Identifikation und Berechnung von Unterschieden und Gemeinsamkeiten von Topic Maps

Stephan Felke

28. März 2012

Abstract

In dieser Arbeit wird gezeigt, wie Unterschiede zweier *Topic Maps* bestimmt und effizient berechnet werden können. Es wird ein Überblick über das Topic Maps Datenmodell gegeben und aufgezeigt, welche Schritte notwendig sind, um einen umfassenden Überblick über die Gemeinsamkeiten und Unterschiede zweier *Topic Maps* zu erlangen. Es wird erörtert, wie sich konzeptionelle von sogenannten atomaren Unterschieden abgrenzen lassen. Unterschiede auf atomarer Ebene beziehen sich auf Eigenschaften bestimmter Objekte, wohingegen auf struktureller Ebene die Objekte an sich und deren Beziehungen betrachtet werden. Außerdem wird erklärt, wie beide Begriffe voneinander getrennt und im Zusammenspiel betrachtet werden können. Hierzu werden Operatoren entwickelt, welche auf *Topic Maps* als Operanden arbeiten und im Ergebnis die Unterschiede auf konzeptioneller und atomarer Ebene darstellen, ohne dabei den semantischen Mehrwert des Topic Maps Datenmodells zu verlieren. Anschließend wird auf Grundlage der definierten Operatoren eine Implementierung entwickelt, welche Effizienz, Korrektheit und Nutzbarkeit zum Ziel hat. Es wird gezeigt, wie intelligente Algorithmen die Laufzeit für das Berechnen der Operationen um ein vielfaches verringern können.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung

1.1. Fragestellung und Thema der Arbeit

Im Zuge der stetigen Digitalisierung und der damit verbundenen Sammlung von Daten ergeben sich vielseitige Anwendungsszenarien. Es entstehen immer mehr Quellen für die Informationsverarbeitung und -beschaffung. Leider ist der Vorgang der Informationsdigitalisierung heute zum großen Teil mit manuellem Aufwand verbunden. Jeder Mensch hat eigene Vorstellungen, wie Informationen und Wissen zu repräsentieren sind. Schon beim selben zugrundeliegenden Modell der Datenhaltung können starke Konflikte zwischen zwei Wissensbasen, also jedweder Art von Informationssammlung, entstehen. Zu diesen zählen Unterschiede in der Modellierung der Domäne, in der Codierung der Daten oder auch der verwendeten Terminologie und vieles mehr [Tho09].
Topic Maps ist ein semantisches Datenmodell, welches das Ausmaß der Heterogenität einzuschränken versucht. Zwei separat entwickelte Wissensbasen, die in Topic Maps repräsentiert sind, lassen sich generisch zusammenführen, wobei auch gleich ein semantischer Mehrwert entsteht. Das ist bei den meisten Datenmodellen nicht oder nur schwer möglich.
Neben der Zusammenführung an sich, ist es auch interessant zu wissen, welche Unterschiede überhaupt in den vorliegenden Wissensbasen vorhanden sind und wo Gemeinsamkeiten bestehen. Dies ist das Thema der vorliegenden Arbeit. Es sollen jene Unterschiede und Gemeinsamkeiten von *Topic Maps* aufgezeigt werden. Dabei ist zu Unterscheiden, ob es sich um strukturelle oder atomare Unterschiede handelt. Je nach Betrachtungswinkel ergeben sich hier möglicherweise unterschiedliche Aussagen zum selben Sachverhalt. Während zwei Datenbasen als *Topic Map* strukturell völlig gleich sein können, können sie sich dennoch in gewissen atomaren Eigenschaften unterscheiden. Dieser Fakt ist im Zuge der Arbeit genau herauszustellen und zu behandeln, um beide Betrachtungsansätze zu unterstützen. Ist ein Modell für das Bestimmen der Unterschiede entwickelt, soll anhand einer Umsetzung gezeigt werden, dass eine solche Betrachtungsweise praktisch möglich und einsetzbar ist.

1.2. Ziel der Diplomarbeit

Das Ziel dieser Arbeit ist es Unterschiede und Gemeinsamkeiten in *Topic Maps* zu finden. Dabei wird genau zu definieren sein, was Unterschied und Gleichheit im Kontext von Topic Maps bedeutet. Hierzu werden Operatoren definiert werden, welche analog zur Mengentheorie Vereinigung, Schnitt und Differenz von *Topic Maps* berechnen können. Es sollen die Eigenschaften dieser Operatoren erläutert werden und eine effiziente Implementierung erfolgen.
Konkret sollen folgende Fragestellungen beantwortet werden:

1. Was bedeutet strukturelle Gleichheit zweier *Topic Maps*?
2. Was bedeutet atomare Gleichheit von *Topics*?
3. In welchem Zusammenhang stehen diese Gleichheitsbegriffe?
4. Was ist atomare und strukturelle Ungleichheit und wie kann sie erkannt und bestimmt werden?

Aufbauend hierauf ist es nun möglich die Unterschiede zweier *Topic Maps* genauer zu betrachten. Ist erst ein Modell für die Unterschiede festgelegt, so sollten damit u.a. auch folgende Fragen beantwortet werden können:

1. Was ist strukturelle Differenz?
2. Welche *Topics* bzw. *Associations* befinden sich ausschließlich in einer der beiden *Topic Maps*?
3. Welche *Topics* bzw. *Associations* kommen in beiden *Topic Maps* vor?
4. Welche *Associations* sind betroffen, wenn ein *Topic* gelöscht wurde?
5. Welche Eigenschaften von *Topics* haben sich geändert?

Nachdem die Beantwortung der gestellten Fragen theoretisch erörtert wurde, wird das Framework `TMDiff` vorgestellt. Es werden die Vorgehensweise der Berechnung und die entsprechenden Algorithmen erklärt, mit Hilfe derer die spezifizierten Ergebnisse der Operationen für zwei gegebene *Topic Maps* berechnet werden. Es wird ein besonderes Augenmerk auf die Effizienz der entwickelten Algorithmen gelegt, um Skalierbarkeit und somit das Verarbeiten großer *Topic Maps* gewährleisten zu können.

1.3. Aufbau der Arbeit

In Kapitel 2 werden die Grundlagen für die Arbeit gelegt. Es wird erläutert, was genau *Topic Maps* sind und wie sie funktionieren, d.h. wie Wissen in Form einer *Topic Map* repräsentiert wird. Außerdem wird ein Einblick in Arbeiten gegeben, die im Zusammenhang mit der Fragestellung relevant sind.
Kapitel 3 ist der Hauptteil der Arbeit. Es erfolgt eine mathematische Formalisierung der Konstrukte des Topic Maps Datenmodells (TMDM). Aufbauend hierauf werden nun die Operatoren zum Berechnen der Unterschiede und Gemeinsamkeiten von *Topic Maps* definiert. Anschließend wird in Kapitel 4 das Framework `TMDiff` vorgestellt und die verwendeten Algorithmen erklärt, analysiert und deren Performance anhand der Umsetzung evaluiert. Im Schlussteil erfolgt ein Ausblick auf eventuelle Anwendungen und Forschungsfragen, welche sich aus der hier geschaffenen Grundlage ergeben und eine Zusammenfassung der Arbeit.

2. Grundlagen

2.1. Terminologie

Im Topic Maps Datenmodell (TMDM) werden viele Begrifflichkeiten eingeführt, welche auch in dieser Arbeit konsequent genutzt werden. Fachterme, wie z.B. „*Occurrence*“ könnten auch mit „Belegstelle“ übersetzt werden. Dies erschwert jedoch nur unnütz das Nachschlagen und das Verständnis beim Lesen von Sekundärliteratur. Alle Begriffe, welche in der Arbeit verwendet werden, sind entweder direkt definiert oder aus dem TMDM entnommen und können dort nachvollzogen werden.
Wenn von einem Konstrukt die Rede ist, so ist damit eine Komponente des TMDM gemeint (s. nächster Abschnitt). Ein Fragment oder auch Topic-Fragment ist ein Tupel. Es besteht aus einer Menge von *Subject Identifiers*, *Subject Locators*, *Item Identifiers*, *Names*, *Variants* und *Occurrences*. Ein Topic-Fragment ist eine Repräsentation eines *Topics* aus der gegebenen *Topic Map*.
Topic Maps-Konstrukte werden zum besseren Verständnis kursiv geschrieben. Topic Maps jedoch nicht, wenn damit das Datenmodell gemeint ist. Zu alldem folgen später noch ausführliche Erläuterungen.

