ entweder auf $b$ oder auf $c$ abgebildet werden kann, da $\mathtt{N}^{\setminus\mathtt{v}}(d, G_D, \xi_Q((q,t)) = \{b,c\}$. Entsprechend muß $s$ auf das andere Objekt abgebildet werden. Insgesamt ergeben sich die folgenden beiden Trefferabbildungen:

| $\psi_1^{q,d}:$ | $Q^q$ | $\rightarrow$ | $D^d$ | $\psi_2^{q,d_q}:$ | $Q^q$ | $\rightarrow$ | $D^d$ |
|---|---|---|---|---|---|---|---|
| | $q$ | $\mapsto$ | $d$ | | $q$ | $\mapsto$ | $d$ |
| | $r$ | $\mapsto$ | $a$ | | $r$ | $\mapsto$ | $a$ |
| | $s$ | $\mapsto$ | $b$ | | $s$ | $\mapsto$ | $c$ |
| | $t$ | $\mapsto$ | $c$ | | $t$ | $\mapsto$ | $b$. |

$\circ$

**Beweis Satz 3.3.29 (2)** "$\Rightarrow$": Analog zum Beweis von Satz 3.3.29 (1). Statt von einer Bijektion geht man nun von einer Injektion $\Psi : Q^q \rightarrow D^d$ aus. Somit kann $d$ einen größeren Knotengrad besitzen als $q$. Aus $\xi_Q((q,q')) = \alpha \Rightarrow \xi_D((\Psi(q), \Psi(q'))) = \alpha$ folgen zusammen mit der Injektivität die Aussagen.
"$\Leftarrow$": Ebenfalls analog zu Beweis von (1) verläuft die Konstruktion der Trefferabbildung $\Psi$. Im Gegensatz zu IC-Treffern, wo $\Psi$ eine Bijektion ist, ist die Trefferabbildung für I-Treffer eine injektive Abbildung von $Q^q$ nach $D^d$. $\Psi$ läßt sich auch hier nach (3.21) konstruieren. Da aus $(\alpha, x) \in \pi_1(\mathtt{v}(q, G_Q))$ folgt, daß es ein $(\alpha, x') \in \pi_1(\mathtt{v}(d, G_D))$ mit $x' \geq x$ gibt, ist die Existenz mindestens einer solchen Trefferabbildung garantiert. $\square$

**Bemerkung 3.3.32.** Für ein I-Vorkommen $d \in D$ von $q \in Q$ existieren insgesamt

$$\prod_{\substack{(\alpha,x)\in\pi_1(\mathtt{v}(q,G_Q))\wedge \\ (\alpha,x')\in\pi_1(\mathtt{v}(d,G_D))}} \frac{x'!}{(x'-x)!} \cdot \prod_{\substack{(\beta,y)\in\pi_2(\mathtt{v}(q,G_Q))\wedge \\ (\beta,y')\in\pi_2(\mathtt{v}(d,G_D))}} \frac{y'!}{(y'-y)!} \cdot \prod_{\substack{(\gamma,\delta,z)\in\pi_3(\mathtt{v}(q,G_Q))\wedge \\ (\gamma,\delta,z')\in\pi_3(\mathtt{v}(d,G_D))}} \frac{z'!}{(z'-z)!}$$

Trefferabbildungen.

**Beispiel 3.3.33.** Für das Dokument $D'$ aus Abbildung 3.11 ergibt sich für $d \in D'$ die Vokabel $\mathtt{v}(d, G_{D'}) = \big(\{(\{1\},2),(\{2\},1),(\{1,2\},3)\},\emptyset,\emptyset\big)$. Betrachtet man bzgl. der Vokabel von $q$ in $G_Q$ alle Elemente aus $\pi_1(\mathtt{v}(q, G_Q))$, so ergibt sich $(\{1\},2) \in \pi_1(\mathtt{v}(q, G_Q)) \cap \pi_1(\mathtt{v}(d, G_{D'}))$. Weiterhin gilt für $(\{1,2\},1) \in \pi_1(\mathtt{v}(q, G_Q)) : (\{1,2\},3) \in \pi_1(\mathtt{v}(d, G_{D'}))$. Da $\pi_2(\mathtt{v}(q, G_Q))$ und $\pi_2(\mathtt{v}(q, G_Q))$ leer sind, sind alle Bedingungen aus Satz 3.3.29 erfüllt. Also ist $d$ ein I-Vorkommen von $q$.

Konstruiert man die Trefferabbildungen nach dem im Beweis von Satz 3.3.29 (2) angegebenen

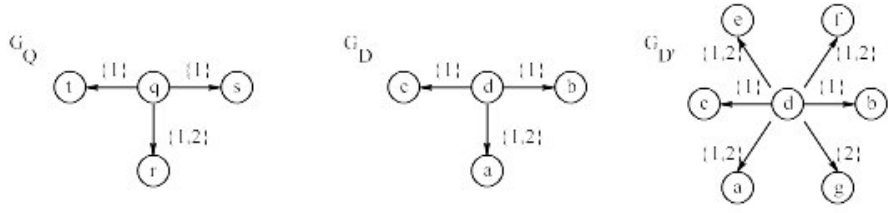



Abbildung 3.11: Eine Anfrage $G_Q$ und zwei Dokumente $G_D$ und $G_{D'}$. $G_Q$ und $G_D$ entsprechen dem Sonderfall aus dem Beweis von Satz 3.3.29 (1). $G_D$ ist IC-Treffer von $G_Q$ mit zwei Trefferabbildungen, siehe Beispiel 3.3.31. $G_{D'}$ ist I-Treffer von $G_Q$ mit 6 Trefferabbildungen, siehe Beispiel 3.3.33.

Vorgehen, so ergeben sich die folgenden 6 injektiven Trefferabbildungen von $Q^q$ nach $D'^d$:

| $p$ | $\psi_1^{q,d}(p)$ | $\psi_2^{q,d}(p)$ | $\psi_3^{q,d}(p)$ | $\psi_4^{q,d}(p)$ | $\psi_5^{q,d}(p)$ | $\psi_6^{q,d}(p)$ |
|---|---|---|---|---|---|---|
| $q$ | $d$ | $d$ | $d$ | $d$ | $d$ | $d$ |
| $r$ | $a$ | $a$ | $e$ | $e$ | $f$ | $f$ |
| $s$ | $b$ | $c$ | $b$ | $c$ | $b$ | $c$ |
| $t$ | $c$ | $b$ | $c$ | $b$ | $c$ | $b$. |

Die Anzahl der Abbildungen ergibt sich aus 3.3.32 mit

$$\frac{2!}{(2-2)!} \cdot \frac{3!}{(3-1)!} = 6.$$

∘

**Bemerkung 3.3.34.** Für IC-Vorkommen und I-Vorkommen gilt: Die zur Konstruktion von $\psi^{q,d}$ (im Beweis mit $\Psi$ bezeichnet) nach (3.21) benötigten Mengen $\mathtt{N}^{\setminus\mathtt{V}}(d, G_D, \alpha)$, $\mathtt{V}^{\setminus\mathtt{N}}(d, G_D, \beta)$ und $\mathtt{NV}(d, G_D, \gamma, \delta)$ sind im Listeneintrag von $E(d, G_D)$ gespeichert. Somit dient (3.21) zur Konstruktion aller Abbildungen $\psi^{q,d}$ aus einem Listeneintrag $E(d, G_D)$.

Nachdem bisher die Berechnung von IC-Vorkommen und I-Vorkommen beschrieben wurde, wird im folgenden auf die Berechnung von S-Vorkommen eingegangen.

Zunächst werden zwei notwendige Bedingungen vorgestellt. Diese können in der Praxis zur Einschränkung des Suchraums verwendet werden. Ziel ist Charakterisierung von S-Vorkommen durch geeignete Eigenschaften der Vokabeln, worauf an späterer Stelle dieses Abschnittes eingegangen wird.

Bei der Berechnung von beschrifteten Graphen aus Dokumenten zusammen mit einer Menge von Relationen $\mathcal{R}$ wurden als Kantenbeschriftungen Mengen von natürlichen Zahlen verwendet. Die geforderte Ordnung auf diesen Mengen ergibt sich durch die Mengeninklusion. Der folgende Satz läßt sich unter der Verwendung der Beschriftung durch Mengen, d.h. $\alpha, \beta, \gamma, \delta \subseteq \{1, \ldots, |\mathcal{R}|\}$, anschaulich formulieren.

**Satz 3.3.35.** *Sei $d \in D$ und $q \in Q$ mit den Vokabeln $v := \mathtt{v}(d, G_D)$ und $v^q := \mathtt{v}(q, G_Q)$. Ist $d$ ein S-Vorkommen von $q$, so gilt:*

$$\bigcup_{(\alpha,x)\in\pi_1(v^q)} \alpha \cup \bigcup_{(\gamma,\delta,z)\in\pi_3(v^q)} \gamma \subseteq \bigcup_{(\alpha,x)\in\pi_1(v)} \alpha \cup \bigcup_{(\gamma,\delta,z)\in\pi_3(v)} \gamma$$

*und*

$$\bigcup_{(\beta,y)\in\pi_2(v^q)} \beta \cup \bigcup_{(\gamma,\delta,z)\in\pi_3(v^q)} \delta \subseteq \bigcup_{(\beta,y)\in\pi_2(v)} \beta \cup \bigcup_{(\gamma,\delta,z)\in\pi_3(v)} \delta.$$

**Beweis** Sei $\Psi$ eine S-Trefferabbildung mit $\Psi(q) = d$. Dann gilt

$$\begin{aligned}
\bigcup_{(\alpha,x)\in\pi_1(v^q)} \alpha \cup \bigcup_{(\gamma,\delta,z)\in\pi_3(v^q)} \gamma &= \bigcup_{q'\in\mathtt{N}(q,G_Q)} \xi_Q((q,q')) \\
&\subseteq \bigcup_{q'\in\mathtt{N}(q,G_Q)} \xi_D((\Psi(q),\Psi(q'))) \\
&\subseteq \bigcup_{(\alpha,x)\in\pi_1(v)} \alpha \cup \bigcup_{(\gamma,\delta,z)\in\pi_3(v)} \gamma
\end{aligned}$$

und

$$\begin{aligned}
\bigcup_{(\beta,y)\in\pi_2(v^q)} \beta \cup \bigcup_{(\gamma,\delta,z)\in\pi_3(v^q)} \delta &= \bigcup_{q'\in\mathtt{V}(q,G_Q)} \xi_Q((q',q)) \\
&\subseteq \bigcup_{q'\in\mathtt{V}(q,G_Q)} \xi_D((\Psi(q'),\Psi(q))) \\
&\subseteq \bigcup_{(\beta,y)\in\pi_2(v)} \beta \cup \bigcup_{(\gamma,\delta,z)\in\pi_3(v)} \delta.
\end{aligned}$$

□

Während der letzte Satz eine sehr grobe Charakteristik von Vokabeln beschreibt, liefert der folgende Satz, der sich auf beliebige Kantenbeschriftungen bezieht, eine Eigenschaft von S-Vorkommen, die mit den Aussagen über IC-Vorkommen und I-Vorkommen aus Satz 3.3.29 vergleichbar ist.

**Satz 3.3.36.** *Sei $d \in D$ und $q \in Q$ mit den Vokabeln $v := \mathtt{v}(d, G_D)$ und $v^q := \mathtt{v}(q, G_Q)$. Ist $d$ ein S-Vorkommen von $q$ so gilt:*
1. $\forall(\alpha, x) \in \pi_1(v^q) : x \leq$

$$\sum_{(\alpha',x')\in\pi_1(v),\alpha'\geq\alpha} x' + \sum_{(\alpha',\delta',z')\in\pi_3(v),\alpha'\geq\alpha} z' - \sum_{(\alpha',x')\in\pi_1(v^q),\alpha'>\alpha} x' - \sum_{(\alpha',\delta',z')\in\pi_3(v^q),\alpha'\geq\alpha} z'$$

2. $\forall(\beta, y) \in \pi_2(v^q) : y \leq$

$$\sum_{(\beta',y')\in\pi_2(v),\beta'\geq\beta} y' + \sum_{(\gamma',\beta',z')\in\pi_3(v),\beta'\geq\beta} z' - \sum_{(\beta',y')\in\pi_2(v^q),\beta'>\beta} y' - \sum_{(\gamma',\beta',z')\in\pi_3(v^q),\beta'\geq\beta} z'$$

3. $\forall(\gamma, \delta, z) \in \pi_3(v^q) : z \leq$

$$\sum_{(\gamma',\delta',z')\in\pi_3(v),\gamma'\geq\gamma\wedge\delta'\geq\delta} z' - \sum_{(\gamma',\delta',z')\in\pi_3(v^q),\gamma'\geq\gamma\wedge\delta'\geq\delta\wedge(\gamma'\neq\gamma\vee\delta'\neq\delta)} z'.$$

**Beweis** Beginnen wir mit dem 3. Fall, wo Knoten $p$ aus $Q^q$ betrachtet werden, die sowohl Vorgänger als auch Nachfolger von $q$ sind. Für ein Paar $(\gamma, \delta)$, für das es mindestens ein $p \in \mathtt{N}(q, G_Q) \cap \mathtt{V}(q, G_Q)$ mit $\xi_Q((p, q)) = \gamma$ und $\xi_Q((q, p)) = \delta$ gibt, gilt:

$$\begin{aligned} &|\{p' \in \mathtt{N}(q, G_Q) \cap \mathtt{V}(q, G_Q) \quad | \quad \xi_Q((q, p')) \geq \gamma \wedge \xi_Q((p', q)) \geq \delta\}| \\ \leq\ &|\{c' \in \mathtt{N}(d, G_D) \cap \mathtt{V}(d, G_D) \quad | \quad \xi_D((d, c')) \geq \gamma \wedge \xi_D((c', d)) \geq \delta\}|. \end{aligned}$$

Für die Vokabeln gilt dann nach Konstruktion

$$\sum_{\substack{(\gamma',\delta',z')\in\pi_3(\mathtt{v}(q,G_Q)),\\ \gamma'\geq\gamma\ \wedge\ \delta'\geq\delta}} z' \ \leq \sum_{\substack{(\gamma',\delta',z')\in\pi_3(\mathtt{v}(d,G_D)),\\ \gamma'\geq\gamma\ \wedge\ \delta'\geq\delta}} z'.$$

Hieraus folgt die Behauptung für den 3. Punkt. Der Beweis der ersten beiden Punkte erfolgt analog. □

Die Rückrichtung des letzten Satzes gilt nicht. Ein Beispiel dafür ist in Abbildung 3.12 angegeben.

Die letzten beiden Sätze liefern keine Äquivalenz zwischen S-Vorkommen und Eigenschaften von Vokabeln. Sie geben eher Hinweise, welche Vokabeln innerhalb des gesamten Vokabulars zur Bestimmung von S-Vorkommen in Frage kommen.

Eigenschaften zwischen Vokabeln, die eine Äquivalenz zu S-Vorkommen herstellen, ergeben sich aus dem Heiratssatz von Hall für bipartite Graphen. Hierfür wird für $q \in Q$ und $d \in D$ aus den beschrifteten Graphen $G_Q^q$ und $G_D^d$ (siehe Definition 3.3.2) ein bipartiter Graph konstruiert. Der Heiratssatz liefert ein notwendiges und zugleich hinreichendes Kriterium für die Existenz eines vollständigen Matchings in einem bipartiten Graphen. Wie sich zeigen läßt, lassen sich aus vollständigen Matchings alle lokalen Trefferabbildungen $\psi^{q,d} : Q^q \to D^d$ für S-Vorkommen $d \in D$ von $q$ berechnen.

Doch beginnen wir mit den Eigenschaften von Vokabeln, die eine Äquivalenz zu S-Vorkommen ergeben:

**Satz 3.3.37.** *Sei $q$ aus $Q$ und $d$ aus $D$ mit den Vokabeln $v := \mathtt{v}(d, G_D)$ und $v^q := \mathtt{v}(q, G_Q)$. Genau dann ist $d \in D$ ein S-Vorkommen von $q \in Q$, wenn $|Q^q| \leq \left|D^d\right|$ und $\forall S_1 \subseteq \pi_1(v^q), S_2 \subseteq$*

$\pi_2(v^q), S_3 \subseteq \pi_3(v^q)$ :

$$\sum_{(\alpha,x)\in S_1} x + \sum_{(\beta,y)\in S_2} y + \sum_{(\gamma,\delta,z)\in S_3} z \leq \sum_{\substack{(\alpha',x')\in\pi_1(v):\\ \exists(\alpha,x)\in S_1:\alpha'\geq\alpha}} x' + \sum_{\substack{(\beta',y')\in\pi_2(v):\\ \exists(\beta,y)\in S_2:\beta'\geq\beta}} y' + \sum_{\substack{(\gamma',\delta',z')\in\pi_3(v):\\ \big(\exists(\alpha,x)\in S_1:\gamma'\geq\alpha\big)\vee\\ \big(\exists(\beta,y)\in S_2:\gamma'\geq\beta\big)\vee\\ \big(\exists(\gamma,\delta,z)\in S_3:\gamma'\geq\gamma\wedge\delta'\geq\delta\big)}} z'.$$

Zunächst kommentieren wir die Aussage des Satzes, geben illustrierende Beispiele und bereiten den Beweis des Satzes vor.

In den Summen spiegeln sich die zahlreichen Kombinationsmöglichkeiten wider, die sich bei S-Treffern ergeben. Diese werden durch das folgende Beispiel motiviert.

**Beispiel 3.3.38.** In Abbildung 3.13 sind eine Anfrage $G_Q$ und zwei einfache Dokumente in Form von beschrifteten Graphen dargestellt. Für $q \in Q$, $d \in D$ und $d \in D'$ seien die Vokabeln $v^q := \mathtt{v}(q, G_Q)$, $v := \mathtt{v}(d, G_D)$ und $v' := \mathtt{v}(d, G_{D'})$. Betrachten wir $G_Q$ erstmal nur bezüglich $G_D$. Die Knoten der Dokumente $G_Q = G_Q^q$ und $G_D = G_D^d$ bestehen nur aus Nachfolgern von $q$ bzw. $d$. Somit ist nur der erste Tupeleintrag beider Vokabeln nichtleer. Es ergibt sich $\pi_1(v^q) = \big\{(\{1\},2),(\{2\},1),(\{1,2\},3)\big\}$ und $\pi_1(v) = \big\{(\{1\},3),(\{3\},1),(\{1,2\},2),(\{1,2,3\},1)\big\}$. Nun soll untersucht werden, ob $d$ ein S-Vorkommen von $q$ ist. Es wird versucht, eine Trefferabbildung $\Psi : Q^q \to D^d$ zu konstruieren. Hier muß für jeden Nachfolger $q'$ von $q$ mindestens ein "passender" Nachfolger $d'$ von $d$ mit $\xi_Q((q,q')) \leq \xi_D((d,d'))$ vorhanden sein, um somit $\Psi(q')$ als $d'$ definieren zu können. Auf die Vokabel bezogen bedeutet dies, daß bzgl. $(\{1\},2) \in \pi_1(v^q)$ mindestens zwei Nachfolger $d'$ von $d$ mit $\xi_D((d')) \geq \{1\}$ existieren müssen. Insgesamt existieren 6 solcher $d'$, was sich aus den Vokabeln $(\{1\},3),(\{1,2\},2)$ und $(\{1,2,3\},1)$ aus $\pi_1(v)$ ergibt $(6 = 3+2+1)$. Betrachtet man analog die anderen beiden Tupel aus $\pi_1(v^q)$, so sind auch hier genügend Nachfolger von $d$ vorhanden. Dies gilt allerdings nur, wenn man beide getrennt voneinander betrachtet. Betrachtet man $(\{2\},1)$ und $(\{1,2\},3)$ zusammen, so stellt sich heraus, daß nur 3 passende Nachfolger von $d$ existieren. Für $Q$ gilt mit $S_1 = \{(\{2\},1),(\{1,2\},3)\}$ und $\pi_1(S_1) = \{\{2\},\{1,2\}\}$:

$$|\{q' \in \mathtt{N}(q,G_Q) \setminus \mathtt{V}(q,G_Q) | \xi_Q((q,q')) \in \pi_1(S_1)\}| = \sum_{(\alpha,x)\in S_1} x \ = 4. \tag{3.22}$$

Bzgl. $D$ ergibt sich die Menge der möglichen Bilder $\Psi(q')$ für Objekte $q'$ aus der in (3.22) angegebenen Objektmenge als

$$\{d' \in \mathtt{N}(d,G_D) \setminus \mathtt{V}(d,G_D) | \exists \alpha \in \pi_1(S_1) : \xi_D((d,d')) \geq \alpha\}. \tag{3.23}$$

Die Kardinalität dieser Menge ergibt sich als

$$\sum_{\substack{(\alpha',x')\in\pi_1(v):\\ \exists(\alpha,x)\in S_1:\alpha'\geq\alpha}} x' \ = 2+1. \tag{3.24}$$

Da diese echt kleiner als die Kardinalität aus (3.22) ist, kann keine injektive Trefferabbildung $\Psi$ existieren. Somit ist $d$ kein S-Vorkommen von $q$.

Untersucht man für $D$ alle Teilmengen $S_1 \subseteq \pi_1(v^q)$, so ist nur für $S_1 = \{(\{2\},1),(\{1,2\},3)\}$ die Ungleichung von Satz 3.3.37 nicht erfüllt. Auch für $S_1 = \pi_1(v^q)$, wenn also alle Knoten bzw. Kanten von $Q$ betrachtet werden, stimmt die Kardinalität der unter (3.22) und (3.23) definierten Objektmengen überein. Dies gewährleistet zwar die Existenz einer injektiven Abbildung $\Psi : Q \to D$, allerdings erfüllt diese im vorliegenden Fall nicht die restlichen an eine Trefferabbildung gestellten Anforderungen.

Betrachten wir nun $G_{D'}$ bzgl. der Anfrage $G_Q$. $G_{D'}$ stellt eine Erweiterung von $G_D$ um einen Nachfolger von $d$ dar, der auch Vorgänger von $d$ ist, d.h. $\pi_3(v') = \{(\{2\},\{1\},1)\}$. Für $t \in Q$ kann dieser zusätzliche Knoten $i \in D'$ als Bild $\Psi(t)$ der Trefferabbildung verwendet werden. Hier wird der "nur" Nachfolger $t$, $t \in \mathtt{N}(q,G_Q) \setminus \mathtt{V}(q,G_Q)$ von $q$ auf einen Nachfolger $i$ von $d$, der


Abbildung 3.12: Für die dargestellten Graphen ergeben sich die Vokabeln als $\mathtt{v}(q, G_Q) = \big(\big\{(\{3\},1),(\{1,2\},1)\big\},\emptyset,\emptyset\big)$ und $\mathtt{v}(d, G_D) = \big(\big\{(\{1,2,3\},1)\big\},\emptyset,\emptyset\big)$. Die drei Punkte der Implikation aus Satz 3.3.36 sind erfüllt, wobei sich sowohl für $(\{3\},1)$ also auch für $(\{1,2\},1)$ aus $\pi_1(\mathtt{v}(q, G_Q))$ $1 \leq 1 + 0 - 0 - 0$ ergibt. Trotzdem ist $d$ kein S-Vorkommen von $q$.

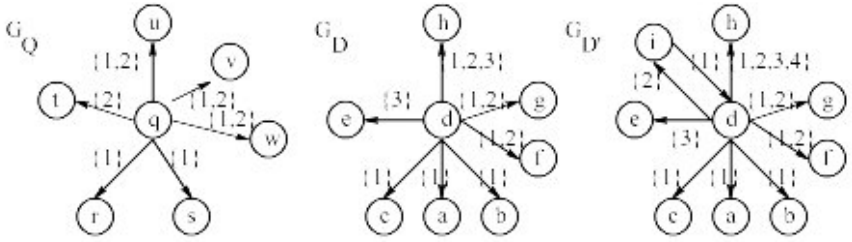



Abbildung 3.13: Drei Dokumente $G_Q$, $G_D$ und $G_{D'}$. Als Vokabeln ergeben sich für $q \in Q$: $\mathtt{v}(q, G_Q) = \big(\big\{(\{1\},2),(\{2\},1),(\{1,2\},3)\big\},\emptyset,\emptyset\big)$ und für $d \in D$: $\mathtt{v}(d, G_D) = \big(\big\{(\{1\},3),(\{3\},1),(\{1,2\},2),(\{1,2,3\},1)\big\},\emptyset,\emptyset\big)$. Da $G_{D'}$ eine Erweiterung von $D$ um das Objekt $i$ ist, ändert sich für die Vokabel von $d \in D'$ im Vergleich zu $d \in D$ nur der dritte Eintrag: $\pi_3(\mathtt{v}(d, G_{D'})) = \big\{(\{2\},\{1\},1)\big\}$. Im Gegensatz zu $d \in D$ ist $d \in D'$ ein S-Vorkommen von $q \in Q$.

auch Vorgänger ist, abgebildet, d.h. $i \in \mathtt{N}(d, G_{D'}) \cap \mathtt{V}(q, G_{D'})$. Für $G_{D'}$ wird die in (3.23) bzgl. $S_1 = \{(\{2\}, 1), (\{1, 2\}, 3)\}$ definierte Menge vereinigt mit

$$\{d' \in \mathtt{N}(d, G_D) \cap \mathtt{V}(d, G_D) | \exists \alpha \in \pi_1(S_1) : \xi_D((d, d')) \geq \alpha\}. \tag{3.25}$$

Die Kardinalität der vereinigten Menge ergibt sich bzgl. $v'$ durch

$$\sum_{\substack{(\alpha', x') \in \pi_1(v): \\ \exists (\alpha, x) \in S_1 : \alpha' \geq \alpha}} x' + \sum_{\substack{(\gamma', \delta', z') \in \pi_3(v): \\ \left(\exists (\alpha, x) \in S_1 : \gamma' \geq \alpha\right)}} z'.$$

Hierbei ergibt sich die erste Summe aus (3.24) und die zweite Summe als die Kardinalität der in (3.25) definierten Objektmenge. Insgesamt ergibt sich bzgl. $S_1$ für beide Summen: $(2+1)+(1) = 4$. $S_1$ erfüllt also die Ungleichung aus Satz 3.3.37. Da für alle anderen Teilmengen $S_1$ von $\pi_1(v^q)$ sich die Summanden im Vergleich zu $D$ nicht ändern und diese die Ungleichung von Satz 3.3.37 erfüllen, ist $d$ S-Vorkommen von $q$. Eine Trefferabbildung $\psi^{q,d} : Q^q \to D^d$ ist z.B. gegeben durch $q \mapsto d, r \mapsto a, s \mapsto b, t \mapsto i, u \mapsto f, v \mapsto g$ und $w \mapsto h$. Auf eine konkrete Berechnung aller Trefferabbildungen wird im Verlauf des Beweises von Satz 3.3.37 eingegangen. ∘

Das letzte Beispiel zeigt, daß schon bei einfachen Dokumenten viele Kombinationsmöglichkeiten zu berücksichtigen sind. Betrachtet man die letzte Summe aus der Ungleichung von Satz 3.3.37, so spiegeln die drei Disjunktionen bei der Bildung der Summe die drei Fälle wider, daß "nur" Nachfolger (bzgl. $S_1$), "nur" Vorgänger (bzgl. $S_2$) oder kombinierte Nachfolger/Vorgänger (bzgl. $S_3$) von $q$ auf kombinierte Nachfolger/Vorgänger von $d$ mit "passenden" Kantenbeschriftungen abgebildet werden können. In obigen Beispiel wurde im Zusammenhang mit $D'$ nur der Fall "nur" Nachfolger von $q$ auf kombinierte Nachfolger/Vorgänger von $d$ besprochen, wodurch die Summe nur bzgl. $S_1$ gebildet wurde.

Zum Beweis von Satz 3.3.37 benötigen wir den Heiratssatz von P. Hall, der durch folgende Definition vorbereitet wird.

**Definition 3.3.39.** Sei $G = (V, E)$ ein ungerichteter Graph. Ein **Matching** $M$ von $G$ ist eine Teilmenge von $E$ mit:

1. $M$ enthält keine Schlingen und
2. keine zwei Kanten aus $M$ sind inzident, d.h. $\forall k, l \in M, k \neq l : k \cap l = \emptyset$.

Sei $G = (V, E)$ ein Graph. Analog zur Definition der Umgebung $\mathtt{U}(v, G)$ eines Knotens $v \in V$ (Defintion 3.3.1) wird für eine Teilmenge $S \subseteq V$ die Umgebung $\mathtt{U}(S, G)$ als die Menge aller Nachbarknoten von Knoten aus $S$ ohne $S$ definiert, d.h. $\mathtt{U}(S, G) := \cup_{v \in S} \mathtt{U}(v, G) \setminus S$.

Bekanntlich heißt ein Graph $G = (V, E)$ **bipartit**, wenn sich $V$ disjunkt aufspalten läßt in $V = X \sqcup Y$, so daß jede Kante aus $E$ jeweils ein Element von $X$ mit einem Element aus $Y$ verbindet. Statt $G = (V, E)$ schreiben wir dann $G = (X, Y, E)$. Ein Matching $M$ von $G = (X, Y, E)$ heißt **vollständig**, wenn $|M| = min\{|X|, |Y|\}$ gilt.

**Satz 3.3.40 (Heiratssatz).** *Sei $G = (X, Y, E)$ ein bipartiter Graph mit $|X| \leq |Y|$. $G$ besitzt ein vollständiges Matching $M$ genau dann, wenn $|\mathtt{U}(S, G)| \geq |S|$ für alle $S \subseteq X$ gilt.*

Für den Beweis des Satzes sei auf [1] verwiesen.

In der folgenden Definition wird bzgl. $q \in Q$ und $d \in D$ der bipartite Graphen definiert, auf den der Heiratssatz angewendet wird. O.B.d.A. seien $Q$ und $D$ disjunkte Mengen.

**Definition 3.3.41.** Für $q \in Q$ und $d \in D$ definiert $\mathcal{G}(q, G_Q, d, G_D)$ den bipartiten ungerichteten Graphen $G = (Q^q, D^d, E)$ mit

$$\begin{aligned} E = \big\{\{q', d'\} \quad | \quad & \big(q' = q \wedge d' = d\big) \vee \\ & \big(q' \in \mathtt{N}(q, G_Q) \setminus \mathtt{V}(q, G_Q) \wedge d' \in \mathtt{N}(d, G_D) \wedge \xi_Q((q, q')) \leq \xi_D((d, d'))\big) \vee \\ & \big(q' \in \mathtt{V}(q, G_Q) \setminus \mathtt{N}(q, G_Q) \wedge d' \in \mathtt{V}(d, G_D) \wedge \xi_Q((q', q)) \leq \xi_D((d', d))\big) \vee \\ & \big(q' \in \mathtt{N}(q, G_Q) \cap \mathtt{V}(q, G_Q) \wedge d' \in \mathtt{N}(d, G_D) \cap \mathtt{V}(d, G_D) \wedge \\ & \quad \xi_Q((q, q')) \leq \xi_D((d, d')) \wedge \xi_Q((q', q)) \leq \xi_D((d', d))\big)\big\}. \end{aligned}$$

Ein Beispiel ist in Abbildung 3.14 dargestellt. Man sieht deutlich den Bezug zu möglichen lokalen Trefferabbildungen. Existiert eine lokale Trefferabbildung $\psi^{q,d} : Q^q \to D^d$, so gilt $\{p, \psi^{q,d}(p)\} \in E$ für alle $p \in Q^q$, wobei $E$ die Kantenmenge des Graphen bzgl. Definition 3.3.41 ist. Satz 3.3.37 stellt sicher, wann der umgekehrte Fall gilt und wie sich aus dem Graphen bzgl. Definition 3.3.41 eine lokale Trefferabbildung konstruieren läßt. Dies wird im folgenden bewiesen. Man beachte, daß die Graphen $\mathcal{G}(q, G_Q^q, d, G_D^d)$ und $\mathcal{G}(q, G_Q, d, G_D)$ gleich sind. Dies wird in Abbildung 3.14 durch den Knoten $f \in D \setminus D^d$ demonstriert, der keinen Knoten in $\mathcal{G}(q, G_Q, d, G_D)$ darstellt.

**Satz 3.3.42.** *Ein Objekt $d \in D$ ist genau dann ein S-Vorkommen von $q \in Q$, wenn $|Q^q| \leq \left|D^d\right|$ gilt und $\mathcal{G}(q, G_Q, d, G_D)$ ein Matching $M$ mit $|M| = |Q^q|$ besitzt.*

**Beweis** "⇒": Sei $d$ ein S-Vorkommen von $q$ mit lokaler Trefferabbildung $\psi^{q,d} : Q^q \to D^d$. Dann gilt für alle $(a, b) \in E_{Q^q} : (\psi^{q,d}(a), \psi^{q,d}(b)) \in E_{D^d}$ und $\xi_Q((a, b)) \leq \xi_D((\psi^{q,d}(a), \psi^{q,d}(b)))$ (Satz 3.3.5). Nach Konstruktion des bipartiten Graphen $\mathcal{G}(q, G_Q, d, G_D) = (Q^q, D^d, E)$ nach Definition 3.3.41 ist $\{a, \psi^{q,d}(a)\} \in E$ für alle $a \in Q^q$. Da $\psi^{q,d}$ injektiv ist, ist $M := \{\{a, \psi^{q,d}(a)\} | a \in Q^q\}$ Matching mit $|M| = |Q^q|$.
"⇐": Sei $M$ ein Matching mit $|M| = |Q^q|$. Dann existiert für jedes $a \in Q^q$ genau eine Kante $\{a, d_a\} \in M$. Nach Konstruktion ist $d_a \in D^d$ und $\{q, d\} \in M$. Die Abbildung $\psi^{q,d} : Q^q \to D^d$ wird definiert durch $\psi^{q,d}(a) = d_a$. Da bei einem Matching keine zwei Kanten inzident sind, ist $\psi^{q,d}$ injektiv. Nach Konstruktion (Definition 3.3.41) gilt $\forall (a, b) \in E_{Q^q} : (\psi^{q,d}(a), \psi^{q,d}(b)) \in E_{D^d}$ und $\xi_Q((a, b)) \leq \xi_D((\psi^{q,d}(a), \psi^{q,d}(b)))$. Damit ist $\psi^{q,d}$ eine Trefferabbildung und $d$ ein S-Vorkommen von $q$ (Satz 3.3.5). □

Das Beispiel in Abbildung 3.14 veranschaulicht die Aussage des vorherigen Satzes.

Da der Graph $\mathcal{G}(q, G_Q, d, G_D)$ im folgenden immer bzgl. $q \in Q$ und $d \in D$ betrachtet wird, wird dieser einfach nur durch $\mathtt{G}$ bezeichnet.

Der bipartite Graph $\mathtt{G}$ dient nur zur Veranschaulichung der Zusammenhänge zwischen $Q^q$ und $D^d$. Bei den späteren Verfahren muß er nicht berechnet werden.

Für den Beweis von Satz 3.3.37 werden Teilmengen von $D^d$ benötigt, die sich bzgl. einer Teilmenge $Q'$ von $Q$ als die Umgebung von $Q'$ in $\mathtt{G}$ ergeben. Diese lassen sich direkt aus den Daten berechnen, die im von $d \in D$ erzeugten Listeneintrag gespeichert sind. In diesem Zusammenhang ergibt sich $\mathtt{U}(Q', \mathtt{G})$ für eine Teilmenge $Q'$ von $Q$ als

$$\{d' \in \mathtt{N}(d, G_D) | \exists q' \in (\mathtt{N}(q, G_Q) \setminus \mathtt{V}(q, G_Q)) \cap Q' : \xi_Q((q, q')) \leq \xi_D((d, d'))\} \cup \tag{3.26}$$

$$\{d' \in \mathtt{V}(d, G_D) | \exists q' \in (\mathtt{V}(q, G_Q) \setminus \mathtt{N}(q, G_Q)) \cap Q' : \xi_Q((q', q)) \leq \xi_D((d', d))\} \cup \tag{3.27}$$

$$\{d' \in \mathtt{N}(d, G_D) \cap \mathtt{V}(d, G_D) | \exists q' \in \mathtt{N}(q, Q) \cap \mathtt{V}(q, Q) \cap Q' : \xi_Q((q, q')) \leq \xi_D((d, d')) \wedge \xi_Q((q', q)) \leq \xi_D((d', d))\} \tag{3.28}$$

$$\cup \{d | q \in Q'\}. \tag{3.29}$$

Falls $q \in Q'$ gilt, besteht die in (3.29) definierte Menge nur aus dem Objekt $d$. Ansonsten ist die Menge leer. Hierbei sind $d \in D$ und $q \in Q$ die Objekte, bzgl. denen der bipartite Graph $\mathtt{G}$ konstruiert wurde.

Für $Q' = \{r, s, t, u\}$ und das Beispiel aus Abbildung 3.14 ergibt sich $\mathtt{U}(Q', \mathtt{G})$ aus den Mengen $\{a, b, c, e\}$ bzgl. (3.26), $\emptyset$ bzgl. (3.27) und (3.29) sowie $\{c\}$ bzgl. (3.28). Die Mengen sind nicht disjunkt. Insgesamt gilt $\mathtt{U}(Q', \mathtt{G}) = \{a, b, c, e\}$.

**Satz 3.3.43.** *$d \in D$ ist genau dann ein S-Vorkommen von $q \in Q$, wenn $|Q^q| \leq \left|D^d\right|$ und für alle $Q' \subseteq Q^q : |Q'| \leq |\mathtt{U}(Q', \mathtt{G})|$ gilt.*

**Beweis** $d$ ist S-Vorkommen von $q \Leftrightarrow$ (Satz 3.3.42)
$|Q^q| \leq \left|D^d\right|$ und für $\mathcal{G}(q, G_Q, d, G_D)$ existiert ein Matching $M$ mit $|M| = |Q^q| \Leftrightarrow$ (Satz 3.3.40)
$|Q^q| \leq \left|D^d\right|$ und $\forall Q' \subseteq Q^q : |\mathtt{U}(Q', \mathcal{G}(q, G_Q, d, G_D))| \geq |Q'|$ □

Zur Demonstration des letzten Satzes betrachten wir folgendes Beispiel.

**Beispiel 3.3.44.** Auf die drei Dokumente aus Abbildung 3.13 wurde schon in Beispiel 3.3.38 eingegangen. Betrachten wir $Q' = \{t, u, v, w\} \subset Q$. Man beachte, daß $Q'$ nur aus reinen Nachfolgern von $q$ besteht.

Bzgl. $G_D$ gilt: Die unter (3.26) definierte Menge ergibt sich als $\{f, g, h\}$. Da die unter (3.27), (3.28) und (3.29) definierten Mengen aufgrund obiger Beobachtung und wegen $q \notin Q'$ leer sind, ergibt sich $\mathtt{U}(Q', \mathtt{G}) = \{f, g, h\}$. Es gilt $|Q'| > |\mathtt{U}(Q', \mathtt{G})|$. Somit ist $d \in D$ wegen Satz 3.3.43 kein S-Vorkommen von $q$.

Bzgl. $G_{D'}$ gilt: $|Q'| \leq |\mathtt{U}(Q', \mathcal{G}(q, G_Q, d, G_{D'}))|$, da sich die Objektmenge bzgl. (3.26) in $\{f$, $g$, $h$, $i\}$ vergrößert. Da die Ungleichung auch für alle anderen $Q' \subseteq Q$ erfüllt ist, bildet $d \in D'$ ein S-Vorkommen von $q$. ∘

Zur weiteren Vorbereitung des Beweises von Satz 3.3.37 wird bei festem $q \in Q$ eine Äquivalenzrelation $\sim$ auf der Potenzmenge von $Q^q$ eingeführt. Satz 3.3.37 wird bewiesen, indem eine Äquivalenz zu Satz 3.3.43 hergestellt wird. Hierbei spielt die Äquivalenzrelation $\sim$ eine wichtige Rolle. Nach der Einführung der Äquivalenzrelation $\sim$ wird auf Eigenschaften der Äquivalenzklassen eingegangen.

**Definition 3.3.45.** Für $q \in Q$ und $d \in D$ wird die Äquivalenzrelation $\sim_{q,G_Q,d,G_D}$ auf $2^{Q^q}$ definiert durch:

$$X \sim_{q,G_Q,d,G_D} Y :\Leftrightarrow \mathtt{U}(X, \mathtt{G}) = \mathtt{U}(Y, \mathtt{G}).$$

Wenn $q, Q, d$ und $D$ feststehen, wird einfach nur $\sim$ anstatt $\sim_{q,G_Q,d,G_D}$ geschrieben.

Mittels $\sim_{q,G_Q,d,G_D}$ werden Mengen von Nachbarknoten von $q$ in $G_Q$ in Äquivalenzklassen unterteilt. Bezeichne $[X]$ die Äquivalenzklasse, die $X \subseteq Q^q$ als Element enthält. Die Äquivalenzklassen besitzen folgende Eigenschaft:

**Satz 3.3.46.** *Sei $q \in Q$ und $d \in D$. Für $X, Y \subseteq Q^q$ gilt:*

$$X \sim Y \Rightarrow X \sim X \cup Y.$$

**Beweis** Für den Beweis verwenden wir den unter 3.3.41 definierten bipartiten Graphen $\mathtt{G}$.

Für $X \sim Y$ gilt $\mathtt{U}(X, \mathtt{G}) = \mathtt{U}(Y, \mathtt{G})$. Nach Definition 3.3.41 folgt $\mathtt{U}(X, \mathtt{G}) = \mathtt{U}(X \cup Y, \mathtt{G})$. □

Aufgrund des letzten Satzes enthält jede Äquivalenzklasse $[X]$ bzgl. der mengentheoretischen Inklusion ein größtes Element. Dieses eindeutig bestimmte maximale Element wird mit $X_{max}$ bezeichnet. Offensichtlich ist

$$X_{max} = \cup_{Y \in [X]} Y.$$

**Beispiel 3.3.47.** Seien $G_Q$ und $G_D$ die in Abbildung 3.14 dargestellten Graphen. Für $q \in Q$ und $d \in D$ ergeben sich bzgl. $\sim$ die folgenden Äquivalenzklassen $[X]$ für $X \subseteq Q^q \setminus \{q\}$:

| $\mathtt{U}(X, \mathtt{G})$ | $[X]$ |
|---|---|
| $\{a, e\}$ | $\{\ \{u\}, \{s, u\}\ \}$ |
| $\{b, c, e\}$ | $\{\ \{r\}, \{r, s\}, \{r, t\}, \{r, s, t\}\ \}$ |
| $\{c\}$ | $\{\ \{t\}\ \}$ |
| $\{e\}$ | $\{\ \{s\}\ \}$ |
| $\{a, b, c, e\}$ | $\{\ \{r, u\}, \{r, s, u\}, \{r, t, u\}, \{r, s, t, u\}\ \}$ |
| $\{c, e\}$ | $\{\ \{s, t\}\ \}$ |
| $\{a, c, e\}$ | $\{\ \{t, u\}, \{s, t, u\}\ \}$. |

Für jede Äquivalenzklasse $[X]$ ist das maximale Element $X_{max}$ das in einer Tabellenzeile letzte aufgeführte Element der rechten Menge. Z.B. ist für $X = \{r\}$ das maximale Element $X_{max} = \{r, s, t\}$; es ist das letzte Element der Menge in der dritten Zeile auf der rechten Seite der Tabelle.

Für $Y \subseteq Q^q$ mit $q \in Y$ ergibt sich die Äquivalenzklasse $[Y]$ aus dem in der Tabelle angegebenen $[X]$ mit

$$[Y] = \{X \cup \{q\} | X \in [Y \setminus \{q\}]\}.$$

∘

Maximale Elemente besitzen eine weitere Eigenschaft, die auf einer Äquivalenz zwischen Nachbarknoten von $q$ basiert.

**Definition 3.3.48.** Sei $q \in Q$. Zwei Knoten $u, v \in \mathtt{U}(q, G_Q)$ heißen äquivalent bzgl. $q$, $u \simeq_{q,G_Q} v$, wenn

$$\begin{aligned} &\big(u, v \in \mathtt{N}(q, G_Q) \setminus \mathtt{V}(q, G_Q) \quad \wedge \quad \xi_Q((q,u)) = \xi_Q((q,v))\big) \\ \vee\ &\big(u, v \in \mathtt{V}(q, G_Q) \setminus \mathtt{N}(q, G_Q) \quad \wedge \quad \xi_Q((u,q)) = \xi_Q((v,q))\big) \\ \vee\ &\big(u, v \in \mathtt{N}(q, G_Q) \cap \mathtt{V}(q, G_Q) \quad \wedge \quad \xi_Q((q,u)) = \xi_Q((q,v)) \wedge \xi_Q((u,q)) = \xi_Q((v,q))\big) \end{aligned}$$

gilt.

Die so definierte Relation $\simeq_{q,G_Q}$ ist eine Äquivalenzrelation auf $Q^q$. Falls $q$ und $G_Q$ feststehen, wird nur $\simeq$ an Stelle $\simeq_{q,G_Q}$ geschrieben.

**Beispiel 3.3.49.** Für den in Abbildung 3.13 dargestellten Graphen $G_Q$ ergibt sich bzgl. $q \in Q$ u.a. $u \simeq v, v \simeq w$ und $r \simeq s$. ∘

Einen Zusammenhang zwischen den maximalen Elementen und der Äquivalenzrelation $\simeq$ stellt der nächste Satz her. Für $X \subseteq Q^q$ werden zwei äquivalente Nachbarn von $q$ betrachtet, wobei nur einer von beiden Element von $X$ sein muß.

**Satz 3.3.50.** *Sei $q \in Q$, $d \in D$ und $X \subseteq Q^q$. Für $x \in X$ und $p \in Q^q$ gilt:*

$$x \simeq p \ \Rightarrow\ p \in X_{max}.$$

**Beweis** Nach Definition 3.3.41 gilt $\mathtt{U}(X, \mathtt{G}) = \mathtt{U}(X \cup \{p\}, \mathtt{G})$. Also ist $X \sim X \cup \{p\}$. Weiterhin gilt $X_{max} \supseteq X \cup \{p\}$ und somit $p \in X_{max}$. □

Jedes $X_{max}$ ist also eine Vereinigung von $\simeq$-Äquivalenzklassen. Umgekehrt ist jede derartige Vereinigung ein $X_{max}$.

Bzgl. Satz 3.3.43 bedeutet dies, daß nicht mehr alle Teilmengen $Q'$ von $Q$ untersucht werden müssen, sondern nur noch die von der Form $Q' = X_{max}$.

**Satz 3.3.51.** *Sei $d \in D$, $q \in Q$ und $\sim$ die Äquivalenzrelation aus Definition 3.3.45. Genau dann ist $d \in D$ ein S-Vorkommen von $q \in Q$, wenn $|Q^q| \leq \left|D^d\right|$ und für alle maximalen Elemente $Q'_{max}$ mit $[Q'] \in 2^{Q^q}/\sim$ gilt:*

$$|Q'_{max}| \leq |\mathtt{U}(Q'_{max}, \mathtt{G})|\,.$$

**Beweis** Für alle $Q' \subseteq Q^q$ gilt: $|Q'| \leq |Q'_{max}| \leq |\mathtt{U}(Q'_{max}, \mathtt{G})| = |\mathtt{U}(Q', \mathtt{G})|$, womit die Behauptung direkt aus Satz 3.3.43 folgt. □

Kommen wir nun zum Beweis von Satz 3.3.37. Die dort formulierte Aussage über S-Vorkommen wird der im vorherigen Satz getroffenen gegenübergestellt. Aus dem Beweis der Äquivalenz dieser beiden Aussagen ergibt sich direkt Satz 3.3.37. Die Äquivalenz formuliert das folgende Lemma.

**Lemma 3.3.52.** *Seien $q \in Q$ und $d \in D$ mit $|Q^q| \leq \left|D^d\right|$, $v := \mathtt{v}(d, G_D)$ und $v^q := \mathtt{v}(q, G_Q)$ die Vokabeln von $d$ und $q$ und $\sim$ die Äquivalenzrelation auf $2^{Q^q}$ aus Definition 3.3.45. Dann gilt für alle $Q'_{max}$:*

$$|Q'_{max}| \leq |\mathtt{U}(Q'_{max}, \mathtt{G})|$$

*genau dann, wenn $\forall S_1 \subseteq \pi_1(v^q), S_2 \subseteq \pi_2(v^q), S_3 \subseteq \pi_3(v^q)$ :*

$$\sum_{(\alpha,x)\in S_1} x + \sum_{(\beta,y)\in S_2} y + \sum_{(\gamma,\delta,z)\in S_3} z \leq \sum_{\substack{(\alpha',x')\in\pi_1(v):\\ \exists(\alpha,x)\in S_1:\alpha'\geq\alpha}} x' + \sum_{\substack{(\beta',y')\in\pi_2(v):\\ \exists(\beta,y)\in S_2:\beta'\geq\beta}} y' + \sum_{\substack{(\gamma',\delta',z')\in\pi_3(v):\\ \big(\exists(\alpha,x)\in S_1:\gamma'\geq\alpha\big)\vee\\ \big(\exists(\beta,y)\in S_2:\gamma'\geq\beta\big)\vee\\ \big(\exists(\gamma,\delta,z)\in S_3:\gamma'\geq\gamma\wedge\delta'\geq\delta\big)}} z'.$$

.

**Beweis** "⇒": Für ein beliebiges Tripel $(S_1, S_2, S_3)$ mit $S_i \subseteq \pi_i(v^q)$ sei die Teilmenge $Q'_{S_1,S_2,S_3}$ von $Q^q \setminus \{q\}$ definiert als

$$\begin{aligned} Q'_{S_1,S_2,S_3} := \quad & \{p \in \mathtt{N}^{\setminus \mathtt{V}}(q, G_Q, \alpha) | (\alpha, x) \in S_1\} \\ \cup \quad & \{p \in \mathtt{V}^{\setminus \mathtt{N}}(q, G_Q, \beta) | (\beta, y) \in S_2\} \\ \cup \quad & \{p \in \mathtt{NV}(q, G_Q, \gamma, \delta) | (\gamma, \delta, z) \in S_3\}. \end{aligned}$$

Sei $Q'_{max}$ das maximale Element in $[Q'_{S_1,S_2,S_3}]_\sim$. Aus der Annahme folgt

$$\left|Q'_{S_1,S_2,S_3}\right| \leq |Q'_{max}| \leq |\mathtt{U}(Q'_{max}, \mathtt{G})|\,. \tag{3.30}$$

Nun wird die Kardinalität von $Q'_{max}$ und $\mathtt{U}(Q'_{max}, \mathtt{G})$ in Abhängigkeit von $(S_1, S_2, S_3)$ berechnet. Wegen Satz 3.3.50 gilt für $Q'_{max}$ :

$$|Q'_{max}| = \delta_{q \in Q'_{max}} + \sum_{(\alpha,x)\in S_1} x + \sum_{(\beta,y)\in S_2} y + \sum_{(\gamma,\delta,z)\in S_3} z.$$

Hierbei ist $\delta_{q \in Q'_{max}}$ das Kronecker-Symbol, das in diesem Zusammenhang definiert ist als

$$\delta_{q \in Q'_{max}} = \begin{cases} 1 & \text{falls } q \in Q'_{max} \\ 0 & \text{sonst.} \end{cases}$$

Für $\mathtt{U}(Q'_{max}, \mathtt{G})$ ergibt sich die Kardinalität der unter (3.26) definierten Menge als

$$\sum_{\substack{(\alpha',x')\in\pi_1(v):\\ \exists(\alpha,x)\in S_1:\alpha'\geq\alpha}} x' + \sum_{\substack{(\gamma',\delta',z')\in\pi_3(v):\\ \left(\exists(\alpha,x)\in S_1:\gamma'\geq\alpha\right)}} z',$$

da hier $d' \in \mathtt{N}(d, G_D)$ entweder aus $\mathtt{N}(d, G_D) \setminus \mathtt{V}(d, G_D)$ (erste Summe) oder aus $\mathtt{N}(d, G_D) \cap \mathtt{V}(d, G_D)$ (zweite Summe) stammt. Analog gilt für die Kardinalität der in Gleichung (3.27) definierten Menge:

$$\sum_{\substack{(\beta',y')\in\pi_2(v):\\ \exists(\beta,y)\in S_2:\beta'\geq\beta}} y' + \sum_{\substack{(\gamma',\delta',z')\in\pi_3(v):\\ \left(\exists(\beta,y)\in S_2:\delta'\geq\beta\right)}} z'.$$

Schließlich ergibt sich für die Kardinalität der in Gleichung (3.28) definierten Menge:

$$\sum_{\substack{(\gamma',\delta',z')\in\pi_3(v):\\ \left(\exists(\gamma,\delta,z)\in S_3:\gamma'\geq\gamma\wedge\delta'\geq\delta\right)}} z'.$$

Insgesamt ergibt sich $|\mathtt{U}(Q'_{max}, \mathtt{G})|$ unter Berücksichtigung von (3.29) als

$$\delta_{q \in Q'_{max}} + \sum_{\substack{(\alpha',x')\in\pi_1(v):\\ \exists(\alpha,x)\in S_1:\alpha'\geq\alpha}} x' + \sum_{\substack{(\beta',y')\in\pi_2(v):\\ \exists(\beta,y)\in S_2:\beta'\geq\beta}} y' + \sum_{\substack{(\gamma',\delta',z')\in\pi_3(v):\\ \left(\exists(\alpha,x)\in S_1:\gamma'\geq\alpha\right)\vee\\ \left(\exists(\beta,y)\in S_2:\delta'\geq\beta\right)\vee\\ \left(\exists(\gamma,\delta,z)\in S_3:\gamma'\geq\gamma\wedge\delta'\geq\delta\right)}} z'.$$

Mit den Berechnungen der Kardinalitäten folgt die Implikation aus (3.30).

"⇐": Sei $Q'_{max}$ ein maximales Element. Dann werden in Abhängigkeit von $Q'_{max}$ die drei Mengen $S_1 \subseteq \pi_1(v^q), S_2 \subseteq \pi_2(v^q)$ und $S_3 \subseteq \pi_3(v^q)$ wie folgt definiert:

$$\begin{aligned} S_1 \quad := \quad & \left\{(\alpha, x) \in \pi_1(v^q) | \exists q' \in Q'_{max} \cap \left(\mathtt{N}(q, G_Q) \setminus \mathtt{V}(q, G_Q)\right) : \right. \\ & \left. \xi_Q((q, q')) = \alpha\right\}, \\ S_2 \quad := \quad & \left\{(\beta, y) \in \pi_2(v^q) | \exists q' \in Q'_{max} \cap \left(\mathtt{V}(q, G_Q) \setminus \mathtt{N}(q, G_Q)\right) : \right. \\ & \left. \xi_Q((q', q)) = \beta\right\}, \\ S_3 \quad := \quad & \left\{(\gamma, \delta, z) \in \pi_3(v^q) | \exists q' \in Q'_{max} \cap \left(\mathtt{N}(q, G_Q) \cap \mathtt{V}(q, G_Q)\right) : \right. \\ & \left. \xi_Q((q, q')) = \gamma \wedge \xi_Q((q', q)) = \delta\right\}. \end{aligned}$$

Analog zum Beweis der Implikation wird die Kardinalität von $Q'_{max}$ und $\mathtt{U}(Q'_{max}, \mathtt{G})$ bestimmt, woraus sich die Behauptung ergibt. □

**Beweis Satz 3.3.37** Die Aussage folgt sofort aus Lemma 3.3.52 und Satz 3.3.51. □

Die Anzahl der verschiedenen Trefferabbildungen $\psi^{q,d} : Q^q \to D^d$ ist gleich der Anzahl der verschiedenen Matchings aus Satz 3.3.42.

### 3.3.7 Zusammenfassung und Fazit

Ausgangspunkt bei der Trefferberechnung ist die lokale Betrachtung jedes einzelnen Knotens eines Graphen $G_D$. Hierbei wird für einen Knoten $d \in D$ die Umgebung $\mathtt{U}(d, G_D)$ und die Kantenbeschriftungen der Kanten zwischen $d$ und $c \in \mathtt{U}(d, G_D)$ betrachtet. Dies führt zu dem in Abschnitt 3.3.2 eingeführten Begriff "Vorkommen" $d \in D$ eines Knotens $q$ einer Anfrage $Q$. Hier müssen die Umgebung und die Kantenbeschriftungen lokal auf $q$ bezogen mit der Umgebung und den Kantenbeschriftungen lokal auf $d$ bezogen "passen". Dies wird durch Definition 3.3.4 für die einzelnen Trefferarten definiert. In Abschnitt 3.3.3 wurde gezeigt, wie sich einzelne Vorkommen zu einem Treffer kombinieren lassen (Satz 3.3.8). Hierdurch lassen sich alle Treffer bzgl. einer Anfrage $Q$ und einem Dokument $D$ berechnen.

Das oben beschriebene Vorgehen zur Berechnung aller Treffer bezog sich auf ein Dokument. Die Trefferberechnung läßt sich simultan in allen Dokumenten einer Datenbasis realisieren. Hierfür werden invertierte Listen verwendet, die für diese Anwendung in Abschnitt 3.3.5 beschrieben wurden. Die lokalen Informationen über die Nachbarschaft eines Knotens $d \in D$ werden auf zwei Arten gespeichert: In der Vokabel $\mathtt{v}(d, G_D)$ (Definition 3.3.12) werden Kantenbeschriftungen und die jeweilige Anzahl an Kantenbeschriftungen von ein- und ausgehenden Kanten bzgl. $d$ festgehalten. In dem Listeneintrag $E(d, G_D)$ (Definition 3.3.23) werden Referenzen auf die jeweiligen Nachbarknoten $p \in \mathtt{U}(d, G_D)$ gespeichert. Kombiniert man die in der Vokabel $\mathtt{v}(d, G_D)$ und in dem Listeneintrag $E(d, G_D)$ gespeicherten Daten, so läßt sich der auf die Umgebung von $d$ beschränkte Teilgraph $G_{D^d}$ von $G_D$ rekonstruieren.

Jedes Objekt $d$ eines Dokuments $D$ der Datenbasis $\mathcal{D}$ erzeugt einen Listeneintrag. Für eine Vokabel $v$ enthält die invertierte Liste $L(v)$ die Listeneinträge der Objekte $d \in D, D \in \mathcal{D}$, mit $\mathtt{v}(d, G_D) = v$.

Schließlich wurde in Abschnitt 3.3.6 gezeigt, wie man mit Hilfe der gespeicherten Daten alle Treffer einer Anfrage $Q$ berechnet. In Abhängigkeit einer Trefferart $T$ werden für jedes Anfrageobjekt $q$ mit der Vokabel $\mathtt{v}(q, G_Q)$ die invertierten Listen bestimmt, aus deren Listeneinträge sich Vorkommen von $q$ konstruieren lassen. Wie schon beschrieben, lassen sich mit diesen Informationen alle Treffer berechnen.

Für IC-Treffer wird für die Berechnung aller Vorkommen eines Objektes $q$ der Anfrage $Q$ die Einträge genau einer invertierten Liste benötigt. Für die restlichen drei Trefferarten müssen i.d.R. die Einträge aus mehreren invertierten Listen betrachtet werden.

Obwohl sich die beschriebene Indexstruktur für alle vier Trefferarten anwenden läßt, wurde das Vokabular und die invertierten Listen primär im Hinblick auf die Trefferarten I und S definiert, da diese i.d.R. am häufigsten angefragt werden. Eine besondere Eingenschaft der anderen beiden Trefferarten, die Übereinstimmung der Kardinalität zwischen Trefferdokument und Anfrage, wurde bei der Indexkonstuktion vernachlässigt. In Abschnitt 4.2.5 wird beschrieben, wie sich die Aufteilung in invertierte Listen für die Trefferaten IC und SC optimieren läßt.

Auch wenn die relevanten Beweise konstruktiv geführt wurden, lassen sich bestimmte Eigenschaften der invertierten Listen für praktische Anwendungen weiter ausnutzen. Hierauf wird im nächsten Abschnitt eingegangen.

## 3.4 Verfahren zur Trefferberechnung

Im vorangegangen Abschnitt wurde beschrieben, wie sich Treffer aus einzelnen Vorkommen zusammensetzen. Weiterhin wurde auf eine Indexstruktur eingegangen, mit deren Hilfe sich Vorkommen direkt berechnen lassen. Darauf aufbauend werden in diesem Abschnitt vier Verfahren zur Trefferberechnung vorgestellt. Die ersten beiden Verfahren basieren auf lokalen Trefferabbildungen $\psi^{q,d} : Q^q \to D^d$. Sie werden in Abschnitt 3.4.1 vorgestellt. Im darauffolgenden Abschnitt werden zwei Verfahren besprochen, in denen sämtliche lokalen Trefferabbildungen zu festem $q \in Q$ und $d \in D$ durch eine Abbildung $\phi$ kompakt beschrieben werden. Abschließend wird in einem ersten Fazit auf die Vor- und Nachteile der Verfahren eingegangen.

In Abschnitt 2.3 wurden vier verschiedene Trefferarten vorgestellt. Zum Anfragezeitpunkt entscheidet sich der Benutzer für eine der vier Trefferarten. Daher müssen die Verfahren für alle vier Trefferarten anwendbar sein. Zum Anfragezeitpunkt kann in Abhängigkeit von einer Anfrage und einer Trefferart eines der vier Berechnungsverfahren automatisch ausgewählt werden.

Alle Verfahren basieren auf der gleichen Indexstruktur, den im letzten Abschnitt eingeführten invertierten Listen. Zur intuitiven Darstellung wird für einen Listeneintrag $E$

$$d(E) := \pi_1(E) \text{ und } D(E) := \pi_2(E)$$

definiert. $d(E)$ entspricht dem Objekt $d$ aus dem Dokument $D = D(E)$, für das der Eintrag $E$ berechnet wurde (vgl. Definition 3.3.23).

Gegeben sei für eine Datenbasis $\mathcal{D}$ das Vokabular $\mathcal{V}$ und alle invertierten Listen $L(v), v \in \mathcal{V}$. Vokabular und invertierte Listen sind unabhängig von einer Anfrage und einer Trefferart und können somit vorab berechnet werden. Weiterhin wird davon ausgegangen, daß für jedes Dokument $D$ der Datenbasis die Kardinalität $|D|$ schnell ermittelt werden kann.