2.2. Einführung in das Topic Maps Datenmodell

Die folgenden Ausführungen sind zu großen Teilen an [Pep00] angelehnt, worin Pepper eine gute Einführung in das Thema Topic Maps gibt.
Es folgt ein Überblick über das TMDM und dessen Konstrukte. Als Konstrukte werden alle Komponenten einer *Topic Map* bezeichnet, also *Topics*, *Names*, *Variants*, *Occurrences*, *Associations* und *Roles* (dazu später mehr) und die *Topic Map* selbst. Außerdem wird der Begriff der Identität erläutert, da dieser eine entscheidende Rolle bei der Definition der Operatoren spielt.

2.2.1. Topic Maps

„Eine Topic Map ist eine Repräsentation von Informationen, welche benutzt wird, um Informationsobjekte zu beschreiben und zwischen ihnen zu navigieren.“[New01] Zentral ist hierbei das sogenannte „Aussagegegenstandszentrierte Modellierungsparadigma“ [Mai08]. Dass heißt, die Informationsobjekte stehen bei der Modellierung der Wissensbasis im Zentrum, sie sind sozusagen das „Herz“ der *Topic Maps* [Gar02]. Darauf aufbauend werden Beziehungen (*Associations*) zwischen diesen Aussagegegenständen (*Topics*) und deren Eigenschaften (*Occurrences*) modelliert. Ausgehend von dieser Modellierung ist es nun möglich von *Topics* über *Associations* zu anderen *Topics* zu navigieren. ***T**opics*, ***A**ssociations* und ***O**ccurrences* werden als das „TAO“ der *Topic Maps* bezeichnet [Pep00]. Dieses wird im folgenden Abschnitt erklärt.
Zusammenfassend kann eine *Topic Map* als selbstverwaltende Wissensbasis bezeichnet werden. Jede Information an sich kann als Informationsobjekt interpretiert und somit als *Topic* repräsentiert werden. Die Zusammenhänge, welche jene Informationsobjekte verbinden, werden dann durch *Associations* repräsentiert. Selbstverwaltend ist die Wissensbasis dadurch, dass Informationen automatisch in den für sie relevanten Kontext gestellt werden. Die repräsentierte Information hat in ihrem Kontext eine definierte Identität. Dadurch wird doppelt repräsentiertes Wissen automatisch erkannt und zusammengeführt, wobei etwaige Konflikte aufgelöst werden. Topic Maps ist also ein mächtiges Datenmodell, mit dem Wissen strukturiert repräsentiert und abgespeichert werden kann.

2.2.2. Das Topic Maps TAO

Topics

Jede Entität, welche durch irgendeine Aussage beschrieben wird, kann in einer *Topic Map* als *Topic* repräsentiert werden. Dabei ist es völlig irrelevant um welches Objekt, Subjekt oder welchen materiellen oder immateriellen Gegenstand es sich handelt, so lange nur in irgendeiner Form eine Aussage über die zu betrachtende Entität getroffen werden kann:

> A **subject** can be anything whatsoever, regardless of whether it exists or has any other specific characteristics, about which anything whatsoever may be asserted by any means whatsoever. In particular, it is anything about which the creator of a topic map chooses to discourse. [GM08]

Ein *Topic* repräsentiert also ein „Ding“ aus der realen Welt. Außerdem steht das Wort *Topic* auch für ein Element in der *Topic Map* (welches eigentlich als Topic Link bezeichnet wird) [Pep00]. In dieser Arbeit wird für beides das Wort *Topic* genutzt, da sich jeweils aus dem Kontext erschließt, welche Bedeutung gemeint ist.

Beispiel (s. Abb. 2.1)

Subjects („Dinge“, Entitäten) einer Wissensbasis, die berühmte Naturwissenschaftler und deren Arbeiten modelliert, könnten z.B. „Albert Einstein“, „Wilhelm Ostwald“, „Werner Heisenberg“, „Relativitätstheorie“ und „Leipzig“ sein. Es werden hier also Personen, ein Forschungsgebiet und ein Ort repräsentiert. Diese drei Oberbegriffe sind gleichzeitig Kategorien oder Typen der *Topics*, welche die genannten Subjects in einer *Topic Map* repräsentieren.
Eine tatsächliche Kategorisierung im Datenmodell erhöht den Grad der Strukturierung. Sie ist in Topic Maps durch sogenannte *Topic*-Typen realisiert. Diese Kategorien (oder Typen) sind ebenfalls *Topics* und es wird eine Typ-Instanz-Beziehung zwischen den jeweiligen Instanz- und Typ-*Topics* erstellt. (siehe Abbildung 2.1).

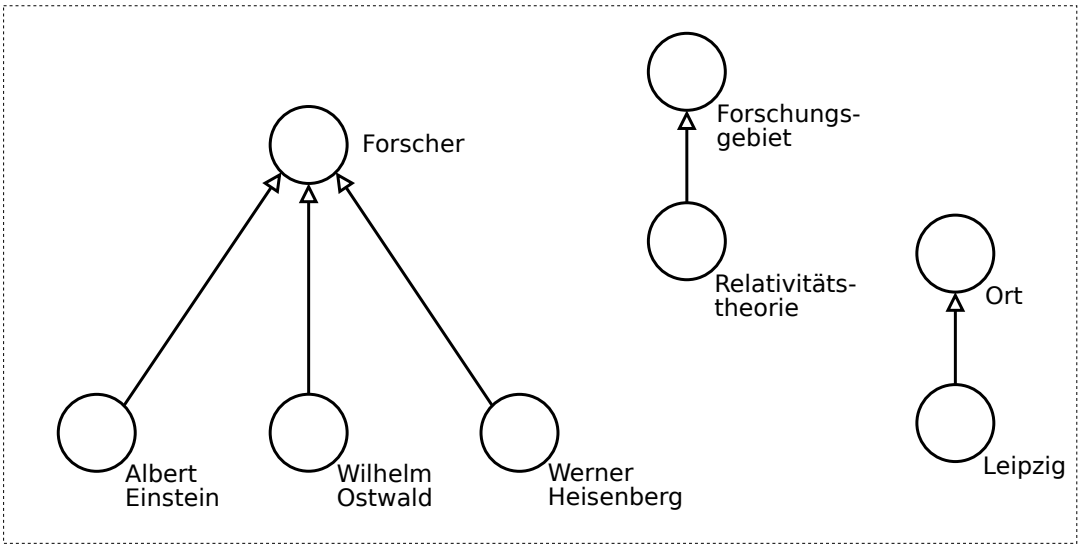

Abbildung 2.1.: Beispiel Topics mit entsprechenden Topic-Typen

Eigenschaften, welche mit einem Aussagegegenstand (*Topic* in der *Topic Map*) verbunden werden, können als *Occurrences* (Abschn. 2.2.2) modelliert werden. Solche Eigenschaften sind z.B. Größe, Geburts- bzw. Sterbeort oder Farbe. Eine besondere Eigenschaft eines Aussagegegenstandes ist dessen Name, da dieser als kontextabhängige Referenz für den Aussagegegenstand genutzt wird. Äquivalent geschieht dies auf Ebene des TMDM. Namen, repräsentiert als *Names*, dienen als kontextabhängiges „Label“ [GM08] des *Topics*. So könnte beispielsweise „Lipsia“ ebenfalls ein *Name* des *Topics* „Leipzig“ im Kontext „historische Namen“ sein. Der *Name* eines *Topics* muss nicht eindeutig in der *Topic Map* sein. So könnte auch ein *Topic*, welches für einen Ort in Nord-Dakota (New Leipzig) steht, durchaus den *Name* „Leipzig“ haben. Dies wirft die Frage auf, wie *Topics* eindeutig referenziert werden können oder anders formuliert: Was ist die Identität eines *Topics*? Dies ist eine wichtige Frage, welche ausführlich in Abschnitt 2.5 beantwortet wird. Der *scope* (s. Def. 3.2) eines *Names* definiert den schon genannten Kontext oder anders gesagt, die Bedingungen, unter welchen der vorgestellte *Name* gültig ist. Er erlaubt es, verschiedenste Namen der selben Sache für unterschiedliche Zwecke frei zu definieren. Diese können z.B. Spitznamen, Login-Namen, Trivialnamen chemischer Verbindungen und vieles mehr sein.
Topics stellen das Grundgerüst der Modellierung von Informationen und Informationsobjekten in *Topic Maps* dar. Stehen einzelne Informationsobjekte in unmittelbarem Zusammenhang, so lässt sich dieser ebenfalls repräsentieren. Dies geschieht durch *Associations*, welche im nächsten Abschnitt erläutert werden.