### 3.4.1 $\psi$-Verfahren

Für eine Anfrage $Q = \{q_1, \ldots q_m\}$ zusammen mit einer Trefferart $T \in \{$ IC, I, SC, S$\}$ werden nun die ersten beiden Verfahren zur Berechnung aller $T$-Treffer besprochen. Ein erstes Verfahren ist Algorithmus `A1`, der auf Seite 72 dargestellt ist und auf den im folgenden detailiert eingegangen wird. Die Kombination von einzelnen Vorkommen zu Treffern basiert auf Satz 3.3.8. Dabei wird die Kombination zweier Abbildungen $f : A \to C$, $g : B \to D$, für die $f \downarrow A \cap B = g \downarrow A \cap B$ gilt, bezeichnet mit

$$\begin{array}{rcl} f \cup g \ : \ A \cup B & \to & C \cup D \\ x & \mapsto & \left\{ \begin{array}{ll} f(x) & \text{falls } x \in A \\ g(x) & \text{falls } x \in B. \end{array} \right. \end{array} \tag{3.31}$$

Der Algorithmus gliedert sich in zwei Teile: Initialisierung und Schleifeniteration. Das Verfahren arbeitet sukzessive alle Objekte der Anfrage $Q = \{q_1, \ldots, q_m\}$ ab: bei der Initialisierung das erste Anfrageobjekt $q_1$ und pro Schleifendurchlauf immer ein weiteres, d.h. $q_2$ bis $q_m$.

Zwischenergebnisse werden in zwei Mengen $F$ und $F'$ von Abbildungen $\psi$ gespeichert. $F'$ ist dabei die Menge aus dem letzten Schleifendurchlauf bzw. der Initialisierung. $F$ wird pro Schleifendurchlauf neu aufgebaut.

Abbildungen $\psi : Q \to D$ werden auf zwei verschiedene Arten konstruiert. Einmal aus einem Eintrag $E$ einer invertierten Liste. Diese Konstruktion ist abhängig von der gewählten Trefferart $T$. Für I-Treffer und IC-Treffer lassen sich alle Trefferabbildungen $\psi^{q,d} : Q^q \to D^d$ aus dem Eintrag $E$ nach Satz 3.3.29 konstruieren, wobei $q$ das aktuell betrachtete Objekt der Anfrage und $d$ das Dokumentobjekt bzgl. $E$ ist, d.h. $d = d(E) \in D(E)$. $d$ ist hier ein I- bzw. IC-Vorkommen von $q$. Für SC- und S-Treffer ist $d$ ein S-Vorkommen von $q$ und alle Trefferabbildungen $\psi^{q,d}$ lassen sich nach Satz 3.3.37 konstruieren. Im Programm befindet sich die Konstruktion bei der Initialisierung in Zeile 15 und in der Schleife in Zeile 24.

**Algorithmus 1:** Gegeben sei zu jeder Vokabel $v$ des Vokabulars $\mathcal{V}$ zur Datenbasis $\mathcal{D}$ die invertierte Liste $L(v)$.
**Eingabe:** Anfrage $Q = \{q_1, \ldots q_m\}$, Trefferart $T$.
**Ausgabe:** Menge $F$ aller Trefferabbildungen.
ComputeHitsBF$(Q, T)$

(1) // *Deklarationen:*
(2) Menge von Abbildungen: $F, F'$;
(3) Injektive Abbildung $\psi^{q,d} : Q^q \to D^d$,
(4) Partielle injektive Abbildungen: $\psi, \hat{\psi} : Q \to D$;
(5) Vokabel: $v^q, v$;
(6) Invertierte Liste: $L$;
(7) Listeneintrag: $E$;
(8) // *Init bzgl.* $q_1 \in Q$*:*
(9) $F := \{\}$;
(10) $v^q := \mathtt{v}(q_1, Q)$;
(11) **for each** $v \in \mathcal{V} : v$ erfüllt Satz 3.3.29 (IC- und I-Treffer) bzw. Satz 3.3.37 (SC- und S-Treffer) bzgl. $v^q$:
(12) $L := L(v)$;
(13) **for each** $E \in L$:
(14) **if** $(T \in \{I, S\}) \vee (T \in \{IC, SC\} \wedge |D(E)| = |Q|)$
(15) **for each** $\psi^{q,d(E)} : Q^q \to D(E)^{d(E)}$ bzgl. $E$ nach Satz 3.3.29 für IC- und I-Treffer und nach Satz 3.3.37 für SC- und S-Treffer:
(16) $F := F \cup \{\psi^{q,d(E)}\}$;
(17) // *Schleife bzgl.* $Q \setminus \{q_1\}$*:*
(18) **for** $i := 2$ **to** $|Q|$
(19) $F' := F$; $F := \{\}$;
(20) $v^q := \mathtt{v}(q_i, Q)$;
(21) **for each** $v \in \mathcal{V}$ : bzgl. $v^q$ analog zu Zeile 11:
(22) $L := L(v)$;
(23) **for each** $E \in L$:
(24) **for each** $\psi^{q_i,d(E)} : Q^{q_i} \to D(E)^{d(E)}$ analog zu Zeile 15, $\hat{\psi} : \hat{Q} \to \hat{D} \in F'$ mit $\hat{D} \subseteq D(E)$:
(25) // *"Schnittbildung" bzw. Kombination:*
(26) **if** $i = \left|\{\psi^{q_i,d(E)}(q_i)\} \cup \{\hat{\psi}(q_j) | 1 \le j \le i-1\}\right|$ (vgl. (3.2))
$\wedge \forall p \in \hat{Q} \cap Q^{q_i} : \hat{\psi}(p) = \psi^{q_i,d(E)}(p)$ (vgl. (3.3))
(27) $F := F \cup \{\hat{\psi} \cup \psi^{q_i,d(E)}\}$; // *vgl. (3.31)*
(28) // *Ende der Schleife.*
(29) **return** $F$

Die zweite Art der Konstruktion von Abbildungen $\psi$ erfolgt in Zeile 27. Dies entspricht der sukzessiven Kombination von Vorkommen bzw. Trefferabbildungen nach Satz 3.3.8. Der Definitionsbereich von $\psi$ ist eine Teilmenge von $Q$ und ergibt sich aus den bisher betrachteten Anfrageobjekten und deren Umgebung. Pro Durchlauf der äußeren Schleife aus Zeile 18 vergrößert sich der Definitionsbereich, bis schließlich ganz $Q$ erfaßt ist. $\hat{\psi}$ ist dabei eine Abbildung, die beim letzten äußeren Schleifendurchlauf bzw. bei der Initialisierung berechnet wurde.

Wie bereits besprochen, erfolgt die Kombination der Vorkommen nach Satz 3.3.8. In Zeile 26 werden die Voraussetzungen des Satzes überprüft. Hier wird für jedes Objekt $q_i$ der Anfrage ein $T$-Vorkommen von $q_i$ mit lokaler Trefferabbildung $\psi^{q_i,d(E)}$ benötigt. Die Voraussetzungen des Satzes werden im Verfahren in abgewandelter Form überprüft, um bereits pro Teilschritt $i$ entscheiden zu können, ob die Voraussetzungen verletzt werden. Voraussetzung (3.2) wird für alle Schleifendurchläufe $2 \leq i \leq |Q|$ durch

$$i = \left| \{ \psi^{q_j,d(E)}(q_j) | 1 \leq j \leq i \} \right|$$

überprüft. Im Verfahren ist $\psi^{q_j,d(E)} = \hat{\psi} \downarrow Q^{q_j}$ für $1 \leq j \leq i-1$, da $\hat{\psi}$ in den vorangegangenen $i-1$ Schritten aus $\psi^{q_j,d(E)}$ konstruiert wurde.

Voraussetzung (3.3) von Satz 3.3.8 ergibt sich im Verfahren für den $i$-ten Schleifendurchlauf mit

$$\psi^{q_i,d(E)}(p) = \psi^{q_j,d(E)}(p),\ \forall p \in Q^{q_i} \cap Q^{q_j}, 1 \leq j \leq i-1.$$

Im einzelnen stellt sich der Programmablauf wie folgt dar. Bei der Initialisierung wird die Vokabel von $q_1$ berechnet und abhängig von der Trefferart $T$ nach den Vokabeln $v$ im Vokabular $\mathcal{V}$ gesucht, die Vorkommen von $q_1$ definieren (Sätze 3.3.29 und 3.3.37). Diese Vokabeln $v$ werden sukzessive durchlaufen (Zeile 11), wobei jeweils die entsprechende invertierte Liste $L(v)$ vom Massenspeicher in den Hauptspeicher transferiert wird (Zeile 12) und deren Einträge $E \in L(v)$ durchlaufen werden (Zeile 13). Jeder Eintrag bestimmt ein Vorkommen $d$ von $q_1$ mit $d = d(E)$ und $d \in D(E)$. In Zeile 15 wird für IC- und SC-Treffer die Kardinalität des Dokuments $D(E)$ überprüft, was bei der Trefferberechnung bzgl. Satz 3.3.8 (1) und (3) benötigt wird. Wie bereits oben besprochen werden dann alle Trefferabbildungen $\psi^{q,d} : Q^q \to D^d$ berechnet und in $F$ zwischengespeichert.

Mittels der äußeren Schleife aus Zeile 18 wird jedes Anfrageobjekt $q_i \in Q \setminus \{q_1\}$ abgearbeitet. $F'$ stellt die Menge der Abbildungen $\hat{\psi}$ dar, die im letzten Schleifendurchlauf konstruiert wurden. Beim ersten Schleifendurchlauf ist $F'$ die in der Initialisierung berechnete Menge $F$ bzgl. $q_1$. Im olgenden wird $F$ analog zur Initialisierung neu berechnet, wobei in $F$ Kombinationen von Trefferabbildungen nach Satz 3.3.8 festgehalten werden. Dabei werden immer zwei Abbildungen $\psi^{q,d}$ und $\hat{\psi}$ miteinander kombiniert, deren Bilder eine Teilmenge des gleichen Dokuments $D(E)$ darstellen. $\psi^{q,d} : Q^q \to D^d$ stammt mit $q = q_i$ und $d = d(E)$ aus einem Listeneintrag $E$, $\hat{\psi} : \hat{Q} \to \hat{D}$ aus $F'$ mit $\hat{Q} \subseteq Q$, $\hat{D} \subseteq D$. In Zeile 26 werden die Voraussetzungen von Satz 3.3.8 überprüft. Werden diese erfüllt, wird eine neue Abbildung $\psi = \hat{\psi} \cup \psi^{q_i,d(E)}$ bzgl. (3.4) aus dem Beweis von Satz 3.3.8 und (3.31) konstruiert und in $F$ zwischengespeichert.

Wurden alle Anfrageobjekte abgearbeitet, enthält $F$ nach Satz 3.3.8 alle Trefferabbildungen $\Psi : Q \to D$ bzgl. $Q$ und Trefferart $T$. Ist $F$ leer, existiert kein Treffer. Die Menge aller Treffer ergibt sich aus $F$ durch $\{D \in \mathcal{D} | \exists \Psi : Q \to D \in F\}$.

**Beispiel 3.4.1.** Betrachten wir die in Abbildung 3.15 dargestellte Anfrage $G_Q$ mit $Q = \{q_1 = r, q_2 = s, q_3 = t\}$ bzgl. einer Datenbasis, die nur aus dem in Abbildung 3.8 b) dargestellten Dokument besteht. In Beispiel 3.3.27 sind alle invertierten Listen bzgl. dem Vokabular $\mathcal{V} = \{(\emptyset, \{(\alpha,1)\}, \emptyset), (\emptyset, \{(\beta,1)\}, \emptyset), (\{(\alpha,2),(\beta,1)\}, \emptyset, \emptyset)\}$ aufgeführt. Im folgenden wird die Berechnung aller I-Treffer besprochen.

Die Initialisierung erfolgt bzgl. $q_1 = r$ mit $v^r := \mathrm{v}(r, G_Q) = (\emptyset, \{(\alpha,1)\}, \emptyset)$. In Zeile 11 erfüllt die Vokabel $v = v^r$ als einzige Vokabel des Vokabulars die Bedingungen aus Satz 3.3.29. Die invertierte Liste enthält zwei Einträge $E_a = [a, D, \emptyset, \{\{d\}\}, \emptyset]$ und $E_c = [c, D, \emptyset, \{\{d\}\}, \emptyset]$.

Der Eintrag $E_a$ erzeugt in Zeile 15 die Abbildung $\psi^{r,a} : Q^r \to D^a$ mit $r \mapsto a, t \mapsto d$ und aus $E_c$ ergibt sich $\psi^{r,c} : Q^r \to D^c$ mit $r \mapsto c, t \mapsto d$. Die Initialisierung wird also mit $F = \{\psi^{r,a}, \psi^{r,c}\}$ beendet.

Beim ersten Durchlauf der Schleife in Zeile 18 für $q_2 = s$ ist $F'$ gleich der bei der Initialisierung berechneten Menge $F$ von Abbildungen. In $F$ werden nun pro Schleifendurchlauf die neu berechneten Abbildungen zwischengespeichert, wobei $F$ bei jedem neuen Schleifendurchlauf als leere Menge startet. Die Vokabel von $s$ ergibt sich als $v^s = \big(\emptyset, \{(\beta, 1)\}, \emptyset\big)$.

In Zeile 21 erfüllt nur $v \in \mathcal{V}$ mit $v = v^s$ die Bedingungen aus Satz 3.3.29. Der Eintrag $E_b = [b, D, \emptyset, \{\{d\}\}, \emptyset]$ bildet das einzige Element der invertierten Liste $L(v)$. Hieraus ergibt sich in Zeile 24 die Abbildung $\psi^{s,b} : Q^s \to D^b$ mit $s \mapsto b$ und $t \mapsto d$. Da $\hat{\psi}(\hat{Q}) \subseteq D(E)$ für beide Abbildungen $\hat{\psi} : \hat{Q} \to \hat{D}$ aus $F'$ gilt, werden die Abbildungen nacheinander auf eine Kombinationsmöglichkeit mit $\psi^{s,b}$ im Rumpf der Schleife in Zeile 24 untersucht.

Für $\hat{\psi} = \psi^{r,a}$ und $\psi^{q_i, d(E)} = \psi^{s,b}$ ist sowohl die erste Bedingung aus Zeile 26 mit

$$2 = i = \left|\{\psi^{s,b}(s)\} \cup \{\hat{\psi}(q_1)\}\right| = |\{b\} \cup \{a\}|$$

als auch die zweite Bedingung mit $\{t\} = \hat{Q} \cap Q^s$ und $\psi^{r,a}(t) = \psi^{s,b}(t) = d$ erfüllt. Somit wird in Zeile 27 eine neue Abbildung

$$\begin{array}{rcccl} \psi^{r,a} \cup \psi^{s,b} & : & Q^r \cup Q^s & \to & D^a \cup D^b \\ & & r & \mapsto & \psi^{r,a}(r) = a \\ & & s & \mapsto & \psi^{s,b}(s) = b \\ & & t & \mapsto & \psi^{r,a}(t) = d\ (= \psi^{s,b}(t)) \end{array}$$

definiert und in Zeile 27 in $F$ zwischengespeichert.

Auch für $\hat{\psi} = \psi^{r,c}$ und $\psi^{q_i, d(E)} = \psi^{s,b}$ sind beide Bedingungen in Zeile 26 erfüllt mit

$$2 = i = \left|\{\psi^{s,b}(s)\} \cup \{\hat{\psi}(q_1)\}\right| = |\{b\} \cup \{c\}|$$

und $\psi^{r,c}(t) = \psi^{s,b}(t) = d$. Hieraus ergibt sich eine neue Abbildung

$$\begin{array}{rcccl} \psi^{r,c} \cup \psi^{s,b} & : & Q^r \cup Q^s & \to & D^a \cup D^b \\ & & r & \mapsto & \psi^{r,c}(r) = c \\ & & s & \mapsto & \psi^{s,b}(s) = b \\ & & t & \mapsto & \psi^{r,c}(t) = d\ (= \psi^{s,b}(t)), \end{array}$$

die ebenfalls in $F$ zwischengespeichert wird. Nach dem ersten Durchlauf der äußeren Schleife aus Zeile 18 ist $F = \{\psi_1^{r,a} \cup \psi^{s,b},\ \psi^{r,c} \cup \psi^{s,b}\}$.

Beide in $F$ enthaltenen Abbildungen besitzen als Definitionsbereich $Q$ und als Bildbereich $D$. Diese ändern sich beim nächsten Schleifendurchlauf bzgl. $q_3 = t$ nicht. Somit enthält $F$ alle Trefferabbildungen $\Psi : Q \to D$. ∘

Das letzte Beispiel verdeutlicht, daß nicht alle Anfrageobjekte durchlaufen werden müssen. Dort standen bereits nach der Abarbeitung von $q_1$ und $q_2$ alle globalen Trefferabbildungen fest. Dies spiegelt sich in Satz 3.3.9 wider, der eine abgewandelte Form von Satz 3.3.8 darstellt und festlegt, wann die Betrachtung aller Objekte einer Basis $Q' \subseteq Q$ von $G_Q$ ausreicht. Für obiges Beispiel gilt dies für $Q' = \{r, s\}$ oder $Q' = \{t\}$. Um Algorithmus `A1` an Satz 3.3.9 anzupassen, sind folgende Änderungen notwendig: In Abhängigkeit von $G_Q$ wird eine Basis $Q'$ von $G_Q$ ausgewählt, d.h. $Q'$ ist eine Teilmenge von $Q$ und erfüllt Gleichung (3.9). O.B.d.A. sei $Q' = \{q_1, \ldots, q_{|Q'|}\}$. Dann ändert sich die äußere Schleife in Zeile 18 in

(18) **for** $i := 2$ **to** $|Q'|$ .

Es werden also nicht mehr alle Anfrageobjekte aus $Q$ durchlaufen, sondern nur die in der Basis $G_{Q'}$ enthaltenen.

Zum einen besitzt Satz 3.3.9 eine schärfere Voraussetzung als Satz 3.3.8, nämlich (3.10) vs. (3.2). Die erste Bedingung in Zeile 26 ändert sich in

$$\left|Q^{q_i} \cup \hat{Q}\right| = \left|\psi^{q_i,d(E)}(Q^{q_i}) \cup \hat{\psi}(\hat{Q})\right|.$$

Dies entspricht im $i$-ten Schleifendurchlauf

$$|\cup_{1\leq j\leq i} Q^{q_j}| = \left|\cup_{1\leq j\leq i} \psi^{q_j,d(E)}(Q^{q_j})\right|.$$

Zum anderen werden in Satz 3.3.9 für I-Treffer und IC-Treffer zusätzliche Anforderungen an die Vorkommen gestellt. An die in den Zeilen 15 und 24 konstruierten lokalen Trefferabbildungen $\psi^{q_i,d(E)}$ wird bei IC-Treffern zusätzlich

$$\forall p \in Q^{q_i} \setminus \hat{Q} : |\mathtt{U}(p, G_Q)| = \left|\mathtt{U}(\psi^{q_i,d(E)}(p), G_{D(E)})\right|$$

gefordert. Dies genügt wegen Bedingung (3.9) an $Q'$, aus der $Q = \cup_{q\in Q'} Q^q = Q$ folgt, und

$$\cup_{1\leq i\leq |Q'|}(Q^{q_i} \setminus \hat{Q}) = (\cup_{1\leq i\leq |Q'|}) \setminus \hat{Q} = Q \setminus \hat{Q}.$$

Bei I-Treffern muß zusätzlich

$$\forall p \in (Q^{q_i} \cup \hat{Q}) \setminus Q' : \left|\mathtt{U}(p, G_Q) \cap (Q^{q_i} \cup \hat{Q})\right| = \left|\mathtt{U}(\psi^{q_i,d(E)}(p), G_D) \cap (\psi^{q_i,d(E)}(Q^{q_i}) \cup \hat{\psi}(\hat{Q}))\right|$$

gelten, wobei $\hat{Q}$ für $i = 1$ bei der Initialisierung leer ist. Diese abgeänderte Variante wird als Algorithmus `A2` bezeichnet.

Algorithmus `A2` besitzt im Vergleich zu Algorithmus `A1` sowohl Vor- als auch Nachteile. Vorteilig ist, daß nicht alle Elemente von $Q$ durchlaufen werden müssen. In obigen Beispiel reichte $Q' = \{t\}$ im Vergleich zu $Q = \{r, s, t\}$. Ein gravierender Nachteil bei der IC- und I-Trefferberechnung ist der Zugriff auf $D$. Dies wird für die Überprüfung der in `A2` hinzugekommenen Bedingungen bei der Berechnung von $\mathtt{U}(d, G_D)$, $(p, d) \in Q \setminus Q' \times D$ mit $\psi^{q,d}(p) = d$, $q \in Q'$, benötigt. Diese Informationen sind nicht in den betrachteten Listeneinträgen gespeichert.

Bei der Berechnung von IC-Treffern läßt sich der Zugriff auf die Dokumente umgehen, da hier $d$ ein IC-Vorkommen eines Anfrageobjektes $q$ ist und die Vokabeln beider Objekte übereinstimmen. Somit kann direkt auf die invertierte Liste zugegriffen werden, die den von $d$ erzeugten Eintrag $E$ enthält. In diesem Listeneintrag sind alle Informationen über die Umgebung von $d$ gespeichert.

Für I-Treffer gilt die Gleichheit der Vokabeln von $q$ und $d$ nicht. Hier gilt nach Satz 3.3.29 $(\alpha, x) \in \pi_1(v_q)$ impliziert $(\alpha, x') \in \pi_1(v_d)$ mit $x' \geq x$, sowie analoge Aussagen für $\pi_2$ und $\pi_3$. Hierdurch wird allerdings nicht die invertierte Liste definiert, die den von $d$ erzeugten Listeneintrag $E_d$ enthält. Somit muß i.d.R. in mehreren invertierten Listen nach dem Eintrag $E_d$ gesucht werden. Die dadurch entstehenden Kosten müssen mit denen verglichen werden, die zum Laden des Dokuments und der anschließenden Berechnung der Umgebung von $d$ anfallen. Bei großen Dokumenten ist i.d.R. eine Suche in invertierten Listen vorzuziehen.

**Bemerkung 3.4.2.** Die Voraussetzungen der Sätze 3.3.8 und 3.3.9 aus Zeile 26 müssen nicht pro Schleifendurchlauf überprüft werden. Es reicht, wenn dies einmal am Ende des Verfahrens für alle in $F$ enthaltenen Abbildungen geschieht.

Das in der letzten Bemerkung beschriebene Vorgehen kann allerdings dazu führen, daß viele Abbildungen in $F$ abgespeichert werden, die sich nicht zu Trefferabbildungen erweitern lassen. Praktische Untersuchungen müssen zeigen, ob sich durch diese Alternative bessere Laufzeiten erzielen lassen.

### 3.4.2 $\phi$-Verfahren

Bei den beiden bisherigen Algorithmen wurden sowohl bei der Initialisierung bzgl. $q_1$ als auch in der Schleife bzgl. $q_i$, $2 \leq i \leq |Q|$, für jeden Listeneintrag alle möglichen lokalen Trefferabbildungen $\psi^{q_i,d(E)}$ berechnet. In den Bemerkungen 3.3.30 und 3.3.32 wurde untersucht, wie viele lokale Trefferabbildungen für ein Vorkommen von $q_i$ existieren. In Abschnitt 3.3.6 wurden Beispiele von Vorkommen besprochen, die eine hohe Anzahl an lokalen Trefferabbildungen besitzen. In Zeile 24 der bisher besprochenen Verfahren kann dies zu einer hohen Anzahl an Kombinationsmöglichkeiten von lokalen Trefferabbildungen führen.

Im folgenden wird eine Alternative beschrieben, mit der sich die Anzahl an zu speichernden lokalen Trefferabbildungen reduzieren läßt. Dabei tritt kein Informationsverlust auf. Für jedes Anfrageobjekt $q \in Q$ und jedes Vorkommen $d_q \in D$ von $q$ werden die Daten aller lokalen Trefferabbildungen $\psi^{q,d_q} : Q^q \to D^{d_q}$ in einer einzigen Abbildung $\phi^{q,d_q} : Q^q \to 2^{D^{d_q}}$ gespeichert.

Nach Definition 3.3.7 wird für ein Vorkommen $d \in D$ von $q \in Q$ mit $\Psi^{q,d}$ die Menge aller lokalen Trefferabbildungen $\psi^{q,d} : Q^q \to D^d$ bezeichnet. Die Abbildung $\phi^{q,d} : Q^q \to 2^{D^d}$ ergibt sich aus $\Psi^{q,d}$ durch

$$\phi^{q,d}(p) = \{\psi^{q,d}(p) | \psi^{q,d} \in \Psi^{q,d}\} \tag{3.32}$$

für alle $p \in Q^q$. Diese Abbildung besitzt die Eigenschaft, daß $q$ auf die Menge $\{d\}$ abgebildet wird, d.h. das einzige Element der Menge ist das Vorkommen $d$ von $q$, bzgl. dem $\phi^{q,d}$ definiert wurde.

**Satz 3.4.3.** *Sei $\phi^{q,d} : Q^q \to 2^{D^d}$ nach Gleichung (3.32) für das Vorkommen $d \in D$ von $q \in Q$ definiert. Dann ist jede injektive Abbildung $\psi^{q,d} : Q^q \to D^d$ mit $\psi(p) \in \phi^{q,d}(p)$ für alle $p \in Q^q$ eine lokale Trefferabbildung.*

**Beweis** Die Aussage folgt aus dem Beweis der Rückrichtung von Satz 3.3.29 und Satz 3.3.37. □

Die Anzahl an verschiedenen lokalen Trefferabbildungen, die sich aus einer Abbildung $\phi^{q,d_q}$ konstruieren lassen, wurde in den Bemerkungen 3.3.30 und 3.3.32 angegeben.

In Satz 3.3.8 wurde untersucht, unter welchen Voraussetzungen sich für eine Anfrage $Q$ und ein Dokument $D$ aus Vorkommen $d_q \in D$ von $q \in Q$ eine globale Trefferabbildung $\Psi : Q \to D$ bilden läßt. Dies wird nun für Abbildungen $\phi^{q,d_q}$ statt $\psi^{q,d_q}$ untersucht.

**Satz 3.4.4.** *Sei $T \in \{IC, I, SC, S\}$ eine Trefferart, $G_Q$ und $G_D$ beschriftete Graphen und $(q, d_q) \in Q \times D$, so daß $d_q$ ein $T$-Vorkommen von $q$ ist. Ferner sei für alle $q \in Q$ die Abbildung $\phi^{q,d_q} : Q^q \to 2^{D^{d_q}}$ nach Gleichung (3.32) definiert. Sei $\phi : Q \to 2^D$ definiert durch*

$$p \mapsto \cap_{q \in Q : p \in Q^q} \phi^{q,d_q}(p). \tag{3.33}$$

*Genau dann ist $\Psi : Q \to D$, $\Psi(q) = d_q$, eine globale $T$-Trefferabbildung, wenn*

1. *$\forall q \in Q : \phi(q) = \phi^{q,d_q}(q) = \{d_q\}$ und*
2. *$|Q| = |\cup_{q \in Q} \phi(q)|$ und*
3. *$T \in \{IC, SC\} \Rightarrow |Q| = |D|$ gilt.*

**Beweis** "⇒": Sei $\Psi : Q \to D$ eine globale Trefferabbildung für einen $T$-Treffer $G_D$ von $G_Q$ mit $\Psi(q) = d_q$. Die drei Punkte werden nacheinander bewiesen.

Nach Satz 3.3.6 bildet für alle $q \in Q$ die Abbildung $\psi^{q,d_q} : Q^q \to D^{d_q}$ mit $\psi^{q,d_q} = \Psi \downarrow Q^q$ eine lokale Trefferabbildung mit $\psi^{q,d_q}(q) = d_q$. Nach Gleichung (3.32) gilt für alle $q \in Q$: $\phi^{q,d_q}(q) = \{\Psi(q)\} = \{d_q\}$. Weiterhin gilt für alle $q \in Q, p \in Q^q : d_p \in \phi^{q,d_q}(p)$, da $\psi^{q,d_q}(p) = \Psi(p) = d_p$. Also ist für alle $p \in Q$: $d_p \in \phi(p)$, da $d_p \in \cap_{q \in Q, p \in Q^q} \phi^{q,d_q}(p)$. Wegen $\phi^{p,d_p} = \{d_p\}$ gilt $\phi(p) = \{d_p\}$ für alle $p \in Q$. Hiermit ist der erste Punkt bewiesen.

Die beiden anderen Punkte ergeben sich aus der Injektivität von $\Psi$ und aus der Eigenschaft, daß $\Psi$ für IC- und SC-Treffer eine Bijektion ist.

"⇐": Für das $T$-Vorkommen $d_q$ von $q$ für alle $q \in Q$ gilt nach der Konstruktion von $\phi^{q,d_q}$ nach Gleichung (3.32) $\phi^{q,d_q}(q) = \{d_q\} = \phi(q)$.

$\psi^{q,d_q} : Q^q \to D^{d_q}$ definiert durch $p \mapsto d_p$ ist wegen Satz 3.4.3 eine lokale $T$-Trefferabbildung.

Nach Satz 3.3.8 ist $\Psi : Q \to D$ mit $q \mapsto d_q$ eine globale $T$-Trefferabbildung. Die im Satz geforderten Bedingungen sind durch die letzten beiden Punkte erfüllt. □

**Beispiel 3.4.5.** Betrachten wir nochmals die Anfrage $G_Q$ aus Abbildung 3.15 und Dokument $G_D$ aus Abbildung 3.8 b). Hierauf wurde schon bei der Berechnung aller I-Treffer mit Algorithmus `A1` eingegangen. Jetzt soll Satz 3.4.4 angewendet werden.

Für $t \in Q$ bildet $d \in D$ das einzige I-Vorkommen von $t$. Aus den beiden lokalen Trefferabbildungen $\psi_1^{t,d}$ und $\psi_2^{t,d}$ ergibt sich $\phi^{t,d}$ als

$$\begin{array}{rcl} \psi_1^{t,d} : Q^t & \to & D^d \\ r & \mapsto & a \\ s & \mapsto & b \\ t & \mapsto & d \end{array} \qquad \begin{array}{rcl} \psi_2^{t,d} : Q^t & \to & D^d \\ r & \mapsto & c \\ s & \mapsto & b \\ t & \mapsto & d \end{array} \qquad \begin{array}{rcl} \phi^{t,d} : Q^t & \to & 2^{D^d} \\ r & \mapsto & \{a, c\} \\ s & \mapsto & \{b\} \\ t & \mapsto & \{d\}. \end{array}$$

Die beiden lokalen Trefferabbildungen unterscheiden sich nur im Bild von $r$. Die Menge $\phi^{t,d}(r)$ besteht aus diesen beiden Bildern. $\phi^{t,d}$ repräsentiert beide lokalen Trefferabbildungen in einer Abbildung.

Für $s \in Q$ und das I-Vorkommen $b \in D$ existiert eine lokale Trefferabbildung $\psi^{s,b}$, aus der sich $\phi^{s,b}$ ergibt:

$$\begin{array}{rcl} \psi^{s,b} : Q^s & \to & D^b \\ s & \mapsto & b \\ t & \mapsto & d \end{array} \qquad \begin{array}{rcl} \phi^{s,b} : Q^s & \to & 2^{D^b} \\ s & \mapsto & \{b\} \\ t & \mapsto & \{d\}. \end{array}$$

Das Anfrageobjekt $r$ besitzt zwei verschiedene I-Vorkommen in $D$: $a$ und $c$, jeweils mit einer lokalen Trefferabbildung. Jedes dieser beiden Vorkommen definiert eine Abbildung $\phi$:

$$\begin{array}{rcl} \psi^{r,a} : Q^r & \to & D^a \\ r & \mapsto & a \\ t & \mapsto & d \end{array} \qquad \begin{array}{rcl} \phi^{r,a} : Q^r & \to & 2^{D^a} \\ r & \mapsto & \{a\} \\ t & \mapsto & \{d\} \end{array} \qquad \left| \quad \begin{array}{rcl} \psi^{r,c} : Q^r & \to & D^c \\ r & \mapsto & c \\ t & \mapsto & d \end{array} \qquad \begin{array}{rcl} \phi^{r,c} : Q^r & \to & 2^{D^c} \\ r & \mapsto & \{c\} \\ t & \mapsto & \{d\}. \end{array} \right.$$

Nun werden die Voraussetzungen von Satz 3.4.4 für $\phi^{t,d}, \phi^{s,b}$ und $\phi^{r,a}$ für das I-Vorkommen $a$ von $r$ untersucht. Berechnet man die Abbildung $\phi : Q \to 2^D$ nach Gleichung (3.33) aus Satz 3.4.4, so ergibt sich

$$\begin{array}{rcl} r \mapsto & \{a, c\} \cap \{a\} & = \{d_r = a\} \\ s \mapsto & \{b\} \cap \{b\} & = \{d_s = b\} \\ t \mapsto & \{d\} \cap \{d\} \cap \{d\} & = \{d_t = d\}. \end{array}$$

Da das Bild von $\phi$ für alle $q \in Q$ einelementig ist und die Kardinalitäten von $Q$ und $\cup_{q \in Q} \phi(q)$ gleich sind, folgt aus Satz 3.4.4, daß $D$ ein I-Treffer von $Q$ mit Trefferabbildung $\Psi : Q \to D$, $q \mapsto x, \phi(q) = \{x\}$ ist.

Analog ergibt sich die andere Trefferabbildung, wenn man Satz 3.4.4 für das I-Vorkommen $c$ von $r$ anwendet. ∘

Die Vorzüge der Modellierung der lokalen Trefferabbildungen durch $\phi$ an Stelle von $\psi$ liegen in der geringeren Anzahl an Abbildungen, die verarbeitet werden müssen. Betrachtet man ein Vorkommen $d_q \in D$ von $q \in Q$, so können hierfür mehrere lokale Trefferabbildungen existieren. Sei $l^{q,d_q}$ die Anzahl an lokalen Trefferabbildungen, die sich aus Bemerkung 3.3.30 bzw. 3.3.32 ergibt. Alle $l^{q,d_q}$ lokalen Trefferabbildungen bzgl. $q$ und $d_q$ werden durch eine einzige Abbildung $\phi$ beschrieben. Bei der Kombination von lokalen Trefferabbildungen nach Satz 3.3.8 werden neue Trefferabbildungen erzeugt. Im schlechtesten Fall kann es $l^{q,d_p} \cdot l^{p,d_p}$ neue Trefferabbildungen bzgl. den Vorkommen $d_q, d_p$ von $q$ bzw. $p$ geben. Bei der Kombination von Abbildungen $\phi^{q,d_q}$ und

$\phi^{p,d_p}$ entsteht maximal eine neue Abbildung. Hierin liegt ein wesentlicher Vorteil der $\phi$-Verfahren gegenüber den $\psi$-Verfahren.

In Satz 3.3.9 wurde untersucht, unter welchen Voraussetzungen nicht alle Objekt einer Anfrage $Q$ zur Trefferberechnung bzgl. Vorkommen betrachtet werden müssen, sondern wann die Abarbeitung aller Objekte einer Basis $Q' \subseteq Q$ von $G_Q$ ausreicht. Vor der Formulierung einer zu Satz 3.3.9 analogen Aussage für Abbildungen $\phi^{q,d_q}$ wird eine dafür benötigte Definition eingeführt.

**Definition 3.4.6.** Für eine Abbildung $\phi : Q \to 2^D$ bezeichne $\mathcal{G}(\phi)$ den ungerichteten bipartiten Graphen $G = (Q, \cup_{p \in Q}\phi(p), E)$ mit Kantenmenge $E = \{\{p, a\} | p \in Q, a \in \phi(p)\}$.
Falls $\mathcal{G}(\phi)$ ein Matching $M$ mit $|M| = |Q|$ besitzt, wird die injektive Abbildung definiert durch

$$\begin{aligned} \Psi^{M,\phi} : Q &\to D \text{ durch} \\ p &\mapsto a, (p, a) \in M. \end{aligned} \tag{3.34}$$

Wenn klar ist, auf welche Abbildung $\phi$ und welchen Graphen $\mathcal{G}(\phi)$ sich ein Matching $M$ bezieht, wird die Kurzschreibweise $\Psi^M$ an Stelle von $\Psi^{M,\phi}$ verwendet.

Die letzte Definition ist eine allgemeinere Form von Definition 3.3.41, wo ein Graph speziell für ein S-Vorkommen $d \in D$ von $q \in Q$ konstruiert wurde. Beide Arten von Graphen werden zur Berechnung von Matchings verwendet.

**Satz 3.4.7.** *Seien $G_Q$ und $G_D$ beschriftete Graphen und $Q' \subseteq Q$ eine Basis von $G_Q$ bzgl. des Anfragegraphen $G_Q$. Für jedes $q \in Q'$ sei $d_q \in D$ ein Vorkommen von $q$ und $\phi^{q,d_q} : Q^q \to D^{d_q}$ bzgl. $\Psi^{q,d_q}$ nach Gleichung (3.32) definiert. Sei $\phi : Q \to 2^D$ definiert durch*

$$p \mapsto \cap_{q \in Q' : p \in Q^q} \phi^{q,d_q}(p). \tag{3.35}$$

*Für $Q'$ und $T \in \{IC, I, SC, S\}$ gilt folgende Äquivalenz:*

*(1) Es existiert eine globale $T$-Trefferabbildung $\Psi : Q \to D$ mit $\Psi(q) = d_q$ für alle $q \in Q'$.*

*(2) Fall $T = IC$: Es gilt $|Q| = |D|$, für alle $q \in Q'$ ist $d_q$ ein IC-Vorkommen von $q$ und $\mathcal{G}(\phi)$ besitzt ein Matching $M$ mit $|Q| = |M|$, für das $\forall p \in Q \setminus Q' : |\mathtt{U}(p, G_Q)| = \left|\mathtt{U}(\Psi^M(p), G_D)\right|$ gilt.*

*Fall $T = I$: Es gilt: Für alle $q \in Q'$ ist $d_q$ ein I-Vorkommen von $q$ und $\mathcal{G}(\phi)$ besitzt ein Matching $M$ mit $|Q| = |M|$, für das $\forall p \in Q \setminus Q' : |\mathtt{U}(p, G_Q)| = \left|\mathtt{U}(\Psi^M(p), G_D) \cap \Psi^M(Q)\right|$ gilt.*

*Fall $T = SC$: Es gilt $|Q| = |D|$, für alle $q \in Q'$ ist $d_q$ ein S-Vorkommen von $q$ und $\mathcal{G}(\phi)$ besitzt ein Matching $M$ mit $|Q| = |M|$.*

*Fall $T = S$: Es gilt: Für alle $q \in Q'$ ist $d_q$ ein S-Vorkommen von $q$ und $\mathcal{G}(\phi)$ besitzt ein Matching $M$ mit $|Q| = |M|$.*

**Beweis** Alle vier Fälle werden zusammen bewiesen.
"(1) $\Rightarrow$ (2)": Sei $\Psi : Q \to D$ eine $T$-Trefferabbildung mit $\Psi(q) = d_q$ für alle $q \in Q'$. Dann ist nach Satz 3.3.6 für alle $q \in Q$ die Abbildung $\psi^{q,\Psi(q)} : Q^q \to D^{\Psi(q)}$, $\psi^{q,\Psi(q)} = \Psi \downarrow Q^q$, eine lokale Trefferabbildung für das $T$-Vorkommen $\Psi(q)$ von $q$. Nach Konstruktion von $\phi^{q,d_q}$ für alle $q \in Q'$ bzgl. dem $T$-Vorkommen $d_q$ von $q$ ist $\psi^{q,d_q}(p) \in \phi^{q,d_q}(p), \forall q \in Q, p \in Q^q$ und somit $\Psi(p) \in \phi(p), \forall p \in Q$. Aus der Konstruktion von $\mathcal{G}(\phi)$ folgt, daß $M = \{\{q, \Psi(q)\} | q \in Q\}$ ein Matching mit $|M| = |Q|$ ist. Setzt man $\Psi^M = \Psi$, so ergeben sich hieraus die Aussagen bzgl. der einzelnen Trefferarten aus den Eigenschaften der Trefferabbildung (vgl. Satz 3.3.9).
"(1) $\Leftarrow$ (2)": Für die Rückrichtung wird Satz 3.3.9 angewendet. Hierfür wird nachgewiesen, daß die Voraussetzungen von Satz 3.3.9 im vorliegenden Fall erfüllt sind.
Nach Konstruktion von $\phi$ gilt $\phi(q) = \{d_q\}$ für alle $q \in Q'$. Aus der Konstruktion von $\mathcal{G}(\phi)$ folgt, daß $\{q, d_q\} \in M$ für alle $q \in Q'$ gilt. Aus $\Psi^M$ lassen sich für alle $q \in Q'$ lokale Trefferabbildungen $\psi^{q,d_q} : Q^q \to D^{d_q}$, $\psi^{q,d_q} = \Psi^M \downarrow Q^q$ konstruieren, wobei die entsprechende Trefferart für alle vier

Fälle durch die Art der gegebenen Vorkommen gewährleistet ist.
Es gilt $|Q| = \left|\cup_{q \in Q'} \psi^{q,d_q}(Q^q)\right| = \left|\Psi^M(Q')\right|$, da $\Psi^M$ injektiv ist, womit Voraussetzung (3.10) von Satz 3.3.9 erfüllt ist. Weiterhin gilt $\psi^{a,d_a}(p) = \psi^{b,d_b}(p) = \Psi^M(p)$ für alle $a, b \in Q', p \in Q^a \cap Q^b$. Also ist auch Voraussetzung (3.11) erfüllt. Die für die einzelnen Trefferarten notwendigen Zusatzbedingungen für Satz 3.3.9 ergeben sich direkt aus den pro Trefferart formulierten Bedingungen. Aus Satz 3.3.9 folgt die Behauptung. □

**Beispiel 3.4.8.** Nachdem in Beispiel 3.4.5 die Trefferberechnung für die Anfrage $G_Q$ aus Abbildung 3.15 und das Dokument $G_D$ aus Abbildung 3.8 b) im Zusammenhang mit Satz 3.4.4 besprochen wurde, wird nun auf die Anwendung von Satz 3.4.7 für $G_Q$ und $G_D$ eingegangen. Unter Anwendung dieses Satzes soll untersucht werden, ob $G_D$ einen I-Treffer von $G_Q$ darstellt.

Als erstes werden die Voraussetzungen von Satz 3.4.7 überprüft. Hierfür wird die Basis $Q' = \{t\}$ von $G_Q$ verwendet. $Q'$ erfüllt Gleichung (3.9), da die beiden restlichen Anfrageobjekte Nachbarknoten von $t$ in $G_Q$ sind.

Das einzige I-Vorkommen von $t$ bildet das Objekt $d$ in $D$. Für $t \in Q'$ und $d \in D$ ist $\phi^{t,d} : Q^t \to 2^{D^d}$ in Beispiel 3.4.5 dargestellt. Die in Gleichung (3.35) definierte Abbildung $\phi : Q \to 2^D$ stimmt mit $\phi^{t,d}$ überein.

Der Graph $\mathcal{G}(\phi)$ ist in Abbildung 3.16 dargestellt. Für $\mathcal{G}(\phi)$ bildet die Kantenmenge $M_1 = \{\{r,a\},\{s,b\},\{t,d\}\}$ ein vollständiges Matching. Das einzige von $M_1$ verschiedene vollständige Matching ist $M_2 = \{\{r,c\},\{s,b\},\{t,d\}\}$. Für $M_1$ ergibt sich die Abbildung $\Psi^{M_1} : Q \to D$ als $r \mapsto a, s \mapsto b$ und $t \mapsto d$. $\Psi^{M_2}$ gleicht $\Psi^{M_1}$ bis auf $\Psi^{M_2}(r) = c$.

Für I-Treffer wird weiterhin $\forall p \in Q \setminus Q' : |\mathtt{U}(p, G_Q)| = \left|\mathtt{U}(\Psi^{M_1}(p), G_D) \cap \Psi^{M_1}(Q)\right|$ gefordert. Hierfür ist $Q \setminus Q' = \{r, s\}$ und es gilt $|\mathtt{U}(r, G_Q)| = |\{t\}| = |\{d\}| = |\mathtt{U}(a, G_d)|$. Analoges gilt für $s$ und $\Psi^{M_1}(s)$. Somit ist $D$ ein I-Treffer von $Q$ mit Trefferabbildung $\Psi^{M_1}$.

Die Aussage ist auch für das Matching $M_2$ mit $\Psi^{M_2}$ erfüllt, d.h. die Menge aller Trefferabbildungen ist $\{\Psi^{M_1}, \Psi^{M_2}\}$. ∘

Das letzte Beispiel demonstriert die Vorteile von Satz 3.4.7 im Zusammenhang mit Abbildungen $\phi$. Es müssen i.d.R. nur wenige Anfrageobjekte abgearbeitet werden. In obigen Beispiel hat ein Objekt gereicht. Es wurde nur eine Abbildung $\phi$ benötigt, aus der sich alle Trefferabbildungen mit Hilfe von Matchings konstruieren lassen. Es reicht der Nachweis der Existenz eines Matchings, wenn nur untersucht werden soll, ob $G_D$ ein Treffer von $G_Q$ ist.

Als nächstes werden die Ergebnisse der letzten Sätze in Algorithmen umgesetzt.

Bei den bisher besprochenen Algorithmen zur Trefferberechnung wurden für ein Anfrageobjekt $q$ alle lokalen Trefferabbildungen $\psi^{q,d(E)} : Q^q \to D^{d(E)}$ aus einem Eintrag $E$ einer invertierten Liste berechnet. Aus den lokalen Trefferabbildungen wurde nach Gleichung (3.32) die Abbildung $\phi^{q,d(E)} : Q^q \to 2^{D^{d(E)}}$ konstruiert.

Sei $E$ ein Listeneintrag und $q$ ein Objekt einer Anfrage, so daß $d(E)$ ein Vorkommen von $q$ ist. Für jede der vier Trefferarten läßt sich die Abbildung $\phi^{q,d(E)}$ direkt aus einem solchen Listeneintrag $E$ berechnen.

Für IC- und I-Vorkommen ergibt sich $\phi^{q,d(E)}$ aus den im Listeneintrag gespeicherten Daten durch

$$p \mapsto \begin{cases} \{d\} & \text{falls } p = q \\ \mathtt{N}^{\setminus \mathtt{V}}(d, G_D, \xi_Q((q,p))) & \text{falls } p \in \mathtt{N}(q, G_Q) \setminus \mathtt{V}(q, G_Q) \\ \mathtt{V}^{\setminus \mathtt{N}}(d, G_D, \xi_Q((p,q))) & \text{falls } p \in \mathtt{V}(q, G_Q) \setminus \mathtt{N}(q, G_Q) \\ \mathtt{NV}(d, G_D, \xi_Q((q,p)), \xi_Q((p,q))) & \text{falls } p \in \mathtt{N}(q, G_Q) \cap \mathtt{V}(q, G_Q) \end{cases} \tag{3.36}$$

und für S-Vorkommen aus

$$p \mapsto \begin{cases} \{d\} & \text{falls } p = q \\ \mathtt{N}(d, G_D, \alpha') & \text{falls } p \in \mathtt{N}^{\setminus \mathtt{V}}(q, G_Q, \alpha) : \alpha' \geq \alpha \\ \mathtt{V}(d, G_D, \beta') & \text{falls } p \in \mathtt{V}^{\setminus \mathtt{N}}(q, G_Q, \beta) : \beta' \geq \beta \\ \mathtt{NV}(d, G_D, \gamma', \delta') & \text{falls } p \in \mathtt{NV}(q, G_Q, \gamma, \delta) : \gamma' \geq \gamma \wedge \delta' \geq \delta. \end{cases} \tag{3.37}$$

Die beiden bisher besprochenen Algorithmen `A1` und `A2` verwenden zur Trefferberechnung lokale Trefferabbildungen bzw. darauf aufbauend injektive Abbildungen $\psi : Q \to D$. Beide Verfahren

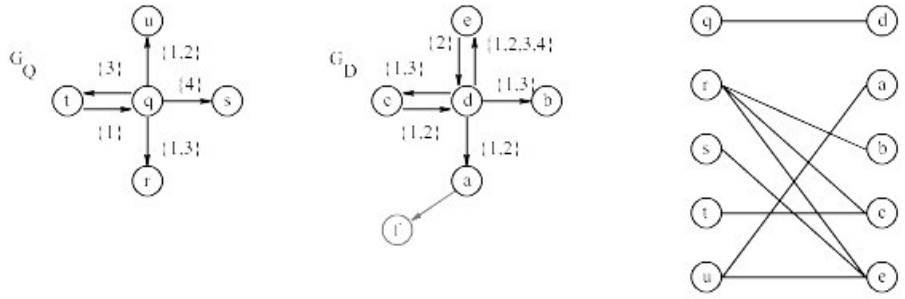



Abbildung 3.14: Ein Beispiel zu Definition 3.3.41. Rechts ist der bipartite Graph $\mathcal{G}(q, G_Q, d, G_D)$ dargestellt. Ein Matching ist $M = \big\{\{q,d\},\{r,b\},\{s,e\},\{t,c\},\{u,a\}\big\}$, woraus sich direkt eine Bijektion $\psi^{q,d} : Q \to D$ mit $\{p, \psi^{q,d}(p)\} \in M$ für alle $p \in Q$ ergibt. $d$ ist S-Vorkommen von $q$ und $D$ ist SC-Treffer von $Q$.


Abbildung 3.15: Die dargestellte Anfrage $G_Q$ dient zur Demonstration der Trefferberechnungsverfahren. Als Vokabeln ergeben sich für die drei Objekte der Anfrage $\mathtt{v}(r, G_Q) = \big(\emptyset, \{(\alpha,1)\}, \emptyset\big)$, $\mathtt{v}(s, G_Q) = \big(\emptyset, \{(\beta,1)\}, \emptyset\big)$ und $\mathtt{v}(t, G_Q) = \big(\{(\alpha,1),(\beta,1)\}, \emptyset, \emptyset\big)$.

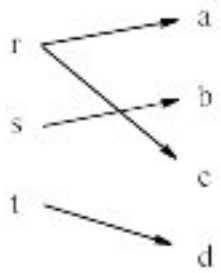



Abbildung 3.16: Der Graph $\mathcal{G}(\phi)$ aus Beispiel 3.4.8.

lassen sich mit Hilfe der Sätze 3.4.4 und 3.4.7 durch Verwendung von Abbildungen $\phi : Q \to 2^D$ an Stelle von lokalen Trefferabbildungen abändern. Hierauf wird als nächstes eingegangen.

Betrachten wir erst die Variante, die auf der Abarbeitung alle Objekte einer Anfrage $Q$ basiert. Verfahren `A1`, das lokale Trefferabbildungen verwendet, basiert auf Satz 3.3.8. Für die Verwendung von Satz 3.4.4 für Abbildungen $\phi$ muß die Generierung der Abbildungen in den Zeilen 15 und 24 abgeändert werden. Aus einem Eintrag $E$ einer invertierten Liste wird die Abbildung $\phi^{q,d(E)}$ für ein $q \in Q$ durch Gleichung (3.36) definiert.

Weiterhin müssen in Zeile 26 die Voraussetzungen an Satz 3.4.4 angepaßt werden. Da es für die später beschriebene Variante bzgl. Satz 3.4.7 vorteilhaft ist, erst die unter Gleichung (3.33) bzw. (3.35) beschriebene Abbildung $\phi$ zu berechnen und danach die Voraussetzungen des Satzes zu überprüfen, wird dieses Vorgehen schon an dieser Stelle angewendet.

In den bisherigen Verfahren wurde jeweils in Zeile 27 die Kombination zweier lokaler Trefferabbildungen definiert. Diese ergibt sich für zwei Abbildungen $\phi_1 : Q_1 \to 2^D$ und $\phi_2 : Q_2 \to 2^D$ als $\phi : Q_1 \cup Q_2 \to 2^D$ mit

$$\phi(p) \mapsto \begin{cases} \phi_1(p) & \text{falls } p \in Q_1 \setminus Q_2 \\ \phi_2(p) & \text{falls } p \in Q_2 \setminus Q_1 \\ \phi_1(p) \cap \phi_2(p) & \text{falls } p \in Q_1 \cap Q_2. \end{cases} \tag{3.38}$$

Für einen Listeneintrag $E$ ergibt sich $\phi : Q^{q_i} \cup \hat{Q} \to 2^{D(E)^{d(E)} \cup \hat{D}}$ aus $\phi^{q_i,d(E)} : Q^{q_i} \to 2^{D(E)^{d(E)}}$ und $\hat{\phi} : \hat{Q} \to 2^{\hat{D}}$ durch Gleichung (3.38).

Die Voraussetzungen für Satz 3.4.4 ergeben sich für den $i$-ten Durchlauf der äußeren Schleife aus

$$\forall q \in Q^{q_i} \cap \hat{Q} : \phi(q) \neq \emptyset$$

und

$$i = \left| \cup_{1 \leq j \leq i} \phi(q_j) \right| .$$

Die für IC- und SC-Treffer geforderte Gleichheit der Kardinalitäten von $Q$ und $D$ wird bereits in Zeile 15 überprüft. Erfüllt eine Abbildung $\phi$ diese Voraussetzungen, so wird sie in der Menge $F$ zwischengespeichert. Der so abgeänderte Algorithmus wird mit `A3` bezeichnet.

Der Vorteil von Satz 3.4.7 gegenüber Satz 3.4.4 besteht darin, daß nur für eine Basis $Q'$ von $G_Q$, also einer Teilmenge $Q'$ von $Q$, die Gleichung (3.9) erfüllt, die Anfrageobjekte abgearbeitet werden müssen. In der Initialisierung und äußeren Schleife werden also nur Objekte aus $Q'$ durchlaufen. Zusätzlich muß Algorithmus `A3` an die Voraussetzungen von Satz 3.4.7 angepaßt werden. Mit der oben beschriebenen Konstruktion von $\phi$ aus $\phi^{q_i,d(E)}$ und $\hat{\phi}$ ergibt sich eine Voraussetzung, die für alle Trefferarten gleich ist: $\mathcal{G}(\phi)$ besitzt ein vollständiges Matching. Für IC-Treffer wird zusätzlich

$$\forall p \in Q \setminus Q' : |\mathtt{U}(p, G_Q)| = \left|\mathtt{U}(\Psi^M(p), G_D)\right|$$

und für I-Treffer

$$\forall p \in Q \setminus Q' : |\mathtt{U}(p, G_Q)| = \left|\mathtt{U}(\Psi^M(p), G_D) \cap \Psi^M(Q)\right|$$

gefordert. Der bipartite Graph $\mathcal{G}(\phi)$ und die von einem vollständigen Matching $M$ induzierte Abbildung $\Psi^M : Q \to D$ wurde in Definition 3.4.6 eingeführt.

Diese Varianten des Berechnungsverfahrens wird mit `A4` bezeichnet.

### 3.4.3 Zusammenfassung und Fazit

In diesem Abschnitt wurden vier Verfahren zur Trefferberechnung vorgestellt. Alle basieren auf der in Abschnitt 3.3 beschriebenen Berechnung und Kombination von Vorkommen. Die Verfahren besitzen unterschiedliche Vor- und Nachteile. Die Verfahren `A1` und `A2` basieren auf lokalen Trefferabbildungen $\psi : Q \to D$. Da der Definitionsbereich aller in $F$ gespeicherten Abbildungen gleich ist, müssen nur die Bilder abgespeichert werden. Hierfür bietet sich ein einfaches eindimensionales Array der Größe $|Q|$ an, in dem nacheinander alle Bilder von $q_1$ bis $q_{|Q|}$ gespeichert werden. So

dargestellte Abbildungen lassen sich leicht kombinieren (vgl. Zeile 27 in `A1`). Die Darstellungsmöglichkeit einer lokalen Trefferabbildung $\psi$ durch eine einfache Datenstruktur ist ein Vorteil der beiden Verfahren. Im Vergleich zu den beiden anderen Verfahren werden i.d.R. bei `A1` und `A2` deutlich mehr lokale Trefferabbildungen $\psi$ erzeugt als Abbildungen $\phi$ bei `A3` und `A4`. Dafür ist die notwendige Datenstruktur zur Speicherung einer Abbildung $\phi : Q \rightarrow 2^D$ komplexer, da hier ein Array von Mengen verwaltet werden muß. Es hängt stark von der Anwendung ab, für welche Anfragen das eine oder das andere Vorgehen vorteilhafter ist. In vielen Fällen kann durch Heuristiken oder Expertenwissen die Auswahl zwischen einem $\psi$- und einem $\phi$-Verfahren getroffen werden.

Für $\psi$- und $\phi$-Verfahren existiert jeweils eine Variante, bei der alle Anfrageobjekte bearbeitet werden: `A1` und `A3`. Bei den anderen beiden Varianten `A2` und `A4` müssen zwar nur die Objekte einer Basis $Q'$ von $G_Q$ betrachtet werden, diese allerdings bzgl. verschärften Bedingungen. Bei IC- und I-Treffern sind zusätzlich Zugriffe auf die Dokumente erforderlich, wodurch sich die Varianten für diese Trefferarten nicht anbieten. Für SC- und S-Treffer bringen diese Varianten i.d.R. erhebliche Vorteile, da die Basis $Q'$ von $G_Q$ in Abhängigkeit der Anfrage und den Verteilungen auf die invertierten Listen getroffen werden kann. Hier bieten sich Ansätze zur Anfrageoptimierung an, auf die in Abschnitt 4.2.4 weiter eingegangen wird.

Zusammenfassend ergibt sich: Die Auswahl des Verfahrens hängt von der Trefferart ab. Für IC- und I-Treffer bieten sich die Verfahren `A1` und `A3` an, während sich für SC- und S-Treffer die Verfahren `A2` und `A4` anbieten.

Die Entscheidung für ein $\psi$- oder $\phi$-Verfahren hängt von der Anwendung bzw. der Anfrage und den Daten der Datenbasis ab.

# Kapitel 4

# Anwendung

In diesem Kapitel werden praktische Aspekte der Trefferberechnungsverfahren besprochen. Als erstes wird die Suche im Vokabular beschrieben. Danach wird auf eine Optimierung der Verfahren eingegangen. Neben einigen Grundlagen werden u.a. die Komprimierung von invertierten Listen und eine Anfrageoptimierung behandelt.

Schließlich werden Implementierungen der Verfahren im praktischen Einsatz untersucht. Hierbei werden u.a. die Verfahren untereinander verglichen und es wird auf Abhängigkeiten der Laufzeiten eingegangen.

## 4.1 Suche im Vokabular

Bei den in Abschnitt 3.4 angegebenen Verfahren zur Trefferberechnung wird für jedes Anfrageobjekt $q \in Q$ mit Hilfe der Vokabel $v^q := \mathtt{v}(q, G_Q)$ bestimmt, welche invertierte Listen $L(v), v \in \mathcal{V}$, abgearbeitet werden müssen. Für die Trefferart $T$ bezeichne $\mathcal{U}(q, G_Q, \mathcal{V}, T)$, kurz $\mathcal{U}_T^q$, die Menge dieser Vokabeln $v \in \mathcal{V}$. Die Sätze 3.3.29 und 3.3.37 legen für die vier Trefferarten fest, aus welchen $v \in \mathcal{V}$ sich $\mathcal{U}_T^q$ zusammensetzt.

Nach Satz 3.3.29 (1) gestaltet sich die Berechnung für IC-Treffern einfach: Es muß nur überprüft werden, ob $v^q \in \mathcal{V}$ gilt. Ist dies der Fall, wird die invertierte Liste $L(v^q)$ abgearbeitet. Ansonsten befindet sich kein Treffer von $Q$ in der Datenbasis. $\mathcal{U}_T^q$ ist also entweder leer oder $\{v^q\}$.

Für I-Treffer ist die Berechnung von $\mathcal{U}_T^q$ komplexer. Hier werden alle Vokabeln $v \in \mathcal{V}$ gesucht, die die einzelnen Elemente der drei Mengen jeweils "beinhalten". Dies wird in Satz 3.3.29 (2) definiert. Es ergibt sich für I-Treffer

$$\begin{aligned}\mathcal{U}_I^q = \{v \in \mathcal{V} \quad | \quad & (\alpha, x) \in \pi_1(v^q) \Rightarrow \exists x' : (\alpha, x') \in \pi_1(v) \wedge x \leq x', \\ & (\beta, y) \in \pi_2(v^q) \Rightarrow \exists y' : (\beta, y') \in \pi_2(v) \wedge y \leq y', \\ & (\gamma, \delta, z) \in \pi_3(v^q) \Rightarrow \exists z' : (\gamma, \delta, z') \in \pi_3(v) \wedge z \leq z'\}.\end{aligned}$$

Noch komplexer gestaltet sich die Berechnung von $\mathcal{U}_T^q$ für SC- und S-Treffer. Hier muß die Bedingung aus Satz 3.3.37 erfüllt sein.

Als nächstes wird auf die Berechnung der Mengen $\mathcal{U}_T^q$ eingegangen. Dabei werden Ergebnisse aus Abschnitt 3.3.4 angewendet, die im Zusammenhang mit dem Vokabular aufgestellt wurden. Ausgangspunkt ist die dort definierte partielle Ordnung $\sqsubseteq_T^{\mathcal{V}}$ auf einem Vokabular $\mathcal{V}$ bzgl. der Trefferart $T$. Mit dieser partiellen Ordnung ergibt sich

$$\mathcal{U}_T^q = \{v \in \mathcal{V} | v^q \sqsubseteq_T^{\mathcal{V}} v\}.$$

Basierend auf einer partiellen Ordnung lassen sich klassische Suchstrukturen und -verfahren zur Bestimmung von $\mathcal{U}_T^q$ verwenden. Diese werden z.B. in [10] und [28] detailliert besprochen.

Ein Beispiel der Suchstruktur ist in Abbildung 4.1 dargestellt.

Satz 3.3.22 gewährleistet, daß die Suchstruktur nur bzgl. $\sqsubseteq_S^{\mathcal{V}}$, d.h. für S-Treffer, aufgebaut werden muß. Diese läßt sich auch für die restlichen Trefferarten verwenden.

## 4.2 Optimierung

Im folgenden wird auf Möglichkeiten zur Optimierung der Suchverfahren eingegangen. Begonnen wird mit einer Darstellung von allgemeinen Grundlagen für eine effiziente Verarbeitung von invertierten Listen.