Associations

Stehen also zwei (oder mehr) Aussagegegenstände in Beziehung zueinander, so lässt sich dies in Topic Maps mit Hilfe von *Associations* repräsentieren. *Associations* verbinden zwei oder mehr *Topics* miteinander und haben einen Typ (dieser ist wieder ein *Topic*), welcher die „Natur der Beziehung repräsentiert“ [GM08]. Jedes Ding, welches in Beziehung zu anderen Dingen steht, spielt dabei eine ganz bestimmte Rolle. So ist „Leipzig“ eine Stadt in „Deutschland“, was beide Dinge miteinander verbindet. „Leipzig“ spielt dabei die Rolle der Stadt und „Deutschland“ die Rolle des Landes. Dieser komplexe Zusammenhang wird in Topic Maps durch die Konstrukte *Association*, *Role* und *Topic* repräsentiert. *Topics* sind Rollenspieler in *Associations*.

Beispiel (s. Abb. 2.2)

Betrachtet man die Beziehung zwischen „Leipzig“ und „Ostwald“, so ist „Leipzig“ die Wirkungsstätte von „Ostwald“ und „Ostwald“ ist Wissenschaftler in Leipzig. In Topic Maps-Terminologie heißt dies: Das *Topic* „Leipzig“ spielt die *Role* der Stadt und das *Topic* „Ostwald“ spielt die *Role* des Wissenschaftlers in der *Association* Forschungsstätte. In Abb. 2.2 wird dieser Sachverhalt verdeutlicht. *Topics* (hier als Spieler oder Typen) werden als Kreise, die *Roles* als Dreieck und die *Association* als Viereck dargestellt.

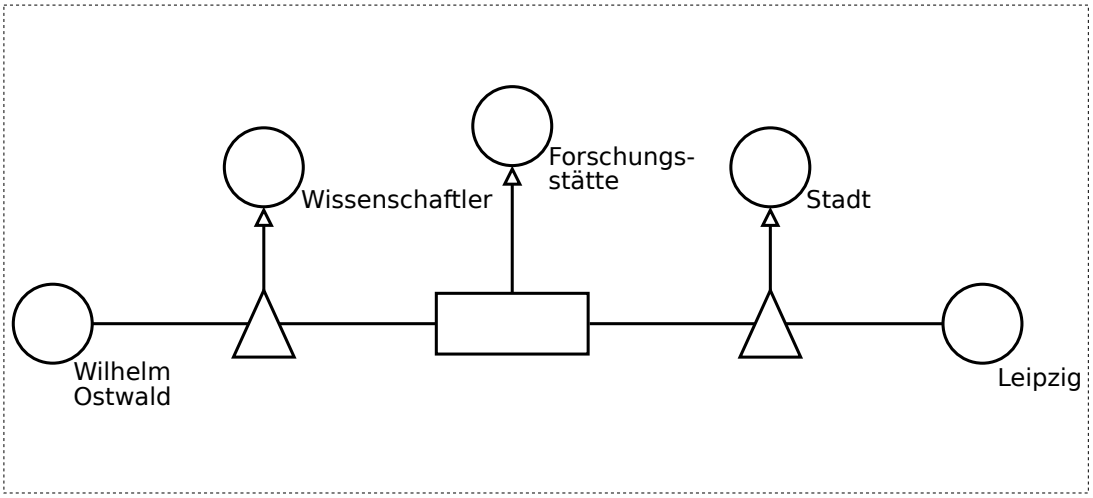

Abbildung 2.2.: Beispiel für die Repräsentation einer Beziehung durch eine *Association* mit *Roles* und Spielern

Occurrences

Wie oben erwähnt, sind *Topics* Stellvertreter für Dinge der realen Welt, welche in einer *Topic Map* repräsentiert werden sollen. Jedes solches „Ding“ hat neben Namen, Identität und Beziehungen zu anderen Dingen auch bestimmte Eigenschaften. Diese Eigenschaften müssen an sich nicht selbst wieder ein eigener Aussagegegenstand sein, sondern sind nur an diesen angehängt. Solche Eigenschaften werden in Topic Maps mit dem Konstrukt *Occurrence* modelliert.

Beispiel (s. Abb. 2.3)

Betrachtet man z.B. Leipzig als *Topic* in einer *Topic Map*, so könnten (je nach modellierter Domäne) Fläche oder Einwohnerzahl relevante Informationen sein. Für genau solche Informationen, welche Eigenschaften eines Aussagegegenstandes sind, bietet sich eine *Occurrence* als Konstrukt an.
Occurrences sind immer über die „Vater-Beziehung“ mit einem *Topic* verbunden und haben u.a. einen Typ, der die Art der Information und einen Wert, der das eigentliche Datum, das gespeichert werden soll, darstellt. Im Beispiel (s. Abb. 2.3) sind „Fläche“ und „Einwohnerzahl“ die Typen der beiden *Occurrences* und die entsprechenden Werte sind $297,36$ und 522.883. Die *Occurrence* mit dem Typ „Fläche“ hat hier zusätzlich noch einen Datentyp (welcher letztendlich eine URI ist, hier aber vereinfacht als km^2 dargestellt wird).

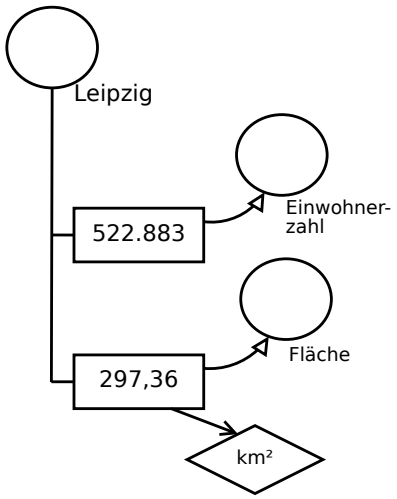

Abbildung 2.3.: Beispiel für ein *Topic* mit zwei *Occurrences*

2.3. Reifikation

Reifikation ermöglicht es Aussagen über Aussagen zu treffen.

> The act of reification is the act of making a topic represent the subject of another topic map construct in the same topic map. For example, creating a topic that represents the relationship represented by an association is reification. [GM08]

Es wird also ein *Topic* erstellt. Dieses *Topic* ist nun Stellvertreter für irgendein anderes *Topic Map* Konstrukt, beispielsweise eine *Association*. So könnten u.a. zusätzliche *Names* mit mehreren *Variants* mit der reifizierten *Association* in Verbindung gebracht werden. *Topics* können nicht reifiziert werden, da sie selbst schon repräsentiert sind und Aussagen über sie direkt durch das *Topic* an sich gemacht werden können. Weiterhin ist definiert, dass zwei identische *Topics* A und B nicht unterschiedliche Konstrukte reifizieren dürfen.

> It is an error if A and B both have non-null values in their [reified] properties which are different. [GM08]

Dies bedeutet, dass zwei *Topics*, die beide identisch nach Definition 3.7 sind, nicht unterschiedliche Konstrukte reifizieren dürfen. Dieser mögliche Fehler in der Modellierung wird an folgendem Beispiel klar:
Gegeben seien zwei *Associations*, die nicht identisch sind. Nun werden beide *Associations* von zwei *Topics* reifiziert, welche den selben *Subject Identifier* haben. Normalerweise sollten die beiden *Topics* von Anfang an zusammengefügt werden, da sie identisch sind. Beim Vergleich von zwei unabhängig entstanden *Topic Maps* oder etwa beim Hinzufügen eines neuen *Subject Identifiers* zu einem der beiden *Topics*, kann dieser Fehler jedoch immer noch auftreten. Diese beiden *Topics* stehen nun für ein und dasselbe Subject, in diesem Fall sind das die *Associations*. Sie repräsentieren ein und den selben Aussagegegenstand, sind aber dennoch nicht identisch.
Dieser logische Widerspruch kann nicht aufgelöst werden. Es ist aber herauszuheben, dass dies kein Fehler im Datenmodell an sich ist, sondern von fehlerhafter (also widersprüchlicher) Modellierung ausgegangen werden kann, sollte dieser Fall auftreten. Deshalb wird in den folgenden Ausführungen davon ausgegangen, dass die beschriebene Konstellation nicht auftritt. Sie wird also, wie im TMDM beschrieben, als Fehler behandelt.

2.4. Topic Maps als Graphen

Wie in Definition 3.8 festgelegt, ist eine *Topic Map* ein Tupel aus einer Menge *Topics* und einer Menge *Associations*. *Associations* stellen eine Beziehung zwischen ihren Spielern her. Diese Struktur erinnert an die mathematische Struktur der Hypergraphen [Wik11a]. Mehrere Knoten können hier über eine Hyperkante miteinander verbunden werden. Die Knoten sind dabei die *Topics* und die Hyperkanten sind die *Associations*, welche eine Menge von *Topics* miteinander verbindet. Unter diesen Voraussetzungen lassen sich *Topic Maps* in Graphen transformieren. Die Rücktransformation gelingt nicht, da bei der Transformation der *Topic Map* in den Graphen sehr viele Informationen verloren gehen (*Roles*, *scopes*, *Typen*, ...).
Alle Darstellungen von *Topic Maps* als Graphen, auch in dieser Arbeit, dienen ausschließlich zur Veranschaulichung eines Sachverhaltes, entbehren jedoch jeglicher Standardisierung. Darum werden hier die Betrachtungen auf die spezifizierten Mengen beschränkt und sich bekannter Sachverhalte aus der Mengentheorie bedient. So gibt es bereits Möglichkeiten Unterschiede und Gemeinsamkeiten zweier Mengen zu bestimmen (s. Abschn. 3.2.1). Voraussetzung dafür ist, dass die Elemente der Mengen eindeutig bestimmbar sind (z.B. das Element „2“ in der Menge der natürlichen Zahlen). Diese Voraussetzung ist für Topic Maps erfüllt, da jedes Element eine eindeutige Identität besitzt. (s. Abschn. 3.1.1)
An dieser Stelle sei erwähnt, dass die Darstellung von *Topic Maps* in graphischer Form durchaus praktischen Nutzen hat. Kleine *Topic Maps* können so leicht veranschaulicht werden, jedoch hilft eine Transformation an dieser Stelle nicht die anfängliche Problemstellung zu vereinfachen. Es wäre zusätzlicher Aufwand nötig, eine *Topic Map* in eine Graphenstruktur zu transformieren, was die Arbeit eher erschwert.

2.5. Gleichheit und Identität

Ob zwei Dinge gleich oder gar dasselbe sind, lässt sich allgemeingültig nur äußerst schwer beantworten. Mit dieser Frage beschäftigten sich schon viele Philosophen und Mathematiker (u.a. Hobbes [Hob49], Leibnitz [Wik11b], siehe auch [Sei11]). Leibnitz argumentiert, dass sich zwei Körper gleichen, wenn kein Unterschied zwischen ihnen besteht [Wik11b]. Betrachtet man zwei nebeneinander liegende Körper, ist so Gleichheit oder Ungleichheit nachweisbar. Jedoch birgt die Leibnitz'sche Definition ein Problem in sich. Was ist, wenn sich Eigenschaften eines Körpers über die Zeit hinweg ändern oder wenn sie für die Betrachtung irrelevant sind? Beispielsweise ist es u.U. (abhängig von der

modellierten Domäne) egal, welche Farbe ein Lieferwagen hat. Es sind für einen Routenplaner also alle gleich großen Lieferwagen identisch. Der Fahrer des Wagens muss jedoch wissen, welchen Wagen er zum Zielort zu fahren hat. Hobbes erkennt folgendermaßen:

> Wenn die Identität eines Gegenstandes in Frage steht, ist vielmehr der Name entscheidend, der ihm gegeben wurde. Es ist etwas anderes, zu fragen, ob Sokrates derselbe Mensch, und etwas anderes, ob er derselbe Körper bleibe; denn sein Körper kann als Greis nicht derselbe sein, wie er es als Kind war, schon der Größenunterschiede wegen. [Hob49]

Welches grundsätzliche Problem der Definition der Identität über den Namen existiert, wurde bereits in Abschn. 2.2.2 diskutiert.
Lässt sich die Identität einer Entität eindeutig bestimmen, so sind nur diejenigen Entitäten *gleich*, welche dieselbe Identität besitzen. In diesem Sinne unterliegt die Bedeutung des Begriffes Identität und die damit einhergehende Bedeutung der *Gleichheit* immer der Interpretation des Modellierers der Domäne. Zumindest kann nicht davon ausgegangen werden, dass sich ein Modellierer an eine festgelegte Definition hält. Deshalb sollte die Idee *Gleichheit* im Datenmodell so flexibel wie möglich umgesetzt werden.
Zwei *Topics* sind gleich, wenn sie dieselbe Identität besitzen. In diesem Sinne unterstützt das TMDM dabei jedwede Definition von *Gleichheit*, da der jeweilige Entwickler die Identität jedes *Topics* frei definieren kann. Dies erfolgt durch das Zuweisen eines Uniform Resource Identifier (URI) zu dem entsprechenden *Topic*.

> Ein Uniform Resource Identifier [...] ist ein Identifikator und besteht aus einer Zeichenfolge, die zur Identifizierung einer abstrakten oder physischen Ressource dient. URIs werden zur Bezeichnung von Ressourcen (wie Webseiten,[...]) im Internet und dort vor allem im WWW eingesetzt. [Wik11c]

Die Idee dahinter ist, dass die URI auf eine Quelle im Internet verweist, welche Informationen über den beschriebenen Aussagegegenstand enthält. Dies ist aber nicht zwingend notwendig. Es existieren auch mehrere fachspezifische [BIO] oder allgemeine Verzeichnisse [Net11], welche URIs für Dinge bereithalten.
Um einem *Topic* eine Identität zuzuweisen, die über eine URI definiert ist, gibt es die Menge der *Subject Identifier*, welche jedem *Topic* zugehörig ist. Wird die URI zu dieser Menge hinzugefügt, so ist die Identität des *Topics* festgelegt (mehr dazu in Abschn. 3.6).
Ein kleines Problem ergibt sich, wenn versucht wird das Dokument an sich, welches sich hinter der URI verbirgt, zu repräsentieren. Soll beispielsweise das Datum der Erstellung eines Wikipedia Artikels in der Wissensbasis abgebildet werden, so kann für das

Topic, welches den Wikipedia Artikel repräsentiert, nicht die URL des Artikels als *Subject Identifier* benutzt werden. Dies liegt an der im TMDM definierten Semantik der *Subject Identifier*. Die Ressource, welche durch den *Subject Identifier* referenziert ist, soll das Topic beschreiben und nicht das Topic selbst sein. Das Erstellungsdatum würde sich in diesem Fall also auf die Entität beziehen, welche durch den Wikipedia Artikel beschrieben ist und nicht auf den Artikel an sich. Abhilfe für dieses Problem schafft das TMDM durch die Menge der *Subject Locator*, welche mit dem *Topic* assoziiert ist. Wird eine URI dieser Menge hinzugefügt, so repräsentiert das *Topic* das Dokument an sich und nicht das Ding, welches durch das Dokument beschrieben wird.
Das TMDM liefert also eine Möglichkeit, jede Entität unabhängig von deren Eigenschaften abzubilden und deren Identität frei zu definieren. Konkret ist dadurch eine Trennung der Identität und Eigenschaften der Entität erfolgt. Diese Trennung erlaubt es nun strukturelle Zusammenhänge in der Wissensbasis unabhängig von einzelnen Eigenschaften der modellierten Entitäten zu untersuchen. Genauso können Unterschiede in den Eigenschaften bestimmt werden, ohne dabei die strukturellen Zusammenhänge betrachten zu müssen. All dies wird später genauer erläutert und erklärt werden.

2.6. Vergleich von RDF und Topic Maps

Topic Maps haben den Zweck, Wissen semantisch zu repräsentieren. Einen ähnlichen Ansatz zur Wissensrepräsentation ist das RDF-Datenmodell. Es soll hier keine ausführliche Einführung in dieses Datenmodell gegeben werden, da dies den Umfang der Arbeit übersteigt. Einen Einstieg in RDF bietet beispielsweise [Ver04] oder [Mil98]. Ein kurzer Vergleich zwischen den beiden Repräsentationsparadigmen bietet sich an, da viele Arbeiten im Bereich von RDF Denkansätze für die Lösung des vorliegenden Problems bieten.

2.6.1. Modellierung

In [Gar04] findet sich eine ausführliche Beschreibung der konzeptionellen und spezifischen Gemeinsamkeiten und Unterschiede von RDF und Topic Maps. Die folgenden Ausführungen sind zum Teil von dort entnommen. Beide Datenmodelle haben „Dinge“ als zentrale Aussagegegenstände. Diese „Dinge“ werden über „Symbole“ in der jeweiligen Wissensbasis repräsentiert.