### 4.2.1 Grundlagen

Die in Abschnitt 3.4 beschriebenen Verfahren zur Trefferberechnung basieren alle auf dem gleichen Vorgehen. Für jedes Anfrageobjekt $q$ werden ein oder mehrere invertierte Listen sequentiell abgearbeitet. Jede invertierte Liste $L$ wird dabei vom Massenspeicher in den Hauptspeicher transferiert. Sobald ein Listeneintrag im Hauptspeicher vorhanden ist, kann dieser abgearbeitet werden. Es existieren also insgesamt drei voneinander abhängige Arbeitsprozesse:

1. Die Berechnung der Teilmenge $\mathcal{U}_T^q \subseteq \mathcal{V}$ für ein Anfrageobjekt $q$.
2. Der Transfer einer invertierten Liste $L(v)$ für eine Vokabel $v \in \mathcal{U}_T^q$ vom Massenspeicher in den Hauptspeicher.
3. Die Abarbeitung der transferierten Listeneinträge $E \in L(v)$.

Der erste Punkt wurde in Abschnitt 4.1 besprochen. Der effiziente Datentransfer der invertierten Listen ist systemabhängig, was z.B. in [42] besprochen wird. Auf den letzten Punkt wird im weiteren Verlauf dieses Abschnitts eingegangen.

Die drei Prozesse bauen wie folgt aufeinander auf: Wurde bei der Berechnung von $\mathcal{U}_T^q$ eine Vokabel $v$ ermittelt, die ein Element von $\mathcal{U}_T^q$ darstellt, so kann diese direkt an den zweiten Prozeß übergeben werden. Dieser muß nicht warten, bis $\mathcal{U}_T^q$ komplett berechnet ist, sondern kann bereits ermittelte Elemente $v \in \mathcal{U}_T^q$ direkt abarbeiten. Für solche Vokabeln $v$ wird im zweiten Prozeß die invertierte Liste $L(v)$ vom Massen- in den Hauptspeicher transferiert. Auch hier gilt, sobald ein Eintrag $E \in L(v)$ in den Hauptspeicher geladen wurde, kann dieser sofort vom dritten Prozeß bearbeitet werden.

Auch wenn die letzten beiden Prozesse auf den Ergebnissen des jeweils vorangegangen Prozesses aufbauen, kann jeder Prozeß getrennt arbeiten. Die bereits berechneten Ergebnisse werden gepuffert und stehen so dem nächsten Prozeß bei Bedarf zur Verfügung. Bei Mehrprozessorsystemen können die Prozesse auf verschiedene Prozessoren aufgeteilt werden. Durch eine weitgehend parallele Abarbeitung wird die Addition der Laufzeiten der drei Prozesse vermieden. Diese Vorgehensweise wird in [50] und [42] allgemein besprochen.

Obige Aufteilung in drei Prozesse wird auch bei den in [50] beschriebenen Verfahren zur Volltextsuche mittels invertierter Listen angewendet. Dabei werden die Einträge einer invertierten Liste sortiert nach ihrer Dokumentreferenz abgespeichert. Dies ermöglicht eine effiziente Schnittbildung von invertierten Listen, wie sie etwa in Gleichung (3.1) dargestellt ist.

Die in dieser Arbeit für die Trefferberechnung verwendeten Einträge in einer invertierten Liste nach Definition 3.3.23 sind komplexer als die in [50] aufgeführten. Auch hier lassen sich die Listeneinträge sortieren. Die ersten beiden Tupeleinträge eines Listeneintrags sind eine Objektreferenz $d(E)$ und eine Dokumentreferenz $D(E)$. Beide können als natürliche Zahlen aufgefaßt werden. Hierauf wurde schon in den Abschnitten 2.1.1 und 3.3.5 eingegangen. Eine Datenbasis $\mathcal{D}$ ist eine Folge $(D_1, \ldots, D_N)$. Eine Dokumentreferenz $D(E) = j$ ist eine Zahl aus dem Intervall $[1 : N]$ und steht stellvertretend für das Dokument $D_j$. Man bezeichnet $j \in [1 : N]$ auch als Dokumentnummer.

Analog wird mit Objektreferenzen $d(E) = i$ verfahren. Die Objektreferenzen oder Objektnummern eines Dokuments $D_j$ sind Elemente des Intervalls $[1 : |D_j|]$, die stellvertretend für die Objektmenge $\{o_1^j, \ldots, o_{|D_j|}^j\}$ steht. Für einen Listeneintrag $E$ ergibt sich $o_{d(E)}^{D(E)} \in D_{D(E)}$. Auch mit den Objektreferenzen aus den restlichen Tupeleinträgen in $E$ wird analog verfahren. Somit besteht ein Listeneintrag nur aus natürlichen Zahlen bzw. Mengen von natürlichen Zahlen. Diese lassen sich effizient verarbeiten und, wie der nächste Abschnitt zeigt, komprimieren. Mit dieser Konvention lassen sich nun die Einträge einer invertierten Liste sortieren:

1. primär nach aufsteigenden Dokumentreferenzen $D(E)$,
2. bei gleichen Dokumentnummern nach aufsteigenden Objektnummern $d(E)$.

Hiermit wird eine totale Ordnung auf der Menge aller Listeneinträge definiert. Man beachte, daß jedes Dokumentobjekt genau einen Listeneintrag erzeugt.

Bei den in Abschnitt 3.3 angegebenen Verfahren zur Trefferberechnung läßt sich die Sortierung der invertierten Listen bei der Umsetzung von Zeile 24 ausnutzen.

Ein Zwischenergebnis $F'$ ist bei den beiden $\psi$-Verfahren eine Menge von Abbildungen $\psi : Q \to D_j$ und bei den beiden $\phi$-Verfahren eine Abbildung $\phi : Q \to 2^{D_j}$. Nach obigen Konventionen zur Darstellung von Dokument- und Objektreferenzen lassen sich die $\psi$-Abbildungen als Funktionen $\psi : [1 : |Q|] \to [1 : |D_j|]$ auffassen. Hierbei muß man sich für jede Abbildung $\psi$ die Dokumentreferenz $j =: D(\psi)$ merken. Mit $\phi$-Abbildungen wird analog verfahren, worauf im weiteren Verlauf nicht näher eingegangen wird. Die Abbildungen in der Menge $F'$ werden aufsteigend nach Dokumentnummern $D(\psi)$ sortiert.

Im Verfahren werden die Einträge $E$ einer invertierten Liste $L(v)$ sequentiell abgearbeitet. Pro Eintrag $E$ werden aus $F'$ diejenigen Abbildungen $\psi$ benötigt, für die $D(\psi) = D(E)$ gilt. Da sowohl $E$ als auch $F'$ nach aufsteigenden Dokumentnummern sortiert sind, lassen sich die für $E$ benötigten Abbildungen $\psi \in F'$ effizient finden.

Die Anordnung der Listeneinträge wird nicht nur an dieser Stelle des Verfahrens ausgenutzt. Bei dem oben besprochenen Vorgehen ist eine Sortierung bei gleicher Dokumentnummer nach aufsteigenden Objektnummern nicht erforderlich. Für die Komprimierung von Listeneinträgen läßt sich diese Anordnung ausnutzen. Hierauf wird als nächstes eingegangen.

### 4.2.2 Komprimierung

Für viele Anwendungen benötigt ein aus invertierten Listen bestehender Index etwa genauso viel Speicherplatz wie die Datenbasis. Jedoch kann der für den Index benötigte Speicherplatz durch eine Komprimierung der invertierten Listen verringert werden. Dies wird in diesem Abschnitt für die bei der Trefferberechnung verwendeten invertierten Listen besprochen. Als erstes wird auf die Komprimierung der invertierten Listen eingegangen, die bei der in Abschnitt 3.2.1 beschriebenen Volltextsuche verwendet werden. Das dort angewendete Vorgehen wird dann auf die bei der Trefferberechnung aus Abschnitt 3.3 verwendeten invertierten Listen übertragen. Es wird von der im vorherigen Abschnitt besprochenen Darstellung der invertierten Listen durch natürliche Zahlen ausgegangen, die entsprechend sortiert sind.

Bei der Volltextsuche besteht eine invertierte Liste $L$ aus einer aufsteigend sortierten Folge von Dokumentnummern $j$, d.h. $L = \left(j_k\right)_{k=1}^{\ell}$. Für eine Datenbasis $\mathcal{D}$ mit $N$ Dokumenten werden $\lceil \log N \rceil$ Bits zur Darstellung einer Dokumentnummer benötigt.

Die im Durchschnitt benötigte Anzahl an Bits läßt sich reduzieren. Dafür werden die Differenzen zwischen aufeinanderfolgenden Dokumentnummern $j_{k-1}$ und $j_k$ in $L$ betrachtet. Setzt man $j_0 := 0$, so ergibt sich aus $L$ ein Folge $L'$ von Differenzen: $L' = \left(j_k - j_{k-1}\right)_{k=1}^{\ell}$. Hierbei tritt kein Informationsverlust auf. Die größtmögliche Differenz in $L'$ ist $N$.

Betrachtet man $L'$ als Folge von natürlichen Zahlen, deren Summe nach oben durch $N$ beschränkt ist, so läßt sich eine kompaktere Darstellungsform finden. Hierbei spielt die Wahrscheinlichkeitsverteilung der Differenzen eine wesentliche Rolle. Gerade bei langen Zahlenfolgen überwiegt die Anzahl an niedrigen Differenzen. Also kann i.d.R. von einer höheren Wahrscheinlichkeit für kleine Differenzwerte ausgegangen werden. Dies wird bei der Kodierung ausgenutzt, wo kleine Werte durch eine kürzere Bitfolge repräsentiert werden als große Werte.

Es existieren verschiedene Modelle zur Beschreibung der Wahrscheinlichkeitesverteilung der Differenzen. Die im Kontext von invertierten Listen wichtigsten, bzw. in der Praxis am häufigsten eingesetzten Modelle, werden z.B. in [50] beschrieben. Hierzu zählt u.a. die Golomb-Kodierung. So komprimierte invertierte Listen belegen nach [50] im Durchschnitt etwa 50 % des ursprünglichen Speicherplatzes in umkomprimiertem Zustand.

Die Listeneinträge $E$, die bei den in Abschnitt 3.4 beschriebenen Verfahren zur Trefferberechnung verwendet werden, bestehen nicht nur aus einer Dokumentnummer. Nach Definition 3.3.23 enthält $E$ zusätzlich eine Objektnummer $d(E)$ und Mengen von Objektnummern.

Für eine invertierte Liste $L = \left(E_k\right)_{k=1}^{\ell}$ werden die Dokumentnummern $D(E_k)$ wie oben besprochen kodiert. Zusätzlich wird für jede Teilfolge $F_j = \left(E'_k\right)_{k=1}^{\ell_j}$ von $L$ mit Listeneinträgen bzgl. des gleichen Dokuments $D_j$, d.h. $d(E'_k) = j \; \forall 1 \leq k \leq \ell_j$, die Objektnummern $d(E'_k)$ analog kodiert. Für $F_j$ ergibt sich die Folge von Objektnummern als $\left(d(E'_k)\right)_{k=1}^{\ell_j}$. Die im letzten Abschnitt besprochene Sortierung der Listeneinträge garantiert auch eine Sortierung von $F_j$ nach aufsteigenden Objektnummern. Somit können die Objektnummern $d(E')$ der Elemente von $F_j$ als Folge von Differenzen $\left(d(E'_k) - d(E'_{k-1}\right)_{k=1}^{\ell_j}$ angesehen und entsprechend kodiert werden. Dabei wird der Einfachheit halber $d(E'_0) := 0$ gesetzt.

Analog wird pro Listeneintrag $E$ mit jeder Menge $M \in \pi_3(E) \cup \pi_4(E) \cup \pi_5(E)$ von Objektnummern verfahren. Hierbei wird vorausgesetzt, daß jede Menge von Objektnummern aufsteigend sortiert ist und somit auch als sortierte Folge von Objektnummern angesehen werden kann.

Bei der Implementierung der Trefferberechnung wurde für die Komprimierung der invertierten Listen die Golomb-Kodierung verwendet.

Eine andere Variante der Komprimierung basiert auf der Annahme, daß eine Menge $M$ von Objektnummern mehrfach in Einträgen einer invertierten Liste vorkommt. Solche mehrfach auftretenden Mengen werden in einem Kodebuch gespeichert. In den Listeneinträgen wird dann nicht mehr die Menge $M$ gespeichert, sondern eine Referenz auf den entsprechenden Kodebucheintrag. Bei der Dekomprimierung werden die Referenzen durch den Eintrag im Kodebuch, d.h. die ursprüngliche Menge $M$, ersetzt.

Diese Art der Kodierung stammt aus der Textkomprimierung [50]. Statt der oben beschriebenen Mengen werden hier Buchstaben oder Wörter durch einen Kode versehen. Im Kodebuch werden diese Zuweisungen gespeichert und im Text werden die Buchstaben oder Wörter durch den entsprechenden Kode ersetzt. Dabei ist das Kodebuch entweder für alle Anwendungen fest gewählt oder wird in Abhängigkeit der Textdokumente zusätzlich zum kodierten Text abgespeichert.

Alternativ dazu existieren Verfahren zur impliziten Speicherung des Kodebuchs [50]. Bei der Textkomprimierung wird ein Wort $w$, das zum wiederholten Mal im Text vorkommt, durch einen Verweis auf das erste Auftreten von $w$ ersetzt. Hier dient der Text selbst als Kodebuch. Bei der Dekodierung werden die Verweise entsprechend ersetzt. Dieses Vorgehen ist nicht auf die Verwendung von Wörtern eingeschränkt sondern kann für beliebige Textausschnitte angewendet werden. Analog kann für die Trefferberechnung mit Objektmengen innerhalb der Listeneinträge verfahren werden.

Natürlich lassen sich die beiden vorgestellten Verfahren zu Komprimierung auch kombinieren. Zum einen werden Objektmengen durch eine Golomb-Kodierung komprimiert. Zusätzlich werden wiederholt vorkommende Mengen durch Verweise ersetzt. Dies wurde bei der Implementierung der Trefferberechnung eingesetzt. Auf die dort erzielten Ergebnisse wird in Abschnitt 4.3.2 eingegangen.

Dieser Abschnitt hat gezeigt, daß sich klassische Verfahren zur Komprimierung auch für die in dieser Arbeit verwendeten invertierten Listen einsetzen lassen. In Abschnitt 4.3.2 wird auf die durch die einzelnen Komprimierungsverfahren erzielte Reduktion des Speicherplatzes eingegangen.

### 4.2.3 Indexgranularität

In Abschnitt 2.1.1 wurde auf die Granularität der Datenbasis bei Textsammlungen und Musikarchiven eingegangen. Bei Büchern läßt sich z.B. das komplette Buch als ein Textdokument oder jede einzelne Seite als ein Dokument ansehen. Bei der ersten Aufteilung wird von einer gröberen Granularität als bei der zweiten gesprochen.

Entsprechend kann die Granularität des Indexes bzw. des Vokabulars geregelt werden. Implizit geschieht dies durch die Auswahl der Relationenmenge $\mathcal{R}$, durch die Beziehungen zwischen Objekten beschrieben werden (vgl. Abschnitt 2.2.1). Hier können für eine Anwendung verschiedene

Mengen von Relationen angegeben werden, wodurch die Detailliertheit der Objektbeziehungen festgelegt werden kann. Eine größere Anzahl an Relationen in $\mathcal{R}$ kann hier für eine feinere Unterscheidung verwendet werden.

Die Menge $\mathcal{R}$ von Relationen definiert die Kantenbeschriftungen der beschrifteten Graphen und somit das Vokabular, wobei $\mathcal{R}$ für eine Anwendung fest gewählt werden muß. Ein hierfür aufgebauter Index läßt sich auch für spezielle Vergröberungen $\mathcal{R}'$ von $\mathcal{R}$ einsetzen. Hiermit ist gemeint, daß Treffer bzgl. $\mathcal{R}'$ mit den bzgl. $\mathcal{R}$ definiertem Vokabular und invertierten Listen berechnet werden. Dies wird nun im Detail betrachtet.

Sei $f : A \to B$ eine Abbildung. Dann wird für $\alpha \subseteq A$ mit $f[\alpha]$ die Menge $\{f(a)|a \in \alpha\}$ und für $b \in B$ mit $f^{-1}[b]$ die Menge $\{a \in A|f(a) = b\}$ bezeichnet.

Seien $\mathcal{R}$ und $\mathcal{R}'$ zwei Mengen von Relationen mit den in Abschnitt 2.2 beschriebenen Eigenschaften. $\mathcal{R}'$ wird als **Vergröberung** von $\mathcal{R}$ bezeichnet, wenn $|\mathcal{R}| > |\mathcal{R}'|$ gilt und sich jede Relationen $R' \in \mathcal{R}'$ aus ein oder mehreren Relationen $R \in \mathcal{R}$ zusammensetzt, d.h. $R' = \cup_{R \in \mathcal{S}_{R'}} R$ für ein $\mathcal{S}_{R'} \subseteq \mathcal{R}$. Allgemein wird dies durch eine surjektive Abbildung $\Gamma_{\mathcal{R},\mathcal{R}'} : [1 : |\mathcal{R}|] \to [1 : |\mathcal{R}'|]$, kurz $\Gamma$, ausgedrückt, die auch als Vergröberungsabbildung bezeichnet wird. Mit dieser gilt für alle $\gamma' \in [1 : |\mathcal{R}'|] : R'_{\gamma'} = \cup_{\gamma \in \Gamma^{-1}[\gamma']} R_\gamma$. $\mathcal{R}$ wird auch als Verfeinerung von $\mathcal{R}'$ bezeichnet.

Betrachten wir hierzu ein Beispiel aus der in Abschnitt 2.2 beschriebenen Suche in Bauplänen.

**Beispiel 4.2.1.** Auf Seite 11 sind die neun Kreuzungsrelationen aufgelistet, die in der Architekturanwendung verwendet werden. Diese bilden die Menge $\mathcal{R}$. Ein Vergröberung von $\mathcal{R}$ ist z.B. die Menge $\mathcal{R}' = \{R'_+, R'_\lfloor, R'_\lceil, R'_\rceil, R'_\rfloor\}$ mit

1. $R'_+ = R_+ \cup R_\top \cup R_\dashv \cup R_\bot \cup R_\vdash$
2. $R'_\lfloor = R_\lfloor$
3. $R'_\lceil = R_\lceil$
4. $R'_\rceil = R_\rceil$
5. $R'_\rfloor = R_\rfloor$.

Hier setzt sich die +-Kreuzung aus $\mathcal{R}'$ aus fünf Relationen aus $\mathcal{R}$ zusammen. Verwendet man obige Durchnummerierung von $\mathcal{R}'$ und die auf Seite 11 angegebene für $\mathcal{R}$, so ergibt sich folgende Vergröberungsabbildung

$$\begin{array}{rcl} \Gamma : [1:9] & \to & [1:5] \\ 1 & \mapsto & 1 \\ 2 & \mapsto & 1 \\ 3 & \mapsto & 1 \\ 4 & \mapsto & 1 \\ 5 & \mapsto & 1 \\ 6 & \mapsto & 2 \\ 7 & \mapsto & 3 \\ 8 & \mapsto & 4 \\ 9 & \mapsto & 5. \end{array}$$

Hier sieht man, daß eine Relation aus $\mathcal{R}$ in mehreren Relationen in $\mathcal{R}'$ enthalten ist. In Abbildung 4.2 sind Beispiele für die oben beschriebene Vergröberung dargestellt. ∘

Bei der Trefferberechnung in Kapitel 3.3 wurde von beschrifteten Graphen ausgegangen. Diese dienten als Darstellung der Dokumente bzgl. einer Menge von Relationen $\mathcal{R}$. Hierbei bilden nach Definition 2.4.7 Kantenbeschriftungen $\alpha \in W_E$ eine nichtleere Teilmenge von $\mathcal{R}$. Invertierte Listen und Vokabular wurden bzgl. einer Datenbasis $\mathcal{D}$ und einer Menge von Relationen $\mathcal{R}$ konstruiert.

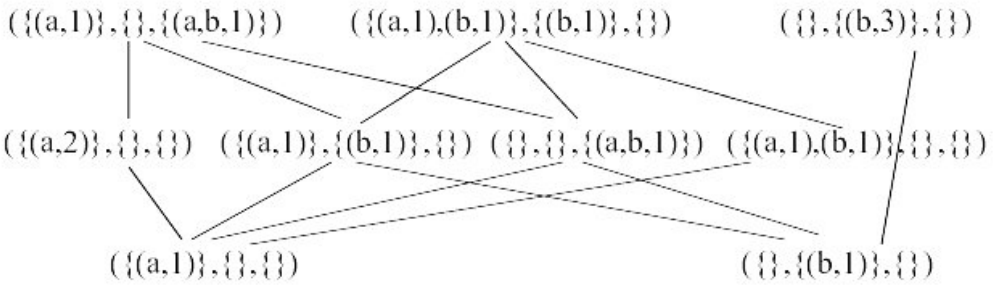



Abbildung 4.1: Für das Vokabular $\mathcal{V} = \big\{\big(\{(a,1)\},\emptyset,\emptyset\big),\ \big(\emptyset,\{(b,1)\},\emptyset\big),\ \big(\{(a,2)\},\emptyset,\emptyset\big),\ \big(\{(a,1)\},\{(b,1)\},\emptyset\big),\ \big(\emptyset,\emptyset,\{(a,b,1)\}\big),\ \big(\{(a,1),(b,1)\},\emptyset,\emptyset\big),\ \big(\{(a,1)\},\emptyset,\{(a,b,1)\}\big),\ \big(\{(a,1),(b,1)\},\{(b,1)\},\emptyset\big),\ \big(\emptyset,\{(b,3)\},\emptyset\big)\big\}$ ist die Anordnung mittels $\sqsubseteq_S^{\mathcal{V}}$ dargestellt. Eine Kante von $v$ nach $w$ symbolisiert $v \sqsubseteq_S^{\mathcal{V}} w$, wobei $v$ in der Darstellung unterhalb von $w$ angeordnet ist. Ist die Kante mit einem $I$ markiert, so gilt zusätzlich $v \sqsubseteq_I^{\mathcal{V}} w$. Die Vokabeln sind in Abhängigkeit der Gesamtanzahl der repräsentierten Kanten gruppiert. Die beiden untersten Vokabeln repräsentieren Knoten im Graphen $G_D$ mit einer ausgehenden bzw. eingehenden Kante. Die vier Vokabeln in der Mitte der Abbildung stehen für Knoten in $G_D$ mit Grad 2 und die drei obersten Vokabeln für Knoten mit Grad 3.

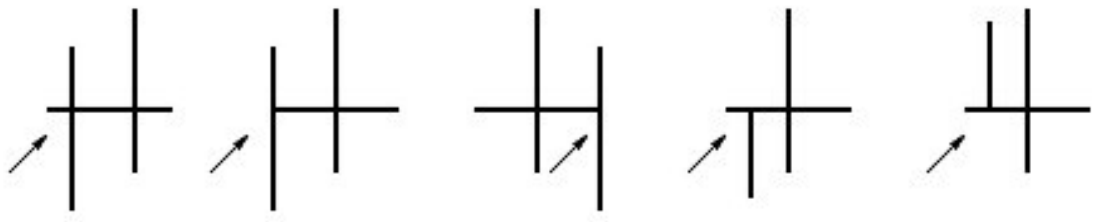


Abbildung 4.2: Fünf Dokumente zur Demonstration der in Beispiel 4.2.1 angegebenen Vergröberung $\mathcal{R}'$ der Menge $\mathcal{R}$ der Kreuzungsrelationen. Die mit einem Pfeil markierten Kreuzungen sind bzgl. der Vergröberung gleich. Wählt man ein beliebiges der dargestellten Dokumente als Anfrage $Q$ aus, so bilden alle fünf Dokumente einen IC-Treffer von $Q$.

Falls ein Index bzgl. $\mathcal{R}$ gegeben ist, so lassen sich damit auch Anfragen bzgl. einer Vergröberung $\mathcal{R}'$ von $\mathcal{R}$ beantworten. Dafür wird die oben eingeführte Vergröberung von Relationen auf Kantenbeschriftungen $\alpha$, also Mengen von Relationen, übertragen. Hierbei ist die Menge aller möglichen Kantenbeschriftungen $W_E = 2^{[1:|\mathcal{R}|]}$ bzw. $W_{E'} = 2^{[1:|\mathcal{R}'|]}$.

Eine Vergröberungsabbildung für Kantenbeschriftungen ist eine Abbildung von $W_E$ nach $W_{E'}$. Sie ergibt sich aus der Abbildung $\Gamma$, in dem jede Kantenbeschriftung $\alpha \in W_E$ auf $\Gamma[\alpha]$ abgebildet wird. So kann $\Gamma$ auch als Vergröberungsabbildung für Kantenbeschriftungen angesehen werden. Vergröberungsabbildungen auf Kantenbeschriftungen sind i.d.R. nicht injektiv.

Beispielsweise ergibt sich mit der in Beispiel 4.2.1 angegebenen Vergröberungsabbildung für die Kantenbeschriftung $\{1,2\}$: $\Gamma[\{1,2\}] = \{1\}$.

Eine komplette Vokabel läßt sich nun wie folgt vergröbern:

**Definition 4.2.2.** Sei $\mathcal{V}$ ein Vokabular, das bzgl. $\mathcal{R}$ konstruiert wurde und $\Gamma$ eine Vergröberungsabbildung für Kantenbeschriftungen bzgl. $\mathcal{R}$ und der Vergröberung $\mathcal{R}'$. Dann wird für eine Vokabel $v \in \mathcal{V}$ die vergröberte Vokabel $\Gamma(v)$ definiert durch $\Gamma(v) = (S_1, S_2, S_3)$ mit

$$\begin{aligned}
S_1 &= \big\{(\alpha', x') | 0 < x' = \sum_{(\alpha,x)\in\pi_1(v):\Gamma[\alpha]=\alpha'} x\big\} \\
S_2 &= \big\{(\beta', y') | 0 < y' = \sum_{(\beta,y)\in\pi_2(v):\Gamma[\beta]=\beta'} y\big\} \\
S_3 &= \big\{(\gamma', \delta', z') | 0 < z' = \sum_{(\gamma,\delta,z)\in\pi_3(v):\Gamma[\gamma]=\gamma'\wedge\Gamma[\delta]=\delta'} z\big\}.
\end{aligned}$$

Mittels dieser Vergröberung der Vokabeln bzgl. $\mathcal{R}$ lassen sich nun Anfragen in vergröberter Form beantworten.

Die in Abschnitt 3.4 angegebenen Verfahren zur Trefferberechnung basieren auf der Bestimmung von Vorkommen für alle Objekte $q$ einer Anfrage $Q$. Sei das Vokabular $\mathcal{V}$ und die invertierten Listen für eine Datenbasis $\mathcal{D}$ bzgl. $\mathcal{R}$ gegeben. Dann lassen sich mittels der Sätze 3.3.29 und 3.3.37 für die einzelnen Anfrageobjekte die Vorkommen aus den invertierten Listen berechnen, wenn die dort gestellten Bedingungen an Stelle der Vokabeln $\mathtt{v}(q, G_Q)$ und $v \in \mathcal{V}$ nun für die vergröberten Vokabeln $\Gamma(\mathtt{v}(q, G_Q))$ und $\Gamma(v)$ gelten.

Unterschiedliche Vokabeln $v, w \in \mathcal{V}$ können die gleiche Vergröberung $\Gamma(v) = \Gamma(w)$ besitzen. Daher müssen bei der Suche nach vergröberten Treffern i.d.R. mehr invertierte Listen abgearbeitet werden als im normalen Fall, wenn der Index direkt bzgl. $\mathcal{R}'$ aufgebaut wird.

Die hier vorgestelle Vergröberung ist mit der in Abschnitt 3.2.2 beschriebenen Fuzzy-Suche vergleichbar. Während bei der Fuzzy-Suche für ein Anfrageobjekt eine Menge von potentiellen Objekten angegeben wurde, kann durch die Vergröberung eine Relation für eine Menge von Relationen stehen.

### 4.2.4 Anfrageoptimierung

In diesem Abschnitt werden zwei Punkte besprochen, die die Laufzeit der in Abschnitt 3.4 angegebenen Verfahren beeinflussen. Dies betrifft zum einen die Reihenfolge der Anfrageobjekte, in der sie von den Verfahren abgearbeitet werden. Als zweites wird auf Kriterien zur Auswahl einer Teilmenge $Q'$ einer Anfrage $G_Q$ eingegangen. Hierdurch läßt sich bei zwei Verfahren die Menge der zu bearbeitenden Anfrageobjekte einschränken.

Bei den in Abschnitt 3.4 angegebenen vier Verfahren zur Trefferberechnung wurde eine Anfrage $G_Q$ mit $Q = \{q_1, \ldots, q_m\}$ abgearbeitet, indem bzgl. $q_1$ die Initialisierung durchgeführt wurde und in einer Schleife die restlichen Anfrageobjekte $q_2, \ldots, q_m$ durchlaufen wurden. Natürlich läßt sich $Q$ in einer anderen Reihenfolge abarbeiten, ohne daß dies die endgültige Treffermenge verändert.

Die Reihenfolge der Abarbeitung von $Q$ hat teilweise erheblichen Einfluß auf die Laufzeit der Trefferberechnung. Dies wird durch die in Abbildung 4.3 dargestellte Anfrage $G_Q$ motiviert. Zunächst wird davon ausgegangen, daß die Verfahren alle Anfrageobjekte von $Q$ durchlaufen müssen.

Auf eine geeignete Einschränkung von $Q$ zu $Q'$ für die zwei Verfahren `A2` und `A4` wird in diesem Abschnitt an späterer Stelle eingegangen.

Untersuchen wir nun die Berechnung aller I-Treffer für die in Abbildung 4.3 dargestellte Anfrage $G_Q$. Werden als erstes die beiden Anfrageobjekte $q_1$ und $q_2$ abgearbeitet, so entsteht bei der Kombination der lokalen Trefferabbildungen eine unnötig große Anzahl an Trefferabbildungen (Zeile 27 der Verfahren). Diese werden alle in der Menge $F$ zwischengespeichert. Bzgl. $G_Q$ gilt für jedes Dokument $G_D$: Jede lokale Trefferabbildung bzgl. eines Vorkommens $d_{q_1}$ in $G_D$ von $q_1$ kann mit jeder lokalen Trefferabbildung eines Vorkommens $d_{q_2}$ in $G_D$ von $q_2$ kombiniert werden. Nur wenige dieser Kombinationen werden durch die Bedingungen in Zeile 26 ausgeschlossen.

In dem in Abbildung 4.3 dargestellten Dokument $G_D$ besitzt $q_1$ sechs I-Vorkommen, ebenso $q_2$. Insgesamt ergeben sich 36 Möglichkeiten zur Kombination der lokalen Trefferabbildungen von $q_1$ und $q_2$, von denen nur vier durch die erste Bedingung in Zeile 26 ausgeschlossen werden. Nach der Abarbeitung von $q_1$ und $q_2$ sind in der Menge $F$ 32 Abbildungen gespeichert (Zeile 27).

Grund für die hohe Anzahl an Kombinationsmöglichkeiten ist die Eigenschaft, daß die Subgraphen $G_Q^{q_1}$ und $G_Q^{q_2}$ von $G_Q$ keine gemeinsamen Knoten besitzen. In einem solchen Fall kann die zweite Bedingung in Zeile 26 nicht greifen, wo gerade die Knoten aus $Q^{q_1} \cap Q^{q_2}$ untersucht werden.