> RDF and topic maps are both identity-based technologies. That is, the key concept in both is „symbols“ representing identifiable „things“, which statements can be made about. [Gar04]

Die genannten Symbole sind in RDF die RDF-Nodes (sie repräsentieren sogenannte Ressources) und in Topic Maps die *Topics* (sie repräsentieren sogenannte Subjects). Ressources und Subjects sind im Prinzip das Gleiche, nämlich ein Gegenstand, welcher modelliert werden soll. RDF erlaubt es nun sogenannte Statements über die jeweiligen Ressources zu erstellen. Statements setzen den Gegenstand der Betrachtung in einen Kontext, sind Träger von Information und Semantik. Sie bestehen aus der Ressource an sich, einer Property und einer zweiten Ressource bzw. einem Literal. Miller beschreibt das RDF-Datenmodell wie folgt:

> Resources have properties (attributes or characteristics). RDF defines a resource as any object that is uniquely identifiable by a Uniform Resource Identifier (URI). The properties associated with resources are identified by property types, and property types have corresponding values. Property types express the relationships of values associated with resources. In RDF, values may be atomic in nature (text strings, numbers, etc.) or other resources, which in turn may have their own properties. [Mil98]

Prinzipiell ähnelt dies dem TMDM. Wichtige Unterschiede sind, dass im TMDM Namen gesondert behandelt werden (in RDF sind sie ein Statement) und dass *Occurrences* mächtigere Konstrukte sind, als einfache RDF-Statements mit atomaren Werten. Sie haben einen Typ, einen Scope, einen Datentyp (Abschn. 3.4) und können direkt reifiziert (s. Seite 16) werden. RDF-Statements mit einer Ressource im Wert, können mit *Associations* in Topic Maps verglichen werden. In beiden Fällen werden zwei Ressources bzw. Subjects in Beziehung gesetzt. Beziehungen in RDF sind jedoch unidirektional und rollenfrei. Trotz der hier genannten Unterschiede ist es möglich, Topic Maps in RDF zu konvertieren und umgekehrt [Gar04, Gar08]. Ein wesentlicher Teilaspekt dieser Konvertierung ist die Feststellung der Identität einzelner Konstrukte.

2.6.2. Identität

Wie die Identität eines *Topics* definiert ist, wurde schon in Abschnitt 2.5 erörtert. RDF unterscheidet drei verschiedene Typen von Nodes, nämlich Literal Nodes, URI Nodes und blank Nodes. Ein Literal Node besteht aus einer atomaren Zeichenkette, welche auch dessen Identität definiert. Haben zwei URI Nodes dieselbe URI als Label, so werden sie

als identisch angesehen. Hierbei gibt es keine Unterscheidung zwischen dem Referenzieren von Ressourcen an sich oder der Objekte, welche durch die Ressource beschrieben werden. Die Identität von blank Nodes ist nur über Statements feststellbar, in denen sie verwendet werden. Auch hier ist prinzipiell ein Mapping auf Konstrukte im TMDM möglich, jedoch aufwändiger [Gar04].
Im folgenden Abschnitt wird noch einmal auf die Erfahrungen eingegangen, die beim Berechnen von Unterschieden in RDF gemacht wurden, um eine effiziente und praktische Methodik für Topic Maps zu entwickeln.

2.7. Bisherige Arbeiten

2.7.1. Methoden der Spezifikation von Unterschieden

Klein definiert eine Ontologie als „a specification of a conceptualization“ [KFKO02]. Auf Grundlage dieser Definition sind *Topic Maps* Ontologien. Eine zugrundeliegende Konzeptualisierung einer Domäne lässt sich mit Hilfe des TMDM einfach spezifizieren. Es existieren bereits einige Arbeiten zum Vergleich von Ontologien, welche mit Hilfe von RDF spezifiziert wurden [KFKO02, CD10, NK04]. Auch wenn zwischen RDF und Topic Maps konzeptionelle Unterschiede bestehen (s. Abschn. 2.6), bietet sich ein Blick auf die bisherigen Erfahrungen mit diesen Technologien an. So erläutert Klein, wie Veränderungen in Ontologien angegeben werden können. Er nennt drei grundsätzliche Ansätze:

1. Liste mit Änderungsoperationen („transformation specification“)
2. Liste mit Mappings („mapping“)
3. Liste mit (TMDM-)Konstrukten („replacement“)

Die erste Methode ist das Angeben einer Liste mit Änderungsoperationen, welche *Topic Map A* in *Topic Map B* überführen. Für Topic Maps gibt es bereits mehrere Anfragesprachen (TMQL [Bar07], tolog [Gar06]), die für dieses Verfahren in Frage kämen. So könnte das Fehlen eines Konstruktes in einer der *Topic Maps*, über ein Update-Statement in einer der Anfragesprachen symbolisiert werden. Die auf diese Art und Weise bestimmte Liste von Änderungsoperationen für jedes einzelne, fehlende bzw. zusätzliche Konstrukt könnte nun direkt angewendet werden, um A in B zu überführen. Jedoch verliert man durch das bloße Angeben von Änderungstransformationen strukturelle Zusammenhänge. So zieht das Löschen eines Topics auch das Löschen der entsprechenden *Association* nach sich, in denen es Spieler ist (man beachte die Definition der Identität von *Association* in Kapitel 3.1.1).

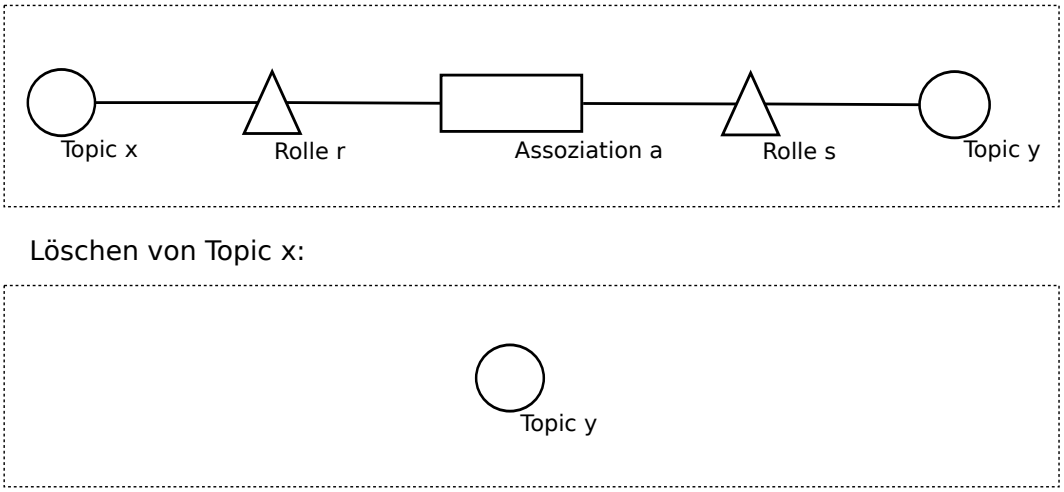

Abbildung 2.4.: Löschen des *Topics* x zieht das Löschen der *Associations*, in welchen es mitspielt, nach sich, da die *Association* ihre Identität verliert

TMQL bietet zwei mögliche Formen des Updates für solch eine Situation. Das Löschen der *Association* kann durch Kaskadierung forciert werden oder nicht erfolgen [Kro10]. Beide Möglichkeiten liefern kein befriedigendes Ergebnis. Wird kaskadiert, ist die Wissensbasis in einem konsistenten Zustand. Das Fehlen der *Association* ist jedoch nicht direkt erkennbar, sondern nur über Betrachtung aller Update-Statements und der ursprünglichen *Topic Map* rekonstruierbar. Wird beim Löschen des *Topics* nicht kaskadiert, so bleibt eine *Role* ohne den entsprechenden Player zurück. Die *Topic Map* ist an dieser Stelle nicht mehr konsistent mit dem TMDM. Des Weiteren ist es so nicht einfach möglich, Gemeinsamkeiten in den beiden *Topic Maps* anzugeben, da tatsächlich nur Unterschiede in einer Änderungsoperation reflektiert werden. Eine Erweiterung des Konzeptes der Änderungsoperationen (bspw. eine *keep*-Operation) würde dieses Problem lösen. Diese sind jedoch in den Query Standards (TMQL, tolog) nicht standardisiert oder vorgesehen.
Die zweite Möglichkeit ist, ein Mapping zwischen den unterschiedlichen *Topic Maps* anzugeben, welches gleiche Konstrukte miteinander verbindet und somit Gemeinsamkeiten und Unterschiede, anhand des Vorhandenseins bzw. Nicht-Vorhandenseins von Mappings identifiziert. Der Nachteil dieser Methode ist die zusätzlich benötigte Datenstruktur. Dies ist, wie die letzte Methode zeigt, nicht zwingend nötig und erfordert abermals eine Interpretation. Das Ergebnis muss anhand beider *Topic Maps* und des Mappings betrachtet werden, um die eingangs gestellten Fragen beantworten zu können. Die letzte Methode, das explizite Angeben der Elemente, welche gemeinsam oder nicht

gemeinsam in den jeweiligen *Topic Maps* auftreten, bietet sich als beste Lösung an, da die komplette Strukturinformation der Unterschiede und Gemeinsamkeiten beider *Topic Maps* erhalten bleibt. Es werden keine Zusatztechniken, wie Transformationssprachen (TMQL) oder eine Mappingstruktur benötigt. Außerdem ermöglicht das Merging-Modell von Topic Maps ein einfaches *Zusammenführen* von *Topic Maps*. Zusätzlich soll noch die *Differenz* und der *Vergleich* ermöglicht werden. Eine genaue Spezifikation und Beschreibung dieser Methoden findet in den folgenden Kapiteln statt.

2.7.2. Der Wandora Ansatz

Mit dem Wandora-Projekt [Kiv11] existiert bereits eine Implementierung, welche Unterschiede zwischen zwei *Topic Maps* berechnet. Die dort verwendete Methode zur Spezifikation der Unterschiede ist die erstgenannte der zuvor erwähnten Methoden. Unterschiede werden als Menge von Änderungen angegeben. Es existieren zwei nicht weiter spezifizierte Formate: *HTML* und *Patch*.
Wird das Format *HTML* gewählt, so ist das Ergebnis im Wesentlichen eine Liste von „Deleted“, „Changed“ und „Added“ Anweisungen, welche *Topic Map A* in *Topic Map B* überführt (Überführung wird hier nicht ausgeführt). Abgesehen von der eben schon erläuterten Kritik an dieser Methode fehlt hier zusätzlich eine formale Beschreibung des Formates. Eine Liste von TMQL-Statements wäre z.B. benutzbarer und frei von jeglicher Fehlinterpretation des Nutzers. Beispielsweise ist nicht klar, ob bei einer „Deleted“-Anweisung kaskadiert werden soll oder nicht.
Wird das Format *Patch* gewählt, wird eine Textdatei erstellt, welche Informationen enthält, die *Topic Map A* intern in *Topic Map B* transformieren kann. Diese Transformation kann auch rückgängig gemacht werden. Auch hier fehlen Informationen und Standardisierung, um das Tool praktisch nutzbar zu machen.
Außerdem tritt auch hier wieder das inhärente Problem auf, Gemeinsamkeiten nicht einfach erkennen zu können. Alles in allem fehlt es dem Wandora-Tool an einer definierten Spezifikation und Dokumentation. Es ist nicht modular (keine beschriebenen Interfaces vorhanden) aufgebaut und somit leider kaum nutzbar.

2.7.3. Syntaxbasierter Vergleich

An dieser Stelle sei auch noch ein Wort zu dateibasierten Diff-Tools, wie z.B. UNIX diff gesagt. In [KFKO02, CD10] sind die jeweils entwickelten Tools am UNIX diff orientiert. Um jedoch einen umfangreichen, strukturellen Vergleich zu erreichen, werden in [KFKO02] Heuristiken hinzugezogen, welche die Semantik der Unterschiede “erraten”

sollen. Die letzte Entscheidung, ob zwei Konzepte identisch sind, wird hier dem Nutzer überlassen. In [CD10] wird eine komplexe Erweiterung der Serialisierung durchgeführt, um die semantischen Implikationen der Änderung zu verstehen.
Die Berechnung von dateibasierten Unterschieden ist per se nicht semantisch. Schon die Frage: „Ist *Topic t* in beiden *Topic Maps* enthalten?“, kann durch ein dateibasiertes diff nicht befriedigend beantwortet werden. Beispielsweise impliziert das Wegfallen eines Namens des *Topics* in der zweiten *Topic Map* keinen strukturellen Unterschied, solange sie nach Datenmodell äquivalent sind. Diesen Unterschied kann ein dateibasiertes diff-Tool nur schwer erkennen und die Komplexität einer solchen Implementierung würde die eines Datenmodell-basierten Diffs, wie hier vorgestellt, schnell übersteigen.

3. Theorie

3.1. Formalisierung und Identität

In diesem Abschnitt soll erläutert werden, wie eine *Topic Map* formal repräsentiert werden kann. Auf Grundlage dieser Repräsentation werden später Operatoren definiert, mit Hilfe derer Unterschiede zwischen *Topic Maps* bestimmt werden können. Die Definition der hier genannten Konstrukte sind stark an den Spezifikationen des TMDM [GM08] angelehnt, worin sich auch noch weitere Informationen und Beschreibungen finden.
Die in Abschnitt 2.5 diskutierte Interpretation von Identität und Gleichheit gibt einen Einblick in die Komplexität der Begriffe im Bezug auf reale „Dinge“. Diese „Dinge“ werden nun als Konstrukte in der *Topic Map* repräsentiert. In diesem Kapitel geht es nun ausschließlich um die im TMDM definierte Identität der jeweiligen Konstrukte und nicht die der „Dinge“.
Es ist für jedes hier definierte Konstrukt eine Menge von Attributen angegeben, welche die Identität des Konstrukts definieren. Dass heißt, sollten zwei Konstrukte in den angegebenen Attributen übereinstimmen, so sind sie identisch. Das TMDM spezifiziert also konkret, was Identität der jeweiligen Konstrukte bedeutet. Es stehen auch hier, gerade im Anwendungskontext, frei wählbare Definitionen der Identität zur Debatte. So kritisiert Vatant,

> dass bei Topic Maps [...] die Identität zweier subjects(sic) auf sehr eingeschränkte Weise ermittelt wird. Er schlägt vor, statt eines einzigen identifiers(sic) (einer URI-Zeichenkette) allgemeiner identische Werte für eine bestimmte Untermenge der Eigenschaften zu verwenden [...]. [Sig04]

Diese Diskussion geht jedoch an der Zielsetzung der Arbeit vorbei und wird deshalb nicht weiter fortgeführt. Wie schon genannt, soll das TMDM und dessen Definition der Identiät als Grundlage der Arbeit dienen.

3.1.1. Topic Map Konstrukte

Die jeweiligen Konstrukte sind Tupel, deren Elemente angegeben werden. Es folgt die Angabe der Gleichheitsattribute. Diese definieren die Identität der Instanzen des jeweiligen Konstruktes. Stimmen die Attribute zweier Konstruktinstanzen überein, so sind sie identisch.
Im Anschluss an die Definition wird jeweils eine kurze Beschreibung der Konstrukte angegeben. Es wird beispielhaft erläutert, was mit Hilfe des definierten Konstruktes modellierbar ist.

Definition 3.1 (Variant)
Ein *Variant* v sei ein 5-Tupel, bestehend aus folgenden Elementen:

- $value$ - der Wert des *Variants* als Zeichenkette,
- $scope$ - Menge von *Topics*, welche den Kontext des *Variants* definiert,
- $reifier$ - ein *Topic*, welches der Reifier des *Variants* oder Null ist,
- $datatype$ - eine Zeichenkette, welche den Datentyp des *Variants* spezifiziert,
- $parent$ - der *Name*, zu welchem der *Variant* gehört.

Also: $v = (value, type, scope, reifier, datatype, parent)$
Gleichheitsattribute: $value, type, scope, datatype, parent$

Ein *Variant* ist eine Alternative eines gegebenen *Name*. Der *Name* ist zwar im Allgemeinen zu bevorzugen, jedoch kann über den *scope* ein Kontext definiert werden, in dem der genannte *Variant* vorzuziehen ist. Ein Beispiel hierfür wäre der *Name* für ein *Topic*, welches das Land Deutschland repräsentiert. Hat das *Topic* den *Name* “Germany”, so könnte der *value* eines *Variants* im *scope* { “deutsch” } “Deutschland” sein.

Definition 3.2 (Name)
Ein *Name* n sei ein 6-Tupel, bestehend aus folgenden Elementen:

- $value$ - der Wert des *Name* als Zeichenkette,
- $type$ - ein *Topic*, welches den Typ des *Name* definiert,
- $scope$ - Menge von *Topics*, welche den Kontext des *Name* definiert,
- $variants$ - Menge von *Variants* des *Name*,
- $reifier$ - ein *Topic*, welches der *Reifier* des *Name* oder Null ist,
- $parent$ - das *Topic*, zu welchem der *Name* gehört.

Also: $n = (value, type, scope, parent, variants, reifier)$
Gleichheitsattribute: $value, type, scope, parent$

Ein *Name* ist ein Bezeichner für ein *Topic*. Dieser muss nicht eindeutig sein, sollte aber so gewählt sein, dass man ihn zur Anzeige oder für den Diskurs verwenden kann. Die Verwendung des *Name* wurde schon in Kapitel 2 erläutert.