Werden dagegen in $G_Q$ die benachbarten Knoten $q_1$ und $q_4$ als erstes abgearbeitet, ergeben sich insgesamt nur 5 Kombinationen von lokalen Trefferabbildungen, die die Bedingungen aus Zeile 26 erfüllen. Hier werden nur 5 Abbildungen in $F$ zwischengespeichert, was im Vergleich zu 32 Abbildungen im vorherigen Beispiel eine erhebliche Verbesserung ist.

Eine aus praktischer Sicht notwendige Bedingung an die Abarbeitungsreihenfolge der Anfrageobjekte in der Schleife in Zeile 18 ist die folgende: Ein Objekt $q_i \in Q$ wird erst dann abgearbeitet, wenn der Graph $G_Q^{q_i}$ mindestens einen Knoten mit dem Graphen $G_Q^p$ eines bereits abgearbeiteten Objektes $q \in Q$ gemeinsam hat. Dies gilt natürlich nicht für das erste Objekt bei der Initialisierung. Da jeder Anfragegraph $G_Q$ nach Annahme 2.3.1 zusammenhängend ist, existiert für jedes $q \in Q$ ein $p \in Q : Q^q \cap Q^p \neq \emptyset$. Weiterhin ist jeder Knoten $q$ mit jedem anderen Knoten $p$ durch einen Pfad in $G_Q$ verbunden, bei dem die Kantenrichtungen vernachlässigt werden, vgl. Annahme 2.4.9.

Zusätzlich zu der bisher besprochenen Bedingung kann durch eine geeignete Wahl der Abarbeitungsreihenfolge der Anfrageobjekte die Anzahl der in $F$ zwischengespeicherten Abbildungen möglichst gering gehalten werden. Betrachten wir hierzu primär die Verfahren `A2` und `A4`, die auf $\phi$-Abbildungen basieren. Bei der Initialisierung bzgl. eines Anfrageobjektes $q_1$ ist die Anzahl der Abbildungen in $F$ gleich der Anzahl der Vorkommen von $q_1$ in allen Dokumenten der Datenbasis. Diese Anzahl entspricht wiederum der Anzahl der Listeneinträge der invertierten Listen $L(v)$ mit $v \in \mathcal{V} : v \sqsubseteq_T^{\mathcal{V}} \mathtt{v}(q_1, G_Q)$. Hierbei ist $\sqsubseteq_T^{\mathcal{V}}$ die partielle Ordnung auf den Vokabeln nach Definition 3.3.20 für eine Trefferart $T$.

In diesem Zusammenhang lassen sich die Kosten eines Anfrageobjektes $q \in Q$ bzgl. eines

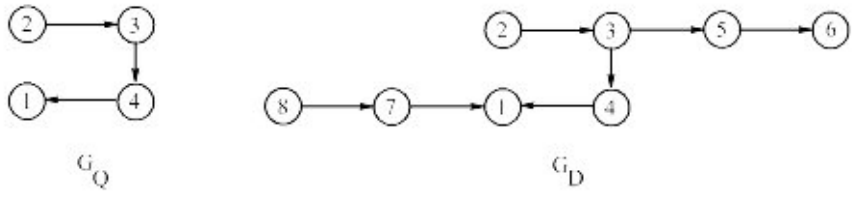



Abbildung 4.3: Eine Anfrage $G_Q$ mit vier Knoten und ein Dokument $G_D$ mit acht Knoten. Sie dienen zur Motivation, daß die Abarbeitungsreihenfolge der Anfrageobjekte die Rechenzeit beeinflußt. Die Ziffer in einem Knoten ist dessen Knotennummer.

Indexes aus invertierten Listen mit dem Vokabular $\mathcal{V}$ durch

$$Kosten_T(q) = \sum_{v \in \mathcal{V}: \mathrm{v}(q_1, G_Q) \sqsubseteq_T^{\mathcal{V}} v} |L(v)|$$

definieren.

Bei der Abarbeitung einer Anfrage $G_Q$ sollte mit einem Objekt $q \in Q$ mit minimalen Kosten begonnen werden, d.h. $q \in Q : \forall p \in Q : Kosten_T(q) \leq Kosten_T(p)$. Hierdurch wird gewährleistet, daß die Anzahl $|F|$ der lokalen Trefferabbildungen bei der Initialisierung minimal ist.

Sei $\hat{Q}$ die Menge der Anfrageobjekte, die noch nicht abgearbeitet worden sind. Nun soll motiviert werden, welches Objekt $q$ aus $\hat{Q}$ als nächstes bearbeitet werden soll.

Für eine grobe Abschätzung der Anzahl der Abbildungen, die sich bei dem Kombinationsschritt in Zeile 27 für den $i$-ten Schleifendurchlauf ergeben, wird folgende Heuristik verwendet: Die Anzahl an neuen Abbildungen ist dann am kleinsten, wenn die Kosten von $q$ am geringsten sind. Wie bereits oben besprochen, muß zusätzlich $Q^q \cap Q^p \neq \emptyset$ für ein bereits abgearbeitetes Objekt $p$ gelten. Die verwendete Heuristik wurde durch praktische Tests bestätigt. Basierend auf der Heuristik wird das als nächstes zu bearbeitende Anfrageobjekt $q \in \hat{Q}$ so gewählt, daß

1. $\forall p \in \hat{Q} : Kosten_T(q) \leq Kosten_T(p)$ und
2. $\exists p \in Q \setminus \hat{Q} : Q^q \cap Q^p \neq \emptyset$.

Hierdurch läßt sich eine Reihenfolge bestimmen, in der die Anfrageobjekte in den Verfahren abgearbeitet werden und dabei die Anzahl an zwischengespeicherten Abbildungen möglichst gering ist.

Praktische Untersuchungen haben ergeben, daß für eine so bestimmte Reihenfolge der Abarbeitung der Anfrageobjekte in den meisten Fällen die geringsten Laufzeiten erzielt werden. Hierauf wird in Abschnitt 4.3.4 eingegangen.

Bei der Bestimmung einer Teilmenge $Q'$ von $Q$ für die Algorithmen `A2` und `A4` werden ähnliche Überlegungen wie bei der zuletzt besprochenen Bestimmung einer Abarbeitungsreihenfolge der Anfrageobjekte angewendet. Auch hier sollte $Q'$ so gewählt werden, daß sowohl möglichst wenig Listeneinträge geladen und abgearbeitet werden müssen als auch die Anzahl an zwischengespeicherten Abbildungen gering ist. Hierfür lassen sich die Kosten von $Q'$ in Abhängigkeit der Trefferart $T$ definieren durch

$$Kosten_T(Q') = \sum_{q \in Q'} Kosten_T(q).$$

Natürlich muß $Q'$ zusätzlich die in Satz 3.3.9 gestellte Bedingung 3.9 erfüllen. Die Basis $Q'$ von $G_Q$ wird nun so gewählt, daß die Kosten von $Q'$ minimal sind, d.h. für alle Basen $Q''$ von $G_Q$ gilt: $Kosten_T(Q') \leq Kosten_T(Q'')$.

Unberücksichtigt bleibt bei der oben gewählten Kostenabschätzung für $Q'$ die Anzahl an invertierten Listen, die geladen werden müssen. Diese bestimmt die Anzahl an wahlfreien Zugriffen auf das Sekundärmedium und hat somit Einfluß auf die Laufzeit. Die Kosten für einen wahlfreien Zugriff kann durch einen systemabhängigen konstanten Faktor zu obiger Kostenabschätzung hinzugefügt werden.

**Fazit** Die Auswahl der Basis $Q'$ von $G_Q$ und die Reihenfolge, in der die Anfrageobjekte abgearbeitet werden, beeinflussen die Laufzeit der angegebenen Verfahren zur Trefferberechnung. Durch das besprochene Auswahlverfahren wird die Abarbeitungsreihenfolge der Anfrageobjekte festgelegt, durch die die Anzahl an zwischenzuspeichernden Abbildungen möglichst gering gehalten wird. Zusätzlich kann durch die Auswahl der Basis $Q'$ die Anzahl an zu betrachtenden invertierten Listen und Listeneinträgen für eine Anfrage $G_Q$ minimiert werden. In Abschnitt 4.3.4 wird durch praktische Ergebnisse gezeigt, daß sowohl die Reihenfolge der Abarbeitung als auch die Auswahl von $Q'$ i.d.R. erheblichen Einfluß auf die Laufzeit der Trefferberechnung haben.

### 4.2.5 Trefferarten IC und SC

In Abschnitt 3.4 wurden Verfahren zur Trefferberechnung vorgestellt, die sich für alle vier Trefferarten eignen. Sie verwenden die in Abschnitt 3.3.4 und 3.3.5 definierte Indexstruktur aus invertierten Listen. Diese Indexstruktur wurde primär für die Trefferarten I und S entwickelt. In diesem Abschnitt wird beschrieben, wie sich diese Indexstruktur durch einfache Erweiterungen an eine spezielle Eigenschaft der Trefferarten IC und SC anpassen läßt, um somit die IC- und SC-Treffer effizienter berechnen zu können.

Von den vier in Abschnitt 2.3 definierten Trefferarten zeichnen sich die Trefferarten IC und SC dadurch aus, daß die Kardinalitäten der Anfrage und der Trefferdokumente gleich sind. Für die anderen beiden Trefferarten I und S muß dies nicht zutreffen.

Die Informationen über die Dokumentkardinalitäten wurden bisher nicht in die Indexstruktur integriert. Statt dessen wurde bei den Verfahren zur Berechnung von IC- und SC-Treffern im Initialisierungsschritt überprüft, ob für die betrachteten Einträge einer invertierten Listen die Dokumentkardinalität mit der Anfragekardinalität übereinstimmt. Hierbei wird i.d.R. ein großer Teil der betrachteten Listeneinträge von der weiteren Bearbeitung ausgeschlossen. Trotzdem werden sie zur Überprüfung in den Hauptspeicher transferiert. Durch eine feinere Indexgranularität läßt sich für die Trefferarten IC und SC dieser überflüssige Datentransfer verhindern. Dafür wird die Kardinalität von Dokumenten mit in das Vokabular einbezogen und die bisherigen invertierten Listen in Abhängigkeit der Dokumentkardinalitäten unterteilt. Dann gilt für alle Einträge einer invertierten Liste, daß die Kardinalität der in den Einträgen referenzierten Dokumente gleich ist.

Für ein Objekt $d \in D$ ist nach Definition 3.3.12 die Vokabel $\mathtt{v}(d, G_D)$ ein Tripel, in dem Informationen über Kantenbeschriftungen gespeichert sind. Die Vokabel wird nun um die Dokumentkardinalität $|D|$ zu einem Quadrupel erweitert. Diese erweiterte Vokabel von $d$ wird mit $\mathtt{v}^+(d, G_D)$ bezeichnet und ist ein Element aus

$$2^{W_E \times \mathbb{N}} \times 2^{W_E \times \mathbb{N}} \times 2^{W_E \times W_E \times \mathbb{N}} \times \mathbb{N},$$

wobei $W_E$ die Menge aller möglichen Kantenbeschriftungen ist. Ein Index bzgl. der erweiterten Vokabeln wird als erweiterter Index und das Vokabular entsprechend als erweitertes Vokabular bezeichnet.

Da nun alle Listeneinträge zusätzlich an Hand der Dokumentkardinalität klassifiziert werden, impliziert das erweiterte Vokabular eine größere Anzahl an invertierten Listen und die oben besprochene feinere Unterteilung der Einträge in invertierte Listen.

Man beachte, daß sich die in Abschnitt 3.4 definierten Verfahren zur Trefferberechnung direkt mit dem hier beschriebenen erweiterten Index einsetzen lassen.

Ein großer Vorteil einer erweiterten Indexstruktur für die Trefferarten IC und SC ist, daß bei der Trefferberechnung nur noch Listeneinträge in den Hauptspeicher transferiert werden, deren Dokumentkardinalität mit der Kardinalität der Anfrage übereinstimmt. Dies führt für diese Trefferarten zu erheblichen Laufzeitverbesserungen, was in Abschnitt 4.3.8 praktisch untersucht wird.

Für die restlichen beiden Trefferarten I und S besitzt ein erweiterter Index im Vergleich zum ursprünglichen Index für die Trefferberechnung allerdings einen Nachteil. Hier muß eine größere Anzahl von invertierten Listen vereinigt werden. Dies führt zu einer größeren Anzahl an wahlfreien Zugriffen auf dem Massenspeicher. Hierauf wird in Abschnitt 4.3.8 weiter eingegangen.

## 4.3 Auswertung

In diesem Abschnitt werden praktische Ergebnisse von Implementierungen der Trefferverfahren vorgestellt. Teilweise werden die praktischen und theoretischen Ergebnisse miteinander verglichen.

### 4.3.1 Testumgebung

Die praktische Auswertung wurden auf einem PC mit einem Intel Celeron 600 Prozessor und 64 MB Hauptspeicher unter dem Betriebssystem MS Windows 98 SE ermittelt. Als Massenspeicher wurde

eine gängige IBM-Festplatte eingesetzt.

Die Verfahren wurden in der Programmiersprache C++ mit dem GNU-Compiler, Version $2.95.3 - 5$, entwickelt.

**Datenbasis** Die Dokumente der Datenbasis wurden zufällig generiert. Hierbei entspricht einem Dokumente ein zufällig generierter zusammenhängender beschrifteter Graph, wobei jedem Objekt ein Knoten entspricht. Im Durchschnitt besitzt jeder Knoten zwei Nachbarn. Als Kantenbeschriftungen wurden zufällig nichtleere Teilmengen aus der Menge $\{1, \ldots, 9\}$ gewählt, d.h. die Menge aller möglichen Kantenbeschriftungen ist $2^{\{1,\ldots,9\}}$. Dabei wurde die Wahrscheinlichkeit, daß eine bestimmte Teilmenge $X$ aus $\{1, \ldots, 9\}$ gewählt wird, in Abhängigkeit von der Kardinalität von $X$ definiert. Dies geschah antiproportional zur Kardinalität von $X$. Kleinere Teilmengen kommen also häufiger vor als große Teilmengen. Dies entspricht den Beobachtungen aus Abschnitt 2.2, wo auf Eigenschaften der Architekturanwendung eingegangen wurde. Weitere spezielle Eigenschaften der Architekturanwendung werden nicht ausgenutzt. Diese werden getrennt in Abschnitt 4.3.11 besprochen.

**Größe der Datenbasis** Für die nachfolgenden Untersuchungen wurde eine Datenbasis aus 200.000 Dokumenten mit durchschnittlich 200 Objekten pro Dokument verwendet. Insgesamt enthält die Datenbasis ca. 40 Millionen Objekte.

Als nächstes wird der theoretisch benötigte Speicherplatz der Datenbasis berechnet und mit dem in der Praxis belegten verglichen.

Jeder beschriftete Graph der Datenbasis wird als Adjazenzliste abgespeichert. Jede so abgespeicherte Kante bestehend aus Startknoten, Zielknoten und Kantenbeschriftung benötigt 2 Byte pro Knotennummer und 2 Byte für die Kantenbeschriftung. Insgesamt belegt eine Kante in einer Adjazenzliste 6 Byte Speicherplatz.

Da die Anzahl der Kanten gleich der Anzahl der Knoten ist, ergibt sich für die Datenbasis eine Gesamtgröße von $40.000.000 \cdot 6$ Byte $= 240.000.000$ Byte, was ungefähr 240 MB entspricht. In der Praxis belegte die Datenbasis insgesamt 262 MB auf der Festplatte. Hier führt u.a. die minimale Blockgröße des Dateisystems zu einer Abweichung vom theoretisch berechneten Speicherplatz [13].

**Indexgröße** Betrachten wir nun den Speicherplatz, den die invertierten Listen benötigen.

Jedes Objekt $d$ eines Dokumentes $G_D$ der Datenbasis erzeugt einen Listeneintrag. Dieser enthält die Objektnummer von $d$, die Dokumentnummer von $G_D$ und die Objektnummern der Nachbarn von $d$. Da im Durchschnitt jeder Knoten zwei Nachbarn besitzt, ergibt sich ein durchschnittlicher Speicherplatz von $4$ Byte $+ 3 \cdot 2$ Byte $= 10$ Byte pro Listeneintrag. Hierbei belegt eine Dokumentreferenz 4 Byte.

Insgesamt ergibt sich eine theoretische Indexgröße von $40.000.000 \cdot 10$ Byte $= 400.000.000$ Byte, was ungefähr 400 MB entspricht. In der Praxis belegt der Index 423 MB Platz auf der Festplatte. Der unterschiedliche Speicherplatzbedarf in Theorie und Praxis beruht im wesentlichen auf den bereits oben erwähnten systemabhängigen Gründen.

**Vokabular** Die Größe des Vokabulars ist stark abhängig von der jeweiligen Datenbasis. Bei praktischen Tests mit verschiedenen Datenbasen der oben beschriebenen Größe bestand das Vokabular aus durchschnittlich 1.2 Millionen unterschiedlichen Vokabeln. Im Durchschnitt belegte das Vokabular 19 MB Speicher, worin die in Abschnitt 4.1 besprochene Suchstruktur enthalten ist.

**Fazit** Bei der Volltextsuche entspricht die Indexgröße ungefähr der Größe der Datenbasis [50]. Der für die Trefferberechnung verwendete Index benötigt fast den doppelten Speicherplatz wie die Datenbasis. Im Vergleich zur Volltextsuche werden hier auch viel detailliertere Informationen in den Listeneinträgen abgespeichert, was zu einem größeren Platzbedarf führt.

Das Vokabular belegt mit knapp 20 MB etwa ein Drittel des Hauptspeichers. Die praktischen Berechnungen hat dies allerdings nicht behindert, da für die Trefferberechnung zu jedem Zeitpunkt

genügend Hauptspeicher zur Verfügung stand. Hierbei ist allerdings zu beachten, daß unter dem gewählten Betriebssystem eine temporäre Auslagerung des internen Speichers auf ein externes Speichermedium möglich ist [31].

Die Größe der Datenbasis wurde für das Testsystem in etwa so gewählt, wie es realistischen Anwendungen in der Praxis auf anderen Systemen entspricht: Der Speicherplatz der Datenbasis ist um ein Vielfaches größer als der vorhandene Hauptspeicher. Eine interne Suche allein im Hauptspeicher ist also ausgeschlossen.

### 4.3.2 Indexkomprimierung

In Abschnitt 4.3.1 wurde auf die Indexgröße im unkomprimiertem Zustand eingegangen. Der benötigte Speicherplatz läßt sich durch die in Abschnitt 4.2.2 beschriebenen Komprimierungsverfahren verringern. Insgesamt wurden drei Verfahren zur Komprimierung der invertierten Listen getestet:

1. die Kodierung der Differenzen zwischen Objektnummern bzw. Dokumentnummern durch eine Golomb-Kodierung,
2. die Verwendung von Verweisen bei wiederholt vorkommenden Objektmengen,
3. ein hybrides Verfahren, das die beiden vorherigen Ansätze kombiniert.

Die Ergebnisse sind in folgender Tabelle zusammengefaßt:

| Komprimierungsverfahren | Indexgröße in MB |
|---|---|
| kein | 423 |
| Golomb | 192 |
| Verweise | 398 |
| hybrid | 177 |

Auffallend ist die deutliche Platzeinsparung durch die Golomb-Kodierung. Statt global 4 Byte für eine Dokumentreferenz zu verwenden ist hier die Darstellungslänge einer Zahl abhängig von dem tatsächlichen Zahlenwert. Da durch die Verwendung von Differenzen die Zahlenwerte i.d.R. gering gehalten werden, greift das Verfahren besonders gut. Dies stimmt mit den in [50] beschriebenen Testergebnissen überein.

Bei den gewählten Byte-Werten für die unkomprimierte Zahlendarstellung wird nicht der volle Zahlenbereich für Dokument- bzw. Objektreferenzen ausgenutzt. Z.B. besitzt ein Dokument im Durchschnitt 200 Objekte, d.h. ein großer Bereich der mit 2 Byte darstellbaren Zahlen bleibt ungenutzt. Dies begünstigt das gute Abschneiden der Golomb-Kodierung im Vergleich zu dem zweiten Verfahren.

Durch die Verwendung von Verweisen läßt sich die Indexgröße nur minimal verringern. Wendet man dieses Vorgehen zusätzlich zur Golomb-Kodierung an, so läßt sich durch das hybride Verfahren der beste Komprimierungswert erzielen.

Da durch das hybride Komprimierungsverfahren die besten Werte erzielt werden, beziehen sich die weiteren Ergebnisse immer auf so komprimierte invertierte Listen.

### 4.3.3 Transferzeiten

Für den Vergleich der Trefferberechnungsverfahren aus Abschnitt 3.4 mit den bekannten Verfahren aus Abschnitt 3.1.2, die ohne Index direkt auf der Datenbasis arbeiten, sind die Transferzeit aller invertierten Listen bzw. der kompletten Datenbasis von Interesse.

Die Transferzeit der Datenbasis mit einer Größe von 240 MB von der Festplatte in den Hauptspeicher beträgt auf dem Testsystem 130 Sekunden. Für den Index wurde die Transferzeit von allen invertierten Listen mit einer Gesamtgröße von 177 MB mit 90 Sekunden gemessen. Diese Werte sind extrem systemabhängig und nur als ungefähre Richtwerte anzusehen. Bei den Messungen konnten beeinflussende Faktoren wie Festplattencache und sonstige systemabhängige Optimierungen der Festplattenoperationen nie ganz ausgeschlossen werden.

Trotzdem liefern sie eine untere Schranke für die Verfahren, die direkt auf der Datenbasis arbeiten, da hier immer alle Daten in den Hauptspeicher transferiert werden müssen. Für den Index liefert die Transferzeit von allen invertierten Listen eine obere Schranke für die gesamte Übertragungszeit, die im schlechtesten Fall auftreten kann. Hier müssen für eine Anfrage alle invertierten Listen abgearbeitet werden.

In Abschnitt 3.1.1 wurde das Ausgangsszenario dieser Arbeit beschrieben. Dabei wurden Werte für den Datentransfer vom Massen- in den Hauptspeicher angegeben. Verwendete man diese Werte zur Berechnung der Transferzeiten der kompletten Datenbasis bzw. aller invertierten Listen, so unterscheiden sich diese stark von den auf dem Testsystem gemessenen Werten.

Für die Datenbasis ergibt sich mit den Werten aus dem Ausgangsszenario eine Transferzeit von 2120 Sekunden, die sich aus den Werten für den wahlfreien Zugriff ($200.000 \cdot 10 \cdot 10^{-3}$ Sekunden) und der eigentlichen Transferzeit der Daten ($240.000.000 \cdot 0,5 \cdot 10^{-6}$ Sekunden) ergibt.

Eine mögliche Erklärung für die starke Diskrepanz der Werte ist, daß die wahlfreien Zugriffe auf dem Testsystem nicht ins Gewicht fallen. Vergleicht man die Dauern für den reinen Datentransfer, so stimmen diese fast überein. Hier unterscheidet sich möglicherweise das Testsystem von großen Datenbanksystemen, bei denen wahlfreie Zugriffe stärker ins Gewicht fallen.

### 4.3.4 Einfluß der Anfrageoptimierung

In Abschnitt 4.2.4 wurde auf eine Optimierung einer Anfrage $G_Q$ bzgl. einer Datenbasis eingegangen. Die Optimierung bestand aus 2 Teilen:

1. Die Reihenfolge wird festgelegt, in der die Objekte der Anfrage abgearbeitet werden.
2. Es wird eine Basis $Q'$ von $G_Q$ ausgewählt, die für zwei Verfahren verwendet wird.

Die Berechnungszeit zur Bestimmung der Reihenfolge und der Basis $Q'$ ist selbst für große Anfragen mit ca. 200 Objekten kaum meßbar und kann somit vernachlässigt werden.

Die erzielten Laufzeiten sind stark von der Anfrage und der Datenbasis abhängig. Exemplarisch wird die in Abbildung 4.4 dargestellte Anfrage $G_Q$ bzgl. der oben beschriebenen Testdatenbasis auf zwei Arten untersucht. Einmal die komplette Anfrage $G_Q$ und einmal nur eine Anfrage, die einen Teil von $G_Q$ dargestellt und aus den drei Objekten $b, c$ und $d$ sowie den entsprechenden Kanten besteht.

Für die komplette Anfrage $G_Q$ wurden in Abhängigkeit der Abarbeitungsreihenfolge die folgenden Werte für die Berechnung aller S-Treffer mit dem Verfahren `A3` gemessen:

| S-Trefferberechnung | |
|---|---|
| Reihenfolge | Laufzeit |
| $a, b, c, d$ | 18 Sek. |
| $b, a, d, c$ | 9 Sek. |
| $b, c, a, d$ | 7 Sek. |
| $b, c, d, a$ | 6 Sek. |
| $d, b, c, a$ | 6 Sek. |
| $a, d, c, b$ | über 5 Min. |

Die in der letzten Zeile der Tabelle aufgeführte Reihenfolge entspricht nicht der in Abschnitt 4.2.4 besprochenen Bedingung: $a$ und $d$ sind nicht benachbart. Hierdurch ergibt sich die hohe Beantwortungszeit.

Ansonsten wird deutlich, wie sehr die Reihenfolge die Antwortzeit bestimmen kann. Ein Grund für die relativ lange Laufzeit der ersten Reihenfolge in der Tabelle ist die hohe Anzahl an Listeneinträgen, die für das Objekt $a$ abgearbeitet werden müssen. $a$ erzielt hier die höchsten, in Abschnitt 4.2.4 definierten Kosten. Jeder Listeneinträge bzgl. $a$ erzeugt eine oder mehrere Trefferabbildungen, die zwischengespeichert werden. Alle anderen Anfrageobjekte verursachen deutlich geringere Kosten. Hier müssen deutlich weniger Trefferabbildungen zwischengespeichert werden. Je später $a$ in der Reihenfolge abgearbeitet wird, um so geringer ist die Anzahl an Abbildungen, die sich

nach der Kombination von Trefferabbildungen im Verfahren in Zeile 27 zwischengespeichert werden. Wie die gemessenen Laufzeiten verdeutlichen, hat dies teilweise erheblichen Einfluß auf die Antwortzeit.

Betrachtet man allerdings nur einen Teil der Anfrage $G_Q$ bestehend aus den drei Objekten $b, c, d$ und den entsprechenden Kanten, so hat hier die Abarbeitungsreihenfolge kaum Einfluß auf die Laufzeit. Alle drei Objekte besitzen fast die gleichen Kosten. Dies ergibt sich in der Tabelle aus den Zeilen, wo in der angegebenen Reihenfolge das Objekt $a$ als letztes abgearbeitet wird.

Die in Abschnitt 4.2.4 besprochene Anfrageoptimierung liefert für $G_Q$ die Abarbeitungsreihenfolge $d, b, c, a$, die mit die schnellste Beantwortungszeit erzielt. Für die eingeschränkte Anfrage gilt dies anlog für die berechnete Reihenfolge $d, b, c$.

Das bisher besprochene Beispiel dient auch zur Demonstration der Laufzeitabhängigkeit von der gewählten Basis $Q'$ von $G_Q$. Hier ergaben sich die folgenden Laufzeiten in Abhängigkeit von $Q'$:

| S-Trefferberechnung | |
|---|---|
| $Q'$ | Laufzeit |
| $\{a, c\}$ | 17 Sek. |
| $\{a, d\}$ | 17 Sek. |
| $\{b, d\}$ | 3 Sek. |
| $\{b\}$ | 2 Sek. |

Einen deutlichen Laufzeitgewinn erzielt man, wenn das Objekt $a$ nicht in $Q'$ enthalten ist. Für $Q' = \{b\}$ lassen sich die besten Ergebnisse erzielen. Dies bestätigt die in Abschnitt 4.2.4 besprochene Anfrageoptimierung, nach der $Q' = \{b\}$ gewählt wird.

**Fazit** Das besprochene Beispiel demonstriert, daß die in Abschnitt 4.2.4 besprochene Anfrageoptimierung teilweise erhebliche Laufzeitvorteile bringt. Dies gilt sowohl für die Bestimmung einer Abarbeitungsreihenfolge als auch für die Auswahl der Basis $Q'$ von $G_Q$ für die Verfahren `A2` und `A4`.

Bei allen im weiteren Verlauf dieses Kapitels besprochenen Beispielen für die vier Berechnungsverfahren wird die Anfrageoptimierung verwendet.

### 4.3.5 Laufzeiten verschiedener Trefferberechnungen

Als nächstes werden die Laufzeiten der Trefferberechnungsverfahren für jede der vier Trefferarten untersucht. Dabei kommen die vier Berechnungsverfahren so zum Einsatz, wie sie in Abschnitt 3.4 beschrieben wurden. Im Gegensatz zu den Varianten der Trefferberechnungsverfahren, die im nächsten Abschnitt untersucht werden, werden hier die Bedingungen zur Kombination von Trefferabbildungen in Zeile 26 des Verfahrens bei jedem Durchlauf durchgeführt.

Bei den Trefferarten IC und SC müssen die Kardinalitäten der Anfrage und der Trefferdokumente übereinstimmen. Um mindestens einen Treffer zu garantieren, wurde ein Dokument der Datenbank mit 200 Objekten als Anfrage ausgewählt.

Für die anderen beiden Trefferarten ist die ausgewählte Anfrage in Abbildung 4.5 dargestellt. Hier wurde die Anfrage relativ klein gewählt, da mit längeren Antwortzeiten zu rechnen ist.

**IC-Treffer** Wie oben bereits erwähnt, müssen bei dieser Trefferart die Kardinalitäten der Anfrage und der Trefferdokumente übereinstimmen. Hierdurch wird der Suchraum stark eingeschränkt. 90% der in der Datenbasis enthaltenen Dokumente können durch diese Bedingung keinen Treffer darstellen. Die Listeneinträge solcher Dokumente werden bei der Abarbeitung ignoriert. Da sie allerdings fester Bestandteil der invertierten Listen sind, müssen sie trotzdem in den Hauptspeicher transferiert werden.

Für die gewählte Anfrage mit 200 Objekten ergaben sich folgende Laufzeiten:

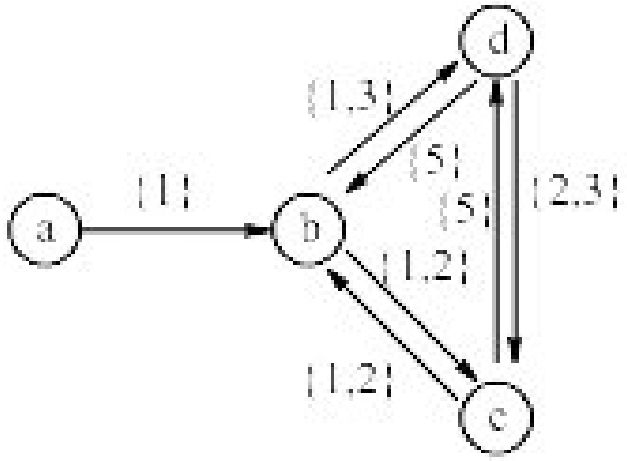



Abbildung 4.4: Die dargestellte Anfrage $G_Q$ dient zur Demonstration der Laufzeitunterschiede in Abhängigkeit der Abarbeitungsreihenfolge und der ausgewählten Basis $Q'$.

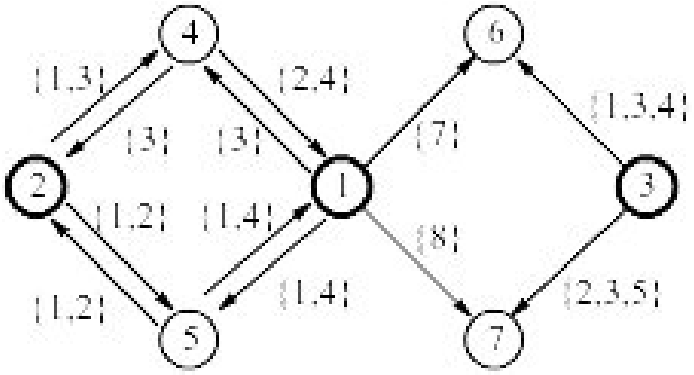



Abbildung 4.5: Anfrage für die Untersuchung der Laufzeiten bei der Berechnung von I- und S-Treffern. Für die beiden Verfahren, die auf einer Teilmenge der Anfrageobjekte arbeiten, wurde $Q' = \{1, 2, 3\}$ gewählt.

| IC-Trefferberechnung | |
|---|---|
| Berechnungsverfahren | Laufzeit |
| `A1` | 7 Sekunden |
| `A2` | über 10 Minuten |
| `A3` | 7 Sekunden |
| `A4` | über 10 Minuten |

Bei dem ersten und dritten Verfahren wird pro Anfrageobjekt genau eine invertierte Liste geladen. Hieraus ergibt sich der Hauptbestandteil der gemessenen Laufzeiten. Von den geladenen Listeneinträgen wurden nur diejenigen genauer bearbeitet, für die das referenzierte Dokument die gleiche Kardinalität wie die Anfrage besaß. Hierdurch blieb die Anzahl der zwischengespeicherten Trefferabbildungen sehr gering.

Beim zweiten und vierten Verfahren wurden statt 200 nur 80 invertierte Listen benötigt, da hier die Basis $Q'$ von $G_Q$ aus 80 Objekten bestand. Allerdings muß bei diesen Verfahren bei jeder Kombination von Trefferabbildungen zur Überprüfung der Bedingungen auf das Dokument zugegriffen werden. So ergaben sich extrem lange Laufzeiten von über 10 Minuten.

Während bei den beiden schnellen Verfahren die Laufzeiten bei mehrmaligen Wiederholungen des Tests weitgehend konstant blieb, gab es bei den beiden anderen Verfahren teilweise erhebliche Abweichungen. Daher ist der angegebene Wert nur als grober Richtwert anzusehen. Allerdings verdeutlicht er, daß sich die beiden langsamen Verfahren kaum für den praktischen Einsatz eignen.

**SC-Treffer** Wie bei der IC-Trefferberechnung wurde auch hier ein Dokument der Datenbasis als Anfrage gewählt, um mindestens einen Treffer zu garantieren. Allerdings müssen für die SC-Trefferberechnung wesentlich mehr invertierte Listen pro Anfrageobjekt abgearbeitet werden, was im Vergleich zur IC-Trefferberechnung zu deutlich längeren Laufzeiten führt.

| SC-Trefferberechnung | |
|---|---|
| Berechnungsverfahren | Laufzeit |
| `A1` | 93 Sekunden |
| `A2` | 67 Sekunden |
| `A3` | 88 Sekunden |
| `A4` | 59 Sekunden |

Verfahren `A3`, das zur Zwischenspeicherung der Bilder der Trefferabbildungen Mengen verwendet, erzielt im Vergleich zu Verfahren `A1` eine bessere Laufzeit. Gleiches gilt für den Vergleich zwischen den Verfahren `A4` und `A2`.

Deutliche Zeitgewinne lassen sich durch die Einschränkung der Suche auf eine Basis $Q'$ von $G_Q$ erzielen (Verfahren `A2` und `A4`). Hier sind Dokumentzugriffe, die bei der Berechnung von IC-Treffern erforderlich waren, nicht nötig.

Bei der absoluten Betrachtung der Antwortzeiten ist zu beachten, daß die gewählte Anfrage mit 200 Objekten als Extremfall anzusehen ist. In der Praxis dürfte mit erheblich kleineren Anfragen im Verhältnis zur Datenbankgröße zu rechnen sein.

**I-Treffer** Für die Untersuchung von I- und S-Treffern wurden Anfragen mit kleinerer Kardinalität gewählt, da hier eine Einschränkung des Suchraums durch die Kardinalität der Anfrage entfällt. Im Vergleich zu den beiden oben besprochenen Trefferarten ist mit längeren Antwortzeiten zu rechnen.

Als Anfrage wurde der in Abbildung 4.5 dargestellte Graph gewählt. Die Datenbasis enthält für diese Anfrage 3 Trefferdokumente mit jeweils einer Trefferabbildung.

Als Basis $Q'$ ergeben sich nach der Anfrageoptimierung die drei in der Abbildung fett markierten Knoten.

Insgesamt wurden die folgenden Laufzeiten ermittelt:

| I-Trefferberechnung | |
|---|---|
| Berechnungsverfahren | Laufzeit |
| `A1` | 4 Sekunden |
| `A2` | über 10 Minuten |
| `A3` | 3 Sekunden |
| `A4` | über 10 Minuten |

Wie schon bei der IC-Trefferberechnung fällt die hohe Laufzeit bei zwei Verfahren auf. Dies liegt, wie bereits oben motiviert, an den notwendigen externen Zugriffen auf die Daten in den Dokumenten.

Für die anderen beiden Verfahren fällt die Antwortzeit deutlich kürzer aus. Wie schon bei der SC-Trefferberechnung läßt sich durch die Verwendung der $\phi$-Abbildungen ein Laufzeitvorteil erzielen.

**S-Treffer** Bei der allgemeinsten Trefferart wird die gleiche Anfrage wie bei der I-Trefferberechnung verwendet. In der Datenbasis wurden 11 S-Treffer ermittelt. Für die Trefferberechnung ergaben sich die folgenden Laufzeiten:

| S-Trefferberechnung | |
|---|---|
| Berechnungsverfahren | Laufzeit |
| `A1` | 17 Sekunden |
| `A2` | 7 Sekunden |
| `A3` | 14 Sekunden |
| `A4` | 5 Sekunden |

Bei der relativen Betrachtung der Laufzeiten ergeben sich die gleichen Ergebnisse wie bei der SC-Trefferberechnung. Insbesondere ergibt die Einschränkung der Suche auf eine Basis $Q'$ von $G_Q$ einen deutlichen Laufzeitvorteil. Die kürzeste Antwortzeit erzielt wieder das Verfahren `A4`.

**Fazit** Die praktischen Ergebnisse bestätigen die im Fazit von Abschnitt 3.4 gestellten Vermutungen (vgl. Bemerkung 3.4.2): Durch den notwendigen Zugriff auf die Dokumente sind die Verfahren `A2` und `A4` für die Berechnung von IC- und I-Treffern nicht geeignet. Für diese Trefferarten liefern die anderen beiden Verfahren gute Ergebnisse.

Bei den Trefferarten SC und S ist die Bewertung der Verfahren genau andersherum. Hier liefern die Verfahren `A2` und `A4` die mit Abstand besseren Laufzeiten. Im Vergleich zwischen den $\psi$- und den $\phi$-basierten Verfahren schneiden die beiden $\phi$-basierten Verfahren `A3` und `A4` besser ab.

Bei dem absoluten Vergleich der Laufzeiten zwischen den einzelnen Trefferarten ist darauf zu achten, daß für IC- und SC-Treffer erheblich größere Anfragen gewählt wurden als bei den anderen beiden Trefferarten. Die Ergebnisse sind in der folgenden Tabelle zusammengefaßt.

| | $\|Q\| = 200$ | | $\|Q\| = 7$ | |
|---|---|---|---|---|
| Berechnungsverfahren | IC-Treffer | SC-Treffer | I-Treffer | S-Treffer |
| `A1` | 7 Sek. | 93 Sek. | 4 Sek. | 17 Sek. |
| `A2` | 10 Min. | 67 Sek. | 10 Min. | 7 Sek. |
| `A3` | 7 Sek. | 88 Sek. | 3 Sek. | 14 Sek. |
| `A4` | 10 Min. | 59 Sek. | 10 Min. | 5 Sek. |

Bei der SC-Trefferberechnung ist bei kleineren Anfragen mit einer Antwortzeit unterhalb der von der S-Trefferberechnung zu rechnen.

In Abschnitt 4.2.5 wurde eine erweiterte Indexstruktur beschrieben, die speziell an Eigenschaften von IC- und SC-Treffern angepaßt ist. Diese Indexvariante wurde bei obigen Messungen nicht eingesetzt. Sie wird getrennt in Abschnitt 4.3.8 im praktischen Einsatz untersucht.

Die in diesem Abschnitt angegeben Laufzeiten sind eher als Tendenz zu werten. Wie sich zeigen wird, hängt die Laufzeit stark von der jeweiligen Anfrage ab. Bevor auf diesen Zusammenhang eingegangen wird, werden im nächsten Abschnitt modifizierte Varianten der Trefferberechnungsverfahren untersucht.

### 4.3.6 Varianten der Trefferberechnung

Bei den in Abschnitt 3.4 angegebenen Verfahren wurde vor jeder Kombination von Trefferabbildungen überprüft, ob die für einen Treffer nötigen Bedingungen erfüllt sind. Es werden nur diejenigen kombinierten Trefferabbildungen zwischengespeichert, die diese Bedingungen erfüllen. Somit werden frühzeitig diejenigen Trefferabbildungen von der Berechnung ausgeschlossen, die auf keinen Fall einen Treffer beschreiben können. Hierdurch wird die Anzahl der in der Menge $F$ gespeicherten Abbildungen gering gehalten und somit der intern benötigte Speicherplatzbedarf eingeschränkt.

Andererseits benötigt die Überprüfung der Bedingungen vor jeder Kombination in Zeile 26 teilweise erhebliche Rechenzeit, worauf schon im letzten Abschnitt eingegangen wurde.

Die Berechnungsverfahren können so abgeändert werden, daß die Bedingungen vor der Kombination nicht überprüft werden. Dies führt i.d.R. zu erheblich mehr Trefferabbildungen, die zwischengespeichert werden müssen. Erst am Ende werden alle Abbildungen daraufhin überprüft, ob sie eine globale Trefferabbildung darstellen.

Für die so abgeänderten Verfahren ergaben sich für die schon im letzten Abschnitt besprochenen Anfragen die folgenden Laufzeiten:

| | $\|Q\| = 200$ | | $\|Q\| = 7$ | |
|---|---|---|---|---|
| Berechnungsverfahren | IC-Treffer | SC-Treffer | I-Treffer | S-Treffer |
| `A1` | 5 Sek. | 95 Sek. | 4 Sek. | 16 Sek. |
| `A2` | 3 Min. | 70 Sek. | 5 Min. | 7 Sek. |
| `A3` | 5 Sek. | 80 Sek. | 3 Sek. | 13 Sek. |
| `A4` | 2 Min. | 55 Sek. | 5 Min. | 5 Sek. |

Ein Vergleich dieser Laufzeiten mit den im letzten Abschnitt gemessenen verlangt nach folgenden Beobachtungen und Kommentaren:

1. Die extremen Laufzeiten bei der Berechnung von IC- und I-Treffern bei den Verfahren `A2` und `A4` sind deutlicher kürzer geworden. Hier werden deutlich weniger Zugriffe auf die Dokumente benötigt, wodurch sich der Zeitunterschied ergibt.

2. Für die Berechnung von SC-Treffern mit dem Verfahren `A1` und `A2` sind die Berechnungszeiten geringfügig länger geworden. Dies kann mit einer höheren Anzahl an zwischenzuspeichernden Abbildungen erklärt werden.

3. Etwas im Widerspruch zum vorigen Punkt ist die Tatsache, daß für die beiden anderen Verfahren bei der SC-Trefferberechnung die Laufzeiten minimal kürzer geworden sind. Dies läßt sich dadurch erklären, daß mit der Verwendung von $\phi$-Abbildungen zur Beschreibung der Trefferabbildungen insgesamt weniger Abbildungen zwischengespeichert werden müssen als bei den beiden ersten Verfahren.

4. Bis auf die in Punkt 2 beschriebene Ausnahme läßt sich durch die neue Variante bei allen Verfahren Zeitgewinne erzielen, die teilweise allerdings nur im Bereich von Bruchteilen einer Sekunde liegen und in der Tabelle dargestellten gerundeten Zeitangaben nicht entnehmbar ist.

**Fazit** Die oben besprochenen Untersuchungen zeigen, daß die für eine Kombination nötigen Bedingungen nicht bei jedem Schleifendurchlauf überprüft werden müssen. Durch diese Modifikation läßt sich i.d.R. ein Laufzeitgewinn erzielen. Dafür ist mit einem erhöhten Speicherplatzbedarf zu rechnen.

Alternativ können die Bedingungen nur bei Bedarf überprüft werden, z.B. wenn der intern benötigte Speicherplatz zu groß wird.

### 4.3.7 Abhängigkeiten der Laufzeiten

Die letzten Abschnitte haben gezeigt, daß die Antwortzeit u.a. von der Trefferart und dem gewählten Verfahren abhängt. Wie schon erwähnt, hängen die angegebenen Zeitangaben auch von der Datenbasis ab. Zusätzlich spielt auch die Anfrage ein Rolle. Allerdings ist hierbei die Größe einer Anfrage eher sekundär. Vielmehr bestimmen die vorhandenen Kantenbeschriftungen bzw. Vokabeln der einzelnen Knoten die Laufzeit entscheidend mit. Dies wird als nächstes an einigen Beispielen untersucht.

Für die beiden in Abbildung 4.6 dargestellten Anfragen ergeben sich für die S-Trefferberechnung mit Verfahren `A4` die folgenden Laufzeiten:

| S-Trefferberechnung | |
|---|---|
| Anfrage | Laufzeit |
| $G_{Q_1}$ | 16 Sekunden |
| $G_{Q_2}$ | 2 Sekunden |

Obwohl beide Anfragen aus zwei Objekten bestehen unterscheiden sich die Antwortzeiten deutlich. Während bei der ersten Anfrage ca. 4 Millionen Listeneinträge abgearbeitet werden müssen, sind es bei der zweiten Anfrage nur ca. 2000. Dies spiegelt sich in den in Abschnitt 4.2.4 eingeführten Kosten für ein Objekt bzw. für eine Anfrage wider. Die Kosten ergeben sich aus der Anzahl an abzuarbeitenden Listeneinträgen. Die Antwortzeit hängt also nicht von der Anzahl der Objekte in der Anfrage ab, sondern von den Kosten für die Anfrageobjekte.

Auch strukturelle Eigenschaften einer Anfrage können die Antwortzeit beeinflussen. Für die in Abbildung 4.7 dargestellten beiden Anfragen wurden die folgenden Laufzeiten gemessen:

| S-Trefferberechnung | |
|---|---|
| Anfrage | Laufzeit |
| $G_{Q_1}$ | 19 Sekunden |
| $G_{Q_2}$ | 30 Sekunden |

Beide Anfragen besitzen ungefähr die gleichen Kosten. Trotzdem ist die Antwortzeit von der ersten Anfrage deutlich kürzer. Die Anzahl an Treffern ist für beide Anfragen fast gleich: 3 für die erste Anfrage, 5 für die zweite.

Betrachtet man beide Anfragen genauer, so besitzt die erste Anfrage einen Zyklus, was bei der zweiten Anfrage nicht der Fall ist. Nach der Abarbeitung der beiden Anfrageobjekte $a$ und $b$ wurden bei der zweiten Anfrage deutlich mehr Trefferabbildungen zwischengespeichert. Durch den Zyklus wird bei der ersten Anfrage der Suchraum stark eingeschränkt, was zu einer geringeren Anzahl an zwischengespeicherten Trefferabbildungen führt. Dies beeinflußt die Abarbeitungszeit der restlichen Anfrageobjekte, da für die zweite Anfrage wesentlich mehr Kombinationsmöglichkeiten vorhanden sind als bei der ersten.

**Fazit** Nicht die Größe einer Anfrage, sondern die Kosten einer Anfrage bestimmen die Beantwortungszeit mit. Hierbei ist die Anzahl der zu bearbeitenden Einträge in den invertierten Listen entscheidend.

Auch strukturelle Eigenschaften einer Anfrage, z.B. wenn ein Zyklus vorhanden ist, können die Antwortzeit beeinflussen. Diese Eigenschaften sind nicht in dem in Abschnitt 4.2.4 definierten Kostenmaß berücksichtigt.

### 4.3.8 Optimierte IC- und SC-Trefferberechnung

In Abschnitt 4.2.5 wurde eine Indexstruktur definiert, die speziell an die Eigenschaften der IC- und SC-Treffer angepaßt wurde. Die dort definierten erweiterten Vokabeln unterscheiden sich von den bisher verwendeten Vokabeln dadurch, daß zusätzlich die Dokumentkardinalität mit abgespeichert wird. In den getesteten Anwendungen benötigte das erweiterte Vokabular über 800 MB an

Speicherplatz und paßt somit nicht mehr in den Hauptspeicher. Dies ist ein erheblicher Nachteil im Vergleich zum nicht erweiterten Vokabular, das mit 19 MB Speicherplatz problemlos im Hauptspeicher vorliegen kann.

Um den benötigten Speicherplatz des erweiterten Vokabulars einzuschränken, wird eine andere Darstellungsform eingesetzt, in der nicht alle erweiterten Vokabeln explizit aufgelistet werden müssen.

Zwar werden die invertierten Listen bzgl. des erweiterten Vokabulars gebildet, zur Speicherung des erweiterten Vokabulars $\mathcal{V}^+$ werden allerdings die nicht erweiterten Vokabeln verwendet. Zusätzlich wird für jede Vokabel $v \in \mathcal{V}$ eine Liste $K_v$ von Dokumentkardinalitäten auf dem Sekundärmedium gespeichert, d.h. für $v = (a, b, c)$ gilt $K_{(a,b,c)} := \{k | (a, b, c, k) \in \mathcal{V}^+\}$. Somit sind alle Informationen verfügbar, die in den erweiterten Vokabeln enthalten sind. Neben dem klassischen Index, der 19 MB an Hauptspeicher benötigt, werden für die Auslagerung der Listen von Dokumentkardinalitäten ca. 280 MB auf dem Massenspeicher belegt. Durch die in Abschnitt 4.2.2 beschriebenen Komprimierungsverfahren läßt sich dieser Speicherbedarf auf 115 MB reduzieren.

Bei der Berechnung von IC- oder SC-Treffern wird auf die Listen von Dokumentkardinalitäten zugegriffen. Hierdurch lassen sich Vokabeln $(a, b, c) \in \mathcal{V}$ zusammen mit einer Dokumentkardinalität $k \in K(v)$ zu erweiterten Vokabeln $(a, b, c, k) \in \mathcal{V}^+$ präzisieren.

So ist man in der Lage, gezielt auf die invertierten Listen zuzugreifen, für deren Einträge die Dokumentkardinalität mit der Kardinalität der Anfrage übereinstimmt. Hierdurch läßt sich das Datentransfervolumen für die Berechnung von IC- oder SC-Treffern im Vergleich zu den in Abschnitt 3.4 untersuchten Verfahren erheblich einschränken, worauf bereits in Abschnitt 4.2.5 eingegangen wurde. Dies spiegelt sich in den gemessenen Laufzeiten wider:

| | $\|Q\| = 200$ | | $\|Q\| = 7$ | |
|---|---|---|---|---|
| invertierte Listen | IC-Treffer | SC-Treffer | I-Treffer | S-Treffer |
| mit Dokumentkardinalitäten | 1 Sek. | 7 Sek. | 4 Sek. | 9 Sek. |
| ohne Dokumentkardinalitäten | 5 Sek. | 55 Sek. | 3 Sek. | 5 Sek. |

Auffallend ist die erheblich geringere Laufzeit bei der IC- sowie bei der SC-Trefferberechnung. Bei der Berechnung von S-Treffern hat sich die Laufzeit fast verdoppelt. Hier macht sich der Mehraufwand für die Vereinigung der vielen kleinen invertierten Listen bemerkbar. Als nächstes wird ein Ansatz beschrieben, der diesen Nachteil beseitigt.

**Alternative Indexstruktur** Die vorherigen Auswertungen haben gezeigt, daß die feinere Indexgranularität die IC- und SC-Trefferberechnung beschleunigt, jedoch die Berechnung von I- und S-Treffern verlangsamt. Als nächstes wird eine alternative Anordnung der invertierten Listen besprochen, die einen direkten Zugriff auf Einträge einer bestimmten Dokumentkardinalität ermöglicht ohne daß die invertierten Listen zerteilt werden müssen.

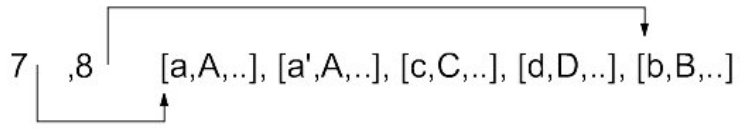
Ausgangspunkt sind die invertierten Listen wie sie für das nicht erweiterte Vokabular eingesetzt werden. Allerdings werden die Listeneinträge nicht nach der Dokumentnummer, sondern nach der Kardinalität des im Eintrag referenzierten Dokuments sortiert gespeichert.

An den Anfang jeder invertierten Liste $L$ wird ein "Directory" gespeichert. Ein Directory ist eine Folge $F_L$ von Paaren $(k, z)$. Für jede Dokumentkardinalität $k$, für die mindestens ein Eintrag $E \in L$ existiert, dessen entsprechende Dokumentkardinalität $k$ ist, enthält $F_L$ ein Paar $(k, z)$. Dabei ist $z$ eine Referenz auf den ersten Listeneintrag in $L$, dessen Dokumentkardinalität $k$ ist. Formal ergibt sich ein Directory $F_L$ einer invertierten Liste $L = (E_1, \ldots, E_\ell)$ mit $|D(E_1)| \leq \ldots \leq |D(E_\ell)|$ bzgl. einer Datenbasis $\mathcal{D}$ als

$$F_L := \Big\{ (k, z) \in \mathbb{N} \times [1 : \ell] \Big| k = |D(E_z)| \wedge \forall 1 \leq i < z : |D(E_i)| < |D(E_z)| \Big\}.$$

**Beispiel 4.3.1.** Betrachten wir eine Datenbasis aus vier Dokumenten $A, B, C, D$ mit $|A| = |C| = |D| = 7$ und $|B| = 8$. Seien $a, a' \in A$, $b \in B$, $c \in C$ und $d \in D$ Objekte mit der gleichen Vokabel $v$. Dann ergibt sich als invertierte Liste $L(v)$ mit Directory: (Hierbei besteht das Directory aus den beiden Dokumentkardinalitäten 7 und 8 mit den entsprechenden Verweisen, die durch Pfeile

dargestellt sind.)




∘

Durch diese Modifikationen lassen sich I- und S-Treffer wie bisher berechnen, ohne daß durch die Verwendung des erweiterten Vokabulars zusätzliche Vereinigungen von invertierten Listen anfallen. Weiterhin kann über die im Directory einer Liste gespeicherten Informationen direkt auf Einträge mit einer bestimmten Dokumentkardinalität zugegriffen werden, was die IC- und SC-Trefferberechnung beschleunigt. In der Praxis werden Referenzen auf Einträge durch Dateipositionen innerhalb der komprimierten invertierten Liste realisiert.

Die Referenzierung von Einträgen in invertierten Listen ist nicht neu. Z.B. werden in [33] verschiedene Varianten diskutiert. Dort wird auch auf folgenden Nachteil eingegangen: Eine Referenzierung von Listeneinträgen setzt bei komprimierten invertierten Listen einen wahlfreien Zugriff auf die referenzierten Einträge voraus. Bei den komprimierten invertierten Listen, die mit den in Abschnitt 4.2.2 beschriebenen Verfahren komprimiert wurden, ist ein wahlfreier Zugriff nicht möglich, da eine Liste immer von Beginn an dekomprimiert werden muß. In [33] werden mehrere Ansätze diskutiert, wie man wahlfreien Zugriff in komprimierten invertierten Listen ermöglichen kann. Alle Ansätze erreichen nicht die Kompressionsrate, wie sie ohne eine Referenzierung erzielt werden kann. Somit ist bei referenzierten komprimierten invertierten Listen mit einem höheren Speicherplatzbedarf zu rechnen.

Für die Trefferberechnung wurden innerhalb einer invertierten Liste die Einträge mit gleicher Dokumentkardinalität mittels des in Abschnitt 4.2.2 beschriebenen Verfahrens basierend auf einer Golomb-Kodierung komprimiert. Auch das Directory läßt sich entsprechend komprimieren. Bei der Testanwendung ergab sich ein Speicherplatzbedarf von 315 MB. Dabei belegen die Directories aller Listen 195 MB und die komprimierten invertierten Listen 220 MB externen Speicher. Damit steigt der Speicherplatzbedarf im Vergleich zu nicht referenzierten invertierten Listen um 138 MB. Dieser kann durch eine kompaktere Wahl zur Darstellung von Referenzen sicherlich noch gesenkt werden.

Bei der Testanwendung wurden für die Trefferberechnung folgende Laufzeiten gemessen:

| | $\|Q\| = 200$ | | $\|Q\| = 7$ | |
|---|---|---|---|---|
| invertierte Listen | IC-Treffer | SC-Treffer | I-Treffer | S-Treffer |
| mit Directory | 2 Sek. | 8 Sek. | 3 Sek. | 6 Sek. |
| mit Dokumentkardinalitäten | 1 Sek. | 7 Sek. | 4 Sek. | 9 Sek. |
| ohne Dokumentkardinalitäten | 5 Sek. | 55 Sek. | 3 Sek. | 5 Sek. |

Im Vergleich zum vorherigen Ansatz hat sich der gestiegene Speicherplatz geringfügig auf die Laufzeiten ausgewirkt. Die Verschlechterungen liegen i.d.R. unterhalb einer Sekunde, was durch die gerundet angegebenen Werte etwas verschleiert wird. Allerdings werden deutliche Laufzeitvorteile bei den Trefferarten IC und SC im Vergleich zu den Verfahren ohne Dokumentkardinalitäten deutlich.

**Fazit** Die Modifikationen haben gezeigt, wie sich die IC- und SC-Trefferberechnung beschleunigen läßt. Allerdings muß hierbei ein erhöhter Speicherplatzbedarf in Kauf genommen werden. Durch den Einbezug der Dokumentkardinalitäten kann der Index durchaus 50 % mehr Speicherplatz belegen als ein Index ohne direkten Einbezug von Dokumentkardinalitäten.

Die Verwendung eines Directory pro invertierter Liste ist i.d.R. einer sturen Auflistung der Dokumentkardinalitäten vorzuziehen. Bei einem Einsatz der Directory-Variante bieten sich noch einige Optimierungsmöglichkeiten an, die bei den bisherigen Implementierungen noch nicht berücksichtigt wurden.

### 4.3.9 Vergleich mit bekannten Verfahren

In Abschnitt 3.1.2 wurden alternative Verfahren zur Berechnung von Graphenisomorphien vorgestellt. Diese arbeiten direkt auf den Dokumenten. Strategien, durch Filterung oder Kodierung den Suchraum einzuschränken, um nicht alle Dokumente einer Datenbasis zu laden und auf eine Isomorphie zu überprüfen, lassen sich nicht auf die Trefferberechnung aller vier Trefferarten übertragen. Hierauf wurde bereits in Abschnitt 3.1.2 eingegangen.

Getestet wurde ein Verfahren, das alle Dokumente einer Datenbasis sequentiell durchläuft und mittels klassischer Tiefensuche jedes Dokument auf (Sub-) Graphenisomorphie untersucht. In Abschnitt 2.4 wurde beschrieben, wie sich jede der vier Trefferarten durch ein Isomorphieproblem zwischen Graphen ausdrücken läßt. Es wurden folgende Laufzeiten für die Anfragen aus Abschnitt 4.3.5 gemessen:

| | $\|Q\| = 200$ | | $\|Q\| = 7$ | |
|---|---|---|---|---|
| Verfahren | IC-Treffer | SC-Treffer | I-Treffer | S-Treffer |
| Tiefensuche | 13 Sek. | 28 Sek. | 139 Sek. | 300 Sek. |
| invertierte Listen | 5 (1) Sek. | 55 (7) Sek. | 3 Sek. | 5 Sek. |

Bei der Bewertung der Laufzeiten ist zu beachten, daß man für den sequentiellen Transfer aller Dokumente der Datenbasis in den Hauptspeicher 130 Sekunden benötigt. Bei den Trefferarten IC und SC müssen allerdings nur 10% aller Dokumente untersucht werden. Die restlichen scheiden wegen unterschiedlicher Kardinalitäten zwischen Anfrage und Dokument aus.

Die Antwortzeit bei der IC- und I-Trefferberechnung wird durch die Transferzeit dominiert. Die eigentliche Berechnungsdauer fällt dagegen kaum ins Gewicht.

Vergleicht man obige Laufzeiten mit den Laufzeiten aus den letzten Abschnitten, so schneiden die klassischen Verfahren bis auf die Berechnung von SC-Treffern deutlich schlechter ab. Bei der SC-Trefferberechnung spricht die Einschränkung des Suchraums auf 10% aller Dokumente der Datenbasis und die Größe der Anfrage für das klassische Verfahren. Wie schon in den letzten Abschnitten besprochen, müssen bei der getesteten, mit 200 Objekten relativ großen Anfragen, viele invertierte Listen abgearbeitet werden. Dies führt im Vergleich zum klassischen Verfahren zu längeren Laufzeiten.

Durch die in Abschnitt 4.3.8 beschriebene Optimierung lassen sich durch die Verwendung von invertierten Listen auch bei der SC-Trefferberechnung bessere Laufzeiten als bei der Tiefensuche erzielen. Diese sind in obiger Tabelle in Klammern aufgeführt. Allerdings wird für die Erweiterung ein sehr großer Index benötigt.

**Fazit** Die sequentielle Untersuchung aller Dokumente auf Treffer mittels Graphenisomorphien hat durchaus seine Berechtigung, wenn für einen Index nicht ausreichend Speicherplatz zur Verfügung steht. Durch die Verwendung von invertierten Listen lassen sich bis auf wenige Ausnahmen kürzere Laufzeiten erzielen. Hierfür ist mit einem moderaten zusätzlichen Speicherplatzbedarf für den Index zu rechnen. Durch eine Optimierung der invertierten Listen lassen sich für die Trefferarten IC und SC noch bessere Laufzeiten erzielen. Allerdings steigt hierbei der für den Index benötigte Speicherplatz deutlich an.

### 4.3.10 Kombinierte Verfahren

Die auf invertierten Listen basierten Verfahren sind i.d.R. schneller als die klassische Tiefensuche. Dies haben die Untersuchungen im letzten Abschnitt gezeigt. Allerdings existieren auch Fälle, in den die klassischen Verfahren, z.B. eine Tiefensuche, schneller ist. Man vergleiche dazu die im vorherigen Abschnitt angegebenen Laufzeiten für die SC-Trefferberechnung.