Definition 3.3 (Role)
Eine *Role* r sei ein 4-Tupel, bestehend aus folgenden Elementen:

- $type$ - ein *Topic*, welches den Typ der *Role* definiert,
- $player$ - das *Topic*, welches der Spieler der *Role* in der $parent$-*Association* ist,
- $parent$ - die *Association*, zu welcher die *Role* gehört,
- $reifier$ - ein *Topic*, welches der *Reifier* der *Role* oder Null ist.

Also: $r = (type, player, parent, reifier)$
Gleichheitsattribute: $player, type, parent$

Jede *Association* stellt eine Beziehung zwischen einer Menge von *Topics* her. Diese *Topics* sind dann nicht ohne weiteres Teil der *Association*, sondern sie „spielen eine Rolle“. Beispielsweise stehen „Vater“ und „Sohn“ in Beziehung zueinander. Diese Beziehung wird in Topic Maps über eine *Association* repräsentiert. Nun spielt hier der „Sohn“ die Rolle des Kindes und der „Vater“ die Rolle des Elternteils. Der Typ der entsprechenden *Role* ist jeweils ein *Topic* „Kind“ bzw. „Elternteil“.

Definition 3.4 (Occurrence)
Eine *Occurrence* o sei ein 6-Tupel, bestehend aus folgenden Elementen:

- $value$ - der Wert der *Occurrence*,
- $type$ - das *Topic*, welches den Typ der *Occurrence* definiert,
- $datatype$ - eine Zeichenkette, welche den Datentyp der *Occurrence* spezifiziert,
- $scope$ - Menge von *Topics*, welche den Kontext der *Occurrence* definiert,
- $parent$ - das *Topic*, zu welchem die *Occurrence* gehört,
- $reifier$ - ein *Topic*, welches der *Reifier* der *Occurrence* oder Null ist.

Also: $o = (value, type, datatype, scope, parent, reifier)$
Gleichheitsattribute: $value, type, datatype, scope, parent$

Occurrences wurden ausführlich in Kapitel 2 beschrieben.

Definition 3.5 (Association)
Eine *Association* a sei ein 4-Tupel, bestehend aus folgenden Elementen:

- $type$ - ein *Topic*, welches den Typ der *Association* definiert,
- $scope$ - Menge von *Topics*, welche den Kontext der *Association* definiert
- $roles$ - nicht leere Menge von *Topics*, welche die *Roles* der *Association* darstellen
- $reifier$ - ein *Topic*, welches der *Reifier* der *Association* oder Null ist.

Also: $a = (type, scope, roles, reifier)$
Gleichheitsattribute: $role, type, scope$

An dieser Stelle ist zu bemerken, dass die Identität der *Association* insbesondere von den *Roles* abhängt, deren Identität wiederum durch deren Player bestimmt ist. Betrachtet man also die Identität von *Associations*, so muss diese immer als "Ganzes" (mit *Roles* und Playern) betrachtet werden. Sollten sich zwei *Associations* nur um eine *Role* mit einem entsprechenden Player unterscheiden, so sind diese voneinander verschieden, auch wenn alle anderen Eigenschaften übereinstimmen.

Definition 3.6 (Topic)
Ein *Topic* t sei ein 7-Tupel, bestehend aus folgenden Elementen:

- SI - (*Subject Identifiers*) eine Menge von Zeichenketten, welche jeweils ein Identifikator des *Topics* ist,
- SL - (*Subject Locators*) eine Menge von Zeichenketten, welche jeweils auf eine Informationsressource zeigen, für die das *Topic* steht,
- II - (*Item Identifiers*) eine Menge von Zeichenketten, welche jeweils ein (interner) Identifikator des *Topics* ist,
- N - eine Menge von *Names* des *Topics*,
- V - eine Menge von *Variants* des *Topics*
- O - eine Menge von *Occurrences* des *Topics*,
- r - das Konstrukt, welches durch das *Topic* reifiziert wird oder Null.

Also: $t = (SI, SL, II, N, V, O, r)$

Topics wurden ausführlich in Abschn. 2.2.2 besprochen. Anzumerken ist, dass die *Variants* der *Names* vom betrachteten *Topic* Teil der Konstruktdefinition sind, da dadurch später die Definition der Operatoren vereinfacht wird. Dabei können keine Probleme mit der Identität der *Variants* auftreten, weil der *Name* als solches ein Gleichheitsattribut des *Variants* ist (s. Def. 3.1).

Die Gleichheitsdefinition der *Topics* unterscheidet sich ein wenig von der, anderer Konstrukte. *Topics* stellen das zentrale Modellierungsgrundgerüst von Topic Maps dar. Sie repräsentieren „Dinge“, über die Aussagen getroffen werden sollen. Deshalb ist die Definition der Identität der *Topics* von zentraler Bedeutung.

Definition 3.7 (Gleichheit von Topics)
Zwei *Topics* sind identisch, wenn sie in

- einer Zeichenkette der *Subject Identifier* oder
- einer Zeichenkette der *Subject Locator* oder
- einer Zeichenkette der *Item Identifier* oder
- einer Zeichenkette der *Subject Identifier* des einen und einer Zeichenkette der *Item Identifier* des anderen *Topics* übereinstimmen oder
- das Reified Attribut ein identisches Konstrukt aufweist.

Schlussendlich ist eine *Topic Map* dann eine Art Container, welcher alle modellierten Informationen beinhaltet:

Definition 3.8 (Topic Map)
Eine *Topic Map* $m = (T, A)$ sei ein Tupel aus einer Menge von *Topics* T und *Associations* A, wobei $T = \{t_1, t_2, ..., t_n\}$ und $A = \{a_1, a_2, ..., a_m\}$

3.2. Operatoren

3.2.1. Einführung in die Mengentheorie

Definition 3.8 besagt, dass eine *Topic Map* ein Tupel zweier Mengen (*Topics* und *Associations*) ist. Das Problem Unterschiede in *Topic Maps* zu erkennen, kann (später genauer erläutert) auf das Problem reduziert werden, Unterschiede in diesen gegebenen Mengen zu finden. Deshalb werden im Folgenden die Grundlagen der Mengentheorie und deren Operatoren dargestellt.

> **Definition 3.9** (Menge (nach Cantor)[Hau65])
> Unter einer „Menge" verstehen wir jede Zusammenfassung M von bestimmten wohlunterschiedenen Objekten m unserer Anschauung oder unseres Denkens (welche die „Elemente" von M genannt werden) zu einem Ganzen.

Hier wird die Wichtigkeit der im vorigen Abschnitt vorgestellten Definitionen der Gleichheit von *Topics* (Def. 3.7) deutlich. Sollten Objekte nicht unterscheidbar sein, so können sie nicht als verschiedene Objekte betrachtet werden. Identität liefert also Wohlunterscheidbarkeit der Konstruktinstanzen und damit die Möglichkeit Mengenbetrachtungen für Topic Maps anzustellen.

Die im Folgenden definierten Operatoren arbeiten auf Mengen. Das Ergebnis der Operationen enthält Elemente, deren Zugehörigkeit zu den Ursprungsmengen anhand des Vorhandenseins in der Ergebnismenge der Operation herausgefunden werden kann. Die Elemente können in beiden (Schnitt), in mindestens einer der beiden (Vereinigung) oder exklusiv in einer der beiden (Differenz) Mengen vorhanden sein. Dies sind genau die Aussagen, die auch für die Konstrukte zweier *Topic Maps* getroffen werden sollen.

> **Definition 3.10** (Vereinigung [Hau73])
> Vereinigungsmenge $M \cup N$ zweier Mengen M, N heißt eine Menge genau dann, wenn sie aus allen Elementen besteht, die der Menge M *oder* der Menge N angehören.

Die Vereinigung zweier Mengen ist eine Menge, welche alle Elemente enthält, die Teil einer oder beider Ausgangsmengen sind. Sollte ein Element in beiden Mengen vorkommen, so ist es auch Teil des Ergebnisses. Es ist zu beachten, dass dieses Element nur einmal in der Ergebnismenge vorkommt, denn identische Elemente sind nicht wohlunterscheidbar und können auch nicht mehrmals Elemente einer Menge sein.

Definition 3.11 (Schnitt [Hau73])
Der Durchschnitt [bzw. Schnitt] $M \cap N$ zweier Mengen M und N besteht aus allen Elementen, die *sowohl* M *als auch* N angehören.