Beide Verfahren lassen sich kombinieren. In einem ersten Schritt werden nur die Objekte einer Anfrage mittels invertierter Listen abgearbeitet, die geringe Kosten besitzen (vgl. Abschnitt 4.2.4). Hierdurch wird der Suchraum i.d.R. stark eingeschränkt. Für Anfrageobjekte mit hohen Kosten werden die invertierten Listen nicht verwendet. Statt dessen wird in einem zweiten Schritt die klassische Tiefensuche auf die Dokumente aus dem eingeschränkten Suchraum angewendet. Hierbei

liefern die bereits abgearbeiteten Anfrageobjekte, bzw. die bei den Trefferberechnungsverfahren zwischengespeicherten Abbildungen in der Menge $F$, konkrete Startpunkte für die Tiefensuche in den Dokumenten.

Eine Implementierung der kombinierten Verfahren erzielte für die SC-Trefferberechnung der Anfrage, die auch im letzten Abschnitt zum Einsatz kam, eine Laufzeit von 20 Sekunden. Damit ist dieses Verfahren für diese Anfrage schneller als die beiden getrennten Verfahren.

Der Zeitpunkt, wann von der Verwendung von invertierten Listen zur klassischen Tiefensuche gewechselt wird, hängt im wesentlichen von zwei Punkten ab:

1. den Kosten der Anfrageobjekte und
2. wie stark der Suchraum eingeschränkt wird.

Beide Werte sind extrem systemabhängig. Durch Testläufe auf den entsprechenden Systeme können diese Werte grob bestimmt werden.

**Fazit** Durch eine Kombination der klassischen Tiefensuche und der Verwendung von invertierten Listen für einen Teil der Anfrage lassen sich die positiven Laufzeiteigenschaften aus beiden Verfahren ausnutzen.

### 4.3.11 Architekturanwendung

In Abschnitt 2.2 wurde eine Anwendung aus dem Architekturbereich besprochen, die als Motivation für die Treffersuche dient. Nun werden exemplarisch Resultate der Trefferberechnung für diese Anwendung untersucht. Vorher wird auf spezielle Eigenschaften der Architekturanwendung eingegangen. Diese anwendungsspezifischen Eigenschaften werden bei der Trefferberechnung ausgenutzt. Hierbei steht die Berechnung der allgemeineren Trefferarten SC und S im Vordergrund.

Als Menge von Relationen $\mathcal{R}$ wird für die Architekturanwendung die in Abschnitt 2.2 definierte Menge $\mathcal{R}_{Kreuzungen}$ verwendet, die aus neun Kreuzungsrelationen besteht.

Objekte lassen sich in waagerechte und senkrechte Objekte disjunkt unterteilen. Für die Kreuzungsrelationen gilt, daß zwei in Relation stehende Objekte immer unterschiedliche Ausrichtungen besitzen.

Allgemein lassen sich Relationen, die Kreuzungen zwischen Objekten beschreiben, symmetrisch definieren. Die in der Architekturanwendung verwendeten neun Kreuzungsrelationen sind ein Beispiel, wie sich symmetrische Relationen durch asymmetrische ersetzen lassen. Hierbei ist die disjunkte Zerlegung von $\mathcal{O}$ in waagerechte und senkrechte Objekte wichtig. Es gilt $\mathcal{O} = W \sqcup S$, wobei $W$ die Menge aller waagerechten und $S$ die Menge aller senkrechten Objekte ist. Hiermit läßt sich für gegebene symmetrische Relationen $R'$ die asymmetrische Relation $R$ definieren durch

$$R = \left\{(w, s) \in W \times S | (w, s) \in R'\right\}.$$

Da der für einen Index benötigte Speicherplatz u.a. von der Anzahl $\sum_{D \in \mathcal{D}} |(D \times D) \cap R|$ der in Relation $R$ zueinander stehenden Objekte abhängt, läßt sich durch die Verwendung von antisymmetrischen Relationen anstelle symmetrischer Relationen i.d.R. teilweise erheblich Speicherplatz einsparen.

Bei der Darstellung von Dokumenten durch Graphen entspricht jedem Objekt ein Knoten. Zwei Knoten sind durch eine Kante verbunden, wenn die entsprechenden Objekte in mindestens einer Relation zueinander stehen.

Da bei der allgemeinen Trefferberechnung zwei Objekte in mehreren Relationen zueinander stehen können, ist die Menge der möglichen Kantenbeschriftungen $W_E$ definiert als die Potenzmenge von $\{1, \ldots, |\mathcal{R}|\}$. Bei der Architekturanwendung sind die Kreuzungsrelationen so definiert, daß zwei Objekte in höchstens einer Relation aus $\mathcal{R}$ stehen. Daher wird hier die Menge der möglichen Kantenbeschriftung als $W_E = \{1, \ldots, |\mathcal{R}|\}$ definiert.

Durch die Verwendung der antisymmetrischen Relationen besitzen alle Graphen die Eigenschaft, daß Knoten entweder nur ausgehenden Kanten oder nur eingehende Kanten besitzen. Knoten mit nur ausgehenden Kanten entsprechen Objekten mit waagerechter Ausrichtung, die anderen Knoten entsprechen senkrechten Objekten.

Primär für die Berechnung von SC- und S-Treffern läßt sich das Vokabular und die Anzahl an Listeneinträgen im Vergleich zu einer allgemeinen Anwendung halbieren. Statt für jedes Objekt einen Listeneintrag zu erzeugen, wird dies entweder nur für alle waagerechte Objekte oder nur für alle senkrechten Objekte getan. Im folgenden wird dies für alle waagerechten Objekte besprochen. Die Ausführungen gelten analog für eine Wahl der senkrechten Objekte.

Für ein waagerechtes Objekt $o$ besteht die Vokabel von $o$ nur aus Informationen über die Anzahl der ausgehenden Kanten und deren Kantenbeschriftungen. Analoges gilt für Listeneinträge. Hier ist nur der erste Tupeleintrag nicht leer.

Im Vergleich zur allgemeinen Trefferberechnung halbiert sich die Indexgröße und Vokabelgröße, wenn die Anzahl von waagerechten und senkrechten Objekten gleich ist. Nur für die Hälfte aller Objekte wird eine Vokabel im Vokabular gespeichert und ein Listeneintrag erzeugt.

Ein so erzeugter Index setzt allerdings den Einsatz eines der beiden Verfahren voraus, die auf einer Basis $Q'$ von $G_Q$ arbeiten. Weiterhin ergibt sich für eine Anfrage $G_Q$ die Basis $Q'$ als die Menge aller Knoten, die waagerechte Objekte repräsentieren. Eine Anfrageoptimierung bzgl. der Wahl von $Q'$ ist hier wegen des eingeschränkten Vokabulars nicht möglich.

**Auswertung** Für die Architekturanwendung wurde eine Datenbasis verwendet, die in Anzahl an Dokumenten und Objekten der in den letzten Abschnitten besprochenen Datenbasis gleicht.

Der für den Index benötigte Speicherplatz ist mit 130 MB ungefähr halb so groß wie für die allgemeine Anwendung.

Untersuchungen mit einem kompletten Index, wie er für die allgemeinen Anwendungen verwendet wird, ergab für einige Anfragen die gleichen Laufzeiten wie bei dem speziellen Index. Allerdings ergab sich für andere Anfragen eine deutlicher Laufzeitunterschied bei der Verwendung des speziellen und des normalen Indexes. Dieser Zusammenhang wird nun exemplarisch beschrieben.

Für die beiden in Abbildung 4.8 dargestellten Anfragen wurden folgende Laufzeiten gemessen:

| S-Trefferberechnung, `A4` | | |
|---|---|---|
| Index | $G_{Q_1}$ | $G_{Q_2}$ |
| normal | 3 Sek. | 3 Sek. |
| speziell | 3 Sek. | ca. 180 Sek. |

Es ergeben sich deutlich längere Laufzeiten bei der Verwendung des speziellen Indexes. Betrachtet man die Kosten der verwendeten Basen der beiden Anfragen, so sind diese für den speziellen Index erheblich höher als für den kompletten Index. Hier macht sich die feste Wahl der Basis durch den speziellen Index deutlich negativ bemerkbar.

**Fazit** Die Anwendung in der Architekturdomäne hat gezeigt, daß sich anwendungsspezifische Eigenschaften bei der Indexkonstruktion ausnutzen lassen. Die Trefferberechnungsverfahren sind so allgemein gehalten, daß sie auch auf so spezialisierte Indexe angewendet werden können.

Die durchgeführten Untersuchungen ergaben, daß sich für die Architekturanwendung die Indexgröße im Vergleich zum allgemeinen Ansatz halbieren läßt. Allerdings muß hier bei einigen Anfragen mit deutlich längeren Antwortzeiten gerechnet werden.

## 4.4 Fazit

In diesem Kapitel wurden Ansätze zur praktischen Optimierung der Berechnungsverfahren vorgestellt. Außerdem wurden praktische Resultate der Berechnungsverfahren untersucht. Hier hat sich gezeigt, daß die Laufzeit primär von den Kosten einer Anfrage bestimmt wird. Die Größe der Anfrage, d.h. die Anzahl der Anfrageobjekte, ist für die Laufzeit kaum entscheidend. Die Kosten

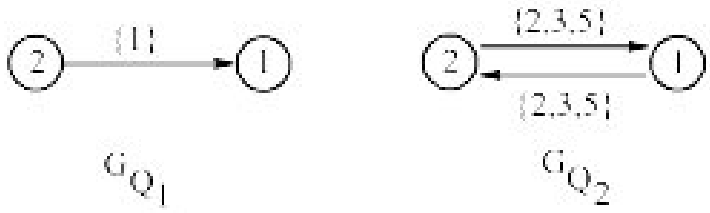



Abbildung 4.6: Zwei Anfragen mit gleicher Knotenzahl, aber unterschiedlichen Antwortzeiten.

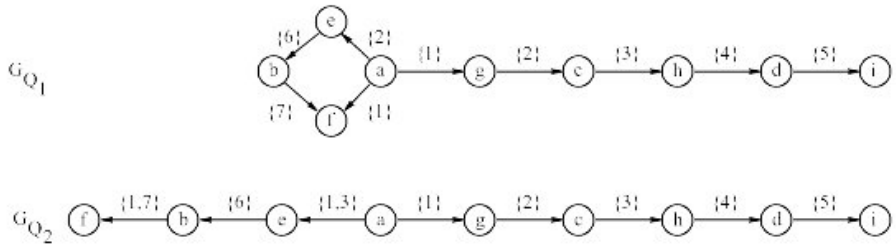



Abbildung 4.7: Zwei Anfragen mit gleichen Kosten, aber unterschiedlichen Antwortzeiten. Die Anzahl an Treffern ist für beide Anfragen fast gleich.

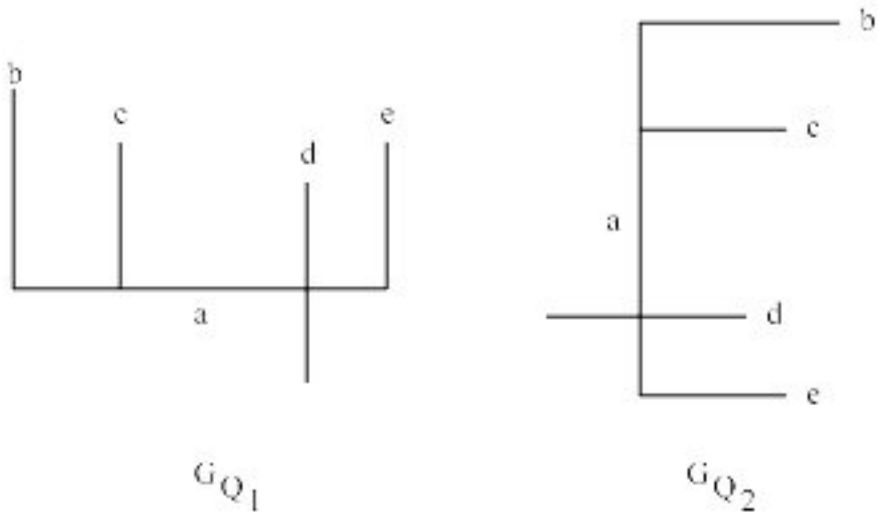



Abbildung 4.8: Zwei Biespielanfragen aus der Architekturdomäne zur Demonstration der Laufzeitunterschiede bei einem speziellen Index.

hängen primär von der Größe der invertierten Listen ab, die für die Berechnung benötigt werden. Hier läßt sich bei zwei Verfahren durch die beschriebene Anfrageoptimierung die Laufzeit teilweise erheblich einschränken. Dies gilt speziell für die beiden allgemeinsten Trefferarten, SC- und S-Treffer.

Die in dieser Arbeit neu vorgestellten Berechnungsverfahren schneiden bei den Tests bis auf einige Sonderfälle deutlich besser ab als die klassischen Verfahren. Für diese Sonderfälle besteht die Möglichkeit, die neuen und die klassischen Verfahren zu kombinieren. Somit ist gewährleistet, daß die hier vorgestellten Verfahren nie schlechtere Laufzeiten besitzen als die klassischen Verfahren.

Die Ergebnisse zeigen auch, daß die Antwortzeiten stark von der Verteilung der Daten auf die invertierten Listen abhängig ist. Dies ist ein allgemeines Problem bei der Verwendung von invertierten Listen [50], [4]. Somit ist für eine konkrete Anwendung zu prüfen, ob dafür typische Anfragen geringe Kosten verursachen.

Bei der absoluten Betrachtung der Antwortzeiten ist darauf zu achten, daß es sich bei den Verfahren um prototypische Implementierungen handelt. Hier sind durch systemabhängige Optimierungen, insbesondere beim Datentransfer zwischen Haupt- und Massenspeicher sowie bei der internen Speicherverwaltung, erhebliche Laufzeitverbesserungen möglich.

# Kapitel 5

# Zusammenfassung

Die in dieser Arbeit vorgestellte Konstellationssuche gliedert sich grob in drei Teile:

1. Die Beschreibung der Konstellationen, d.h. woraus sich Dokumente einer Datenbasis sowie Anfragen zusammensetzen, und wie die Treffer der Konstellationssuche definiert sind. Als formale Grundlage werden beschriftete Digraphen eingesetzt.

2. Die theoretische Trefferberechnung basiert auf einer Indexierung durch invertierte Listen. Das Problem der Trefferberechnung ist mit einer Variante des (Sub-)Graphenisomorphieproblems vergleichbar.

3. Die Trefferberechnung in der Praxis. Es werden vier Verfahren zur Trefferberechnung vorgestellt, Optimierungsstrategien erläutert und praktische Ergebnisse anhand einer prototypischen Implementierung diskutiert.

Ausgangspunkt der Konstellationssuche ist eine Datenbasis von Dokumenten, wobei ein Dokument eine Menge von Objekten ist. Zusätzlich existiert eine Menge von Relationen, durch die Beziehungen zwischen Objekten erfaßt werden.

Die in dieser Arbeit verwendeten Konstellationen werden primär durch die Beziehungen zwischen den Objekten und nur sekundär durch die Objekte selbst beschrieben.

Eine Anfrage an eine Datenbasis besteht, wie ein Dokument, aus einer Menge von Objekten und der Angabe der Beziehungen zwischen den Anfrageobjekten.

Ein Treffer bzgl. einer Anfrage ist eine Teilmenge der Objekte eines Dokuments, zwischen den die gleichen Beziehungen wie bei der Anfrage gelten. Es findet eine detaillierte Unterscheidung in vier Trefferarten statt. Hierdurch wird z.B. festgelegt, ob zusätzlich zu den in einer Anfrage enthaltenen Beziehungen noch weitere in einem Treffer vorhanden sein dürfen (I-Treffer vs. S-Treffer). Die Trefferbegriffe wurden ausführlich in Abschnitt 2.3 beschrieben.

Dokumente und Anfragen lassen sich als beschriftete Digraphen darstellen. Dabei entspricht jedem Knoten ein Objekt und jeder Kante entspricht einer oder mehreren Beziehungen zwischen zwei Objekten. In den Kantenbeschriftungen werden die beteiligten Relationen kodiert. Treffer lassen sich dann durch (Sub-)Graphenisomorphieprobleme beschreiben.

Anwendungsgebiete für die in dieser Arbeit eingeführte Konstellationssuche bzw. Trefferberechnung sind z.B. im Architekturbereich die Suche in Bauplänen, in der Geoinformatik, in der Chemie im Zusammenhang mit der Suche in Molekül datenbanken und in der Wissensrepräsentation.

Zur Berechnung aller Treffer wird ein Index bestehend aus invertierten Listen verwendet. Invertierte Listen haben sich im Zusammenhang mit der Volltextsuche in großen Textbeständen etabliert. Das Konzept der invertierten Listen wurde für die Konstellationssuche erweitert.

Jedes Objekt $d$ eines Datenbankdokuments $D$ erzeugt in genau einer invertierten Liste einen Listeneintrag. In diesem Eintrag werden Informationen über $d$ und über in Relation zu $d$ stehende Objekte persistent gespeichert.

Grob formuliert definiert die Auflistung aller beteiligten Relationen, mit denen $d$ in Bezug zu mindestens einem anderen Objekt steht, die Vokabel von $d$. Die Vokabel spezifiziert die invertierte Liste, in die der Listeneintrag gespeichert wird.

In Abhängigkeit von der Trefferart werden für jedes Objekt $q$ einer Anfrage mit der Hilfe der Vokabel von $q$ die invertierten Listen ermittelt, deren Einträge Vorkommen von $q$ definieren. Ein Vorkommen $d \in D$ besitzt die gleichen Beziehungen zu Nachbarobjekten im Dokument $D$ wie $q$ zu Nachbarobjekten in der Anfrage.

Diese lokalen Informationen werden für Vorkommen von verschiedenen Anfrageobjekten passend zu Treffern kombiniert.

In Kapitel 3 werden alle Schritte der theoretischen Trefferberechnung formal untersucht.

Der letzte Abschnitt von Kapitel 3 bildet den Übergang von der Theorie zur Praxis. Dort werden vier Verfahren zur Berechnung aller Treffer vorgestellt. Die Verfahren eignen sich für alle Trefferarten, besitzen allerdings davon abhängige Vor- und Nachteile.

Die Verfahren setzen die vorangegangenen theoretischen Konzepte der Trefferberechnung um und verwenden den besprochenen Index aus invertierten Listen.

In Kapitel 4 werden Strategien zur Optimierung der Verfahren diskutiert. Dies betrifft hauptsächlich die Handhabung der invertierten Listen: die Bestimmung der für ein Anfrageobjekt benötigten invertierten Listen, die Komprimierung von invertierten Listen und mögliche Anfrageoptimierungen.

Schließlich werden anhand prototypischer Implementierungen der Verfahren die praktischen Resultate der Trefferberechnung untersucht.

**Fazit** Die in dieser Arbeit eingeführten Trefferbegriffe für eine Konstellationssuche sind sehr allgemein gehalten und eignen sich für viele Anwendungen.

Die Trefferberechnungen lassen sich als Spezialfälle von Problemen aus der Graphentheorie darstellen. Hierbei spielen Varianten der Subgraphenisomorphie eine zentrale Rolle. Dies verdeutlicht die enorme Komplexität der Trefferbegriffe.

Trotz der theoretisch zeitaufwendigen Verfahren zur Lösung der allgemeinen Subgraphenisomorphieprobleme wurden spezielle Trefferberechnungsverfahren entwickelt. Dafür wurde das Konzept der Indexierung durch invertierte Listen erweitert und an die Konstellationssuche angepaßt. Die Verfahren verwenden invertierte Listen zum effizienten Zugriff auf die für die Berechnung notwendigen Daten.

Im Gegensatz zu bekannten Verfahren, wo eine Anfrage sequentiell mit allen Dokumenten (Graphen) einer Datenbasis verglichen wird, werden alle Dokumente der Datenbasis gleichzeitig untersucht. Gerade bei großen extern gespeicherten Datenbeständen lassen sich i.d.R. durch die Verwendung von invertierten Listen und durch die damit verbundene gleichzeitige Betrachtung aller Dokumente erhebliche Laufzeitvorteile im Vergleich zu klassischen Verfahren erzielen.

Die vorgestellte Konstellationssuche läßt sich allerdings nicht für alle Anwendungen effizient anwenden. Im schlechtesten Fall sind die neuen Verfahren langsamer als die klassischen sequentiellen Verfahren. Gründe dafür sind Datenbasen und Relationen, für die der Datenbestand auf wenige invertierte Listen verteilt wird. Hierdurch entsteht u.a. ein hohes Datentransfervolumen.

Auch die Anfrage bestimmt die Laufzeit mit. Dies wird durch das eingeführtes Kostenmodell erfaßt. Hierdurch läßt sich abschätzen, wie zeitaufwendig die Anfragebeantwortung ist. Die Kosten hängen primär von den Längen der zu betrachtenden invertierten Listen ab. Die Anzahl der Anfrageobjekte ist hierbei nicht entscheidend.

Die neuen Verfahren lassen sich mit den klassischen Verfahren kombinieren. Erst wird mittels invertierter Listen der Suchraum eingeschränkt. Hierbei werden nur die Anfrageobjekte mit geringen Kosten abgearbeitet. Anschließend werden die im eingeschränkten Suchraum enthaltenen Dokumente sequentiell untersucht.

Abschnitt 4.3.5 hat gezeigt, daß die vier Verfahren für die verschiedenen Trefferarten unterschiedliche Vor- und Nachteile besitzen. Hier kann zum Anfragezeitpunkt automatisch das Verfahren ausgewählt werden, das i.d.R. die für die jeweilige Trefferart besten Antwortzeiten erzielt.

**Ausblick** Speziell für die in dieser Arbeit vorgestellten Trefferberechnungsverfahren lassen sich einige Punkte noch genauer untersuchen, worauf im folgenden kurz eingegangen wird.

Auch wenn sich invertierte Listen für die Indexierung von Textdatenbeständen etabliert haben, muß das in dieser Arbeit erweiterte Konzept weiter auf der "Low-Level" Ebene untersucht werden. Dieser Optimierungsschritt ist allerdings extrem abhängig vom konkret eingesetzten Hardware-System.

In das Kostenmodell aus Abschnitt 4.2.4, das u.a. zur Bestimmung der Abarbeitungsreihenfolge der Anfrageobjekte verwendet wird, sollten systemabhängige Parameter integriert werden. Hierdurch lassen sich die Verfahren besser an die Systemeigenschaften anpassen.

Für die Suche im Vokabular wurde auf klassische Verfahren zurückgegriffen. Hier kann u.U. durch speziell an das Vokabular angepaßte Verfahren Speicherplatz eingespart oder Laufzeitverbesserungen erzielt werden.

Wie schon in Abschnitt 4.2.1 angedeutet, bietet sich eine Umsetzung auf Parallelrechner oder Rechnercluster an. Dieses Umfeld bietet erhebliches Potential zur Laufzeitoptimierung.

Die in Abschnitt 4.3.8 beschriebene Erweiterung der invertierten Listen zur schnelleren Berechnung von IC- und SC-Treffern bietet vielfältige Optimierungsmöglichkeiten (siehe z.B. [33]), die bei der Testimplementierung nicht berücksichtigt wurden. Hier ist mit weiteren Laufzeitverbesserungen zu rechnen.

Schließlich sollte die Verwendung der Trefferberechnungsverfahren in weiteren Anwendungsdomänen untersucht werden. Die in dieser Arbeit verwendete Anwendung im Architekturbereich ist ein erstes Beispiel, durch das eine Portierung in andere Bereiche motiviert wird.

# Literaturverzeichnis


[1] M. Aigner. *Combinatorial Theory.* Grundlehren der mathematischen Wissenschaften 234, Springer-Verlag, 1979.

[2] N. Bartelme. *Geoinformatik: Modelle, Strukturen, Funktionen.* Springer-Verlag, 1995.

[3] W. Bibel, S. Hölldobler, T. Schaub. *Wissensrepräsentation und Inferenz: eine grundlegende Einführung.* Vieweg, 1993.

[4] M. Clausen, R. Engelbrecht, D. Meyer, J. Schmitz. *PROMS: A Web-based Tool for Searching in Polyphonic Music.* Proceedings Int. Symp. on Music Information Retrieval 2000, Plymouth, M.A., USA, 2000.

[5] M. Clausen, H. Goeman, F. Kurth. *Full Media Retrieval: An Algebraic Approach.* In Preparation, 2002.

[6] C.-H. Coulon. *Automatic Indexing, Retrieval and Reuse of Topologies in Architectural Layouts.* In M. Tan, R. Teh (Eds.), The Global Design Studio - Proceedings of the 6th International Conference on Computer-Aided Architectual Design Futures. Singapure, 1995.

[7] J. K. Cringean, M. F. Lynch. *Subgraphs of reduced chemical graphs as screens for substructure searching of specific chemical structures.* J. Information Science, Principles & Practice 15, Elsevier Science Publishers B. V., 1989.

[8] C. J. Date. *A Guide to the SQL Standard.* Addison-Wesley, Reading, MA, USA, 1997.

[9] H. Engesser (Ed.) *Duden Informatik: ein Sachlexikon für Studium und Praxis.* Dudenverlag, 1993.

[10] G. Ellis. *Compiled hierarchical retrieval.* In T. Nagle, J. Nagle, L. Gerholz, P. Eklund (Eds.), Conceptual Structures: Current Research and Practice. Ellis Horwood, 1992.

[11] G. Ellis, F. Lehmann. *Exploiting the induced order on type-labeled graphs for fast knowledge retrieval.* Proc. 2nd Int. Conf. Conceptual Structures: Current Practices, Springer-Verlag, 1994.

[12] R. Englert, A. B. Cremers, J. Seelmann-Eggebert. *Recognition of polymorphic patterns in parameterized graphs for 3d building reconstruction.* In J.-M. Jolion, W. Kropatsch (Eds.), Proceedings of the 1st Workshop on Graph-Based Representations (GbR'97). Springer-Verlag, 1997.

[13] K. Fahnenstich, R. Haselier. *MS Dos 6.0.* Addison-Wesley, 1993.

[14] G. Fischer. *Lineare Algebra.* Vieweg-Studium; 17: Grundkurs Mathematik, 1989.

[15] M. R. Garey, D. S. Johnson. *Computers and Intractability. A Guide to the Theory of NP-Completeness.* Freeman, 1979.

[16] G. Görz (Ed.). *Einführung in die Künstliche Intelligenz.* Addison-Wesley, 1995.

[17] G. Graefe. *Query Evaluation Techniques for Large Databases.* ACM Computing Surveys, 25(2):73-170, 1993.

[18] W. Gräther. *Computing distances between attribute-value representations in an associative memory.* In A. Voß (Ed.), Similarity concepts and retrieval methods. FABEL-Report 13, Gesellschaft für Mathematik und Datenverarbeitung mbH (GMD), Sankt Augustin, 1993.

[19] F. Haller. *ARMILLA - ein installationsmodell.* IFIB, 1985.

[20] F. Harary. *Graphentheorie.* Oldenburg Verlag, München, 1974.

[21] J. E. Hopcroft, J. K. Wong. *Linear Time Algorithm for Isomorphism of Planar Graphs.* ACM Symp. Theory of Computing, 1974.

[22] L. Hovestadt. *A4 - Digital Building: Extensive Computer Support for the Design, Construction, and Management of Buildings.* FABEL-Report 15, Gesellschaft für Mathematik und Datenverarbeitung mbH (GMD), Sankt Augustin, 1993.

[23] T. Ibaraki, T. Kameda. *Optimal nesting for computing N-relational joins.* ACM Trans. on Database Systems, 9(3):482-502, 1984.

[24] A. Kemper, A. Eickler. *Datenbanksysteme: eine Einführung.* Oldenbourg, 1999.

[25] D. E. Knuth. *The Art of Computer Programming - Sorting and Searching.* Volume 3. Addison-Wesley, 1973.

[26] W. Köhler. *The Task of Gestalt Psychology.* Princeton, University Press, 1969.

[27] J. L. Kolodner. *Case-Based Reasoning.* Morgan Kaufmann, San Mateo, 1993.

[28] T. A. Lipkis. *A KL-ONE classifier.* In J. G. Schmolze, R. J. Brachman (Eds.), 1981 KL-ONE Workshop, Cambridge, MA, 1982.

[29] B. D. McKay. *nauty User's Guide (Version 1.5).* Computer Science Department, Australian National University. `http://cs.anu.edu.au/people/bdm/nauty/`.

[30] B. D. McKay. *Practical graph isomorphism.* Congressus Numeratium 30, 1981.

[31] Microsoft Corporation. Microsoft Windows 95, Programmierleitfaden. Microsoft Press Deutschland, 1996.

[32] D. Meyer. *Gestalterkennung in CAD-Plänen - eine Anwendung der Graphentheorie.* Diplomarbeit, Universität Bonn, 1997.

[33] A. Moffat, J. Zobel. Self-Indexing Inverted Files for Fast Text Retrieval. ACM Transaction on Information Systems, Vol 14, No. 4, 1996.

[34] M. Mongeau, D. Sankoff. *Comparison of Musical Sequences.* Computers and the Humanities 24, 1990.

[35] J. Noll. *Musik-Programmierung: MIDI, C und Multimedia.* Addison-Wesley, 1994.

[36] M. N. Nowak. *SAKE - Gestalterkennung in graphischen Layouts durch Abstraktion und Fokussierung.* Diplomarbeit, Universität Bonn, 1996.

[37] W. Oertel, H. Dürschke. *Architekturkonzept für FABEL.* FABEL-Report 5, Gesellschaft für Mathematik und Datenverarbeitung mbH (GMD), Sankt Augustin, 1993.

[38] R. Read, D. Corneil. *The graph isomorphism disease.* Journal of Graph Theory 1, 1977.

[39] E. Rome. *Simulierte Gestalt-Erkennung in Präsentationsgrafiken.* Dissertation zur künstlichen Intelligenz; Bd. 99. Infix, 1995.

[40] D. Shasha. *Database Tuning: A Principled Approach.* Prentice Hall, Englewood Cliffs, NJ, USA, 1992.

[41] A. L. Uitdenbogerd, A. Chattaraj, J. Zobel. *Music IR: Past, Present and Future.* Proceedings International Symposium for Audio Information Retrieval (ISMIR 2000) Plymouth, USA, October 2000.

[42] J. D. Ullman *Principles of Database and Knowledge-Base Systems. Volume 1: Classical Database Systems.* Computer Science Press, 1988.

[43] J. D. Ullman *Principles of Database and Knowledge-Base Systems. Volume 2: The New Technologies.* Computer Science Press, 1989.

[44] J. R. Ullmann. *An algorithm for subgraph isomorphism.* Journal of the ACM, 23, 1976.

[45] L. G. Valiant. *The complexity of enumeration and reliability problems.* SIAM J. Comp. 8, 1979.

[46] J. van Leeuwen. *Algorithms and Complexity.* Handbook of Theoretical Computer Science, Volume A. Amsterdam, Elsevier, 1990.

[47] A. Voß. *FABEL im Überblick.* FABEL-Report 1, Gesellschaft für Mathematik und Datenverarbeitung mbH (GMD), Sankt Augustin, 1993.

[48] K. Wagner. *Graphentheorie.* BI-Hochschultaschenbücher 248/248a, 1970.

[49] P. Willett. *Processing of three-dimensional chemical structure information using graph-theoretic techniques.* In D. I. Raitt, Proc. 14th International Online Information Mtg. Learned Information, Oxford, UK, 1990.

[50] I. H. Witten, A. Moffat, T. C. Bell. *Managing Gigabytes, Compressing and Indexing Documents and Images.* Morgan Kaufman Publishers, 1999.