Definition 3.12 (Differenz [Hau73])
Diese Menge [(Ergebnis der *Differenz*)] $M \setminus N$ besteht aus genau den Elementen von M, die übrig bleiben, wenn man diejenigen Elemente aus M entfernt, die auch Elemente von N sind. Eine derartige Menge, auf die dies zutrifft, heißt **Differenzmenge** $M \setminus N$ der Mengen M und N.

Durch die gegebenen Operatoren für Mengen kann also die Zugehörigkeit der Elemente zu den Ausgangsmengen festgestellt werden. Diese Zugehörigkeitsfeststellung ist das, was für Topic Maps die Eingangsfragestellung beantworten kann. Deshalb werden nun Operatoren definiert, welche an die obigen Mengenoperatoren angelehnt sind.

3.2.2. Definition der Operatoren

Bei der Betrachtung der Unterschiede von Wissensbasen kristallisieren sich zwei elementar verschiedene Formen der Unterschiedlichkeit heraus. Zum einen gibt es Unterschiede struktureller Natur und zum anderen können sich als gleich identifizierte Objekte, in ihren Eigenschaften unterscheiden. Im Kontext Topic Maps bezieht sich der erste Fall auf das tatsächliche Vorhandensein von *Topics* oder *Associations* in den beiden untersuchten *Topic Maps*. Im zweiten Fall werden *Topics*, welche als identisch identifiziert wurden, auf Unterschiede in ihren Attributen (*Names*, *Occurrences*, *Identifiers*) hin geprüft. Es entstehen also zwei verschiedene Abstraktionsniveaus oder Betrachtungsebenen, nämlich die strukturelle Ebene und die Eigenschaftsebene, auf welcher die jeweiligen Unterschiede zu betrachten sind.
Für die strukturelle Ebene werden nun Operatoren definiert, mit denen sich ein mengentheoretischer Zugang zu den Unterschieden und Gemeinsamkeiten von *Topic Maps* eröffnet. Zu den im vorigen Kapitel definierten Operatoren für Mengen sollen nun also ähnliche Operatoren auch für Topic Maps definiert und nutzbar gemacht werden. Wie

schon die Unterschiede der *Topic Maps* auf unterschiedlichen Abstraktionsniveaus zu betrachten sind, so sind auch die Operatoren für unterschiedliche Abstraktionsniveaus zu definieren. Die erste Gruppe, angewendet auf ganze *Topic Maps*, gibt einen Überblick über Unterschiede auf struktureller Ebene, während die zweite Gruppe auf *Topics* arbeitet. Warum diese Trennung nötig ist, wird im Folgenden erläutert.

Strukturelle Ebene

Die nachfolgend definierten Operatoren arbeiten auf Topic Maps und erzeugen Mengen von *Topics* und *Associations*. Durch die hohe Komplexität des TMDM ist es notwendig jeweils einzelne Operatoren für *Topics* und *Associations* zu definieren. Dies soll an folgendem Beispiel erläutert werden:

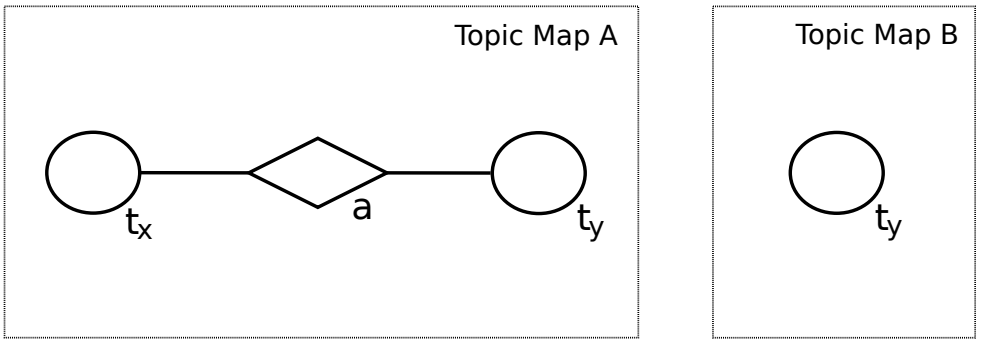

Abbildung 3.1.: Beispiel für zwei einfache *Topic Maps*

Versucht man ganz Intuitiv die (an der Mengentheorie orientierte) Differenz der in Abbildung 3.1 dargestellten *Topic Maps*, mit den *Topics* t_x, t_y und der *Association* a als Elemente, zu bilden, so sind wohl die *Association* a und das *Topic* t_y Elemente einer Ergebnismenge $A \setminus B$, da sie Teil der ersten, aber nicht Teil der zweiten *Topic Map* sind (s. Def. 3.12).
Betrachtet man nun die Definition der Identität von *Associations* aus Abschnitt 3.1.1, so stellt man fest, dass diese untrennbar an deren *Roles* und damit auch untrennbar an deren *Role-Players* gebunden ist. Wird also der Spieler einer *Association* entfernt, so muss auch die entsprechende *Role* entfernt werden, was wiederum dazu führt, dass die Identität der *Association* verändert wird. Somit ist das *Topic* t_x aus *Associations* - Sicht zwingend Element der Differenz $A \setminus B$. Ist das Topic t_x nun Element der Differenz bedeutet dies jedoch, dass t_x (mengentheoretisch gesprochen) Element von A, jedoch nicht von B ist. An diesem einfachen Beispiel wird deutlich, dass einfache Operatoren

der Art $A \backslash B$ für Topic Maps A und B nicht genügen, um alle nötigen Informationen über die strukturellen Unterschiede zweier *Topic Maps* abzubilden. Die Lösung des Problems ist eine Aufsplittung des Operators in einen Operator für *Topics* und einen Operator für *Associations*. Dabei liefern diese Operatoren eine Menge von Konstrukten, welche alle nötigen strukturellen Informationen enthalten.

Definition 3.13 ($\backslash_t$ - „strukturelle Topic-Differenz“)
Die strukturelle Topic-Differenz einer *Topic Map* A mit den *Topics* t_i und einer *Topic Map* B mit den *Topics* t_j ist genau die Menge von *Topics*, welche folgende Bedingung erfüllen:
$t_i \in A$ und $t_i \notin B$

Definition 3.14 ($\backslash_a$ - „strukturelle Association-Differenz“)
Die strukturelle Association-Differenz einer *Topic Map* A mit den *Associations* a_i und einer *Topic Map* B mit den *Associations* a_j ist genau die Menge von *Associations*, welche folgende Bedingung erfüllen:
$a_i \in A$ und $a_i \notin B$

Definition 3.15 ($\cap_t$ - „struktureller Topic-Schnitt“)
Der strukturelle Topic-Schnitt einer *Topic Map* A mit den *Topics* t_i und einer *Topic Map* B mit den *Topics* t_j ist genau die Menge von *Topics*, welche folgende Bedingung erfüllen:
$t_i \in A$ und $t_i \in B$

Definition 3.16 ($\cap_a$ - „struktureller Association-Schnitt“)
Der strukturelle Association-Schnitt einer *Topic Map* A mit den *Associations* a_i und einer *Topic Map* B mit den *Associations* a_j ist genau die Menge von *Associations*, welche folgende Bedingung erfüllen:
$a_i \in A$ und $a_i \in B$

Definition 3.17 ($\cup$ - „Merge“)
Der Merge einer *Topic Map* A in eine *Topic Map* B entspricht den im [GM08] definierten Merge aller mergbaren *Associations* und *Topics* der jeweiligen *Topic Maps*.

An dieser Stelle ist zu beachten, dass das Ergebnis dieser Merge-Operation eine nicht nur strukturell zu den Ausgangs-Maps verschiedene Topic Map ist, sondern auch Eigenschaften der Topics zusammengeführt werden. Diese Operation stellt somit eine nicht rein strukturelle Operation dar. Im Folgenden werden nun die Operatoren der Eigenschaftsebene definiert.

Eigenschaftsebene

Im vorangegangenen Abschnitt wurden die Unterschiede von *Topic Maps* auf struktureller Ebene betrachtet. Nun soll untersucht werden, ob sich einzelne Eigenschaften (*Occurrences*, *Names*, ...) von strukturell äquivalenten *Topics*, unterscheiden. Ob zwei *Topics* auf struktureller Ebene identisch sind oder nicht, ist im TMDM definiert. Wenn zwei *Topic Maps* A und B mit *Topics* t_i^A und t_j^B gegeben sind, so ist der strukturelle Topic-Schnitt $A \cap_t B$ (Def. 3.15) eine Menge von *Topics*, für die gilt: $t_i^A \in A, t_j^B \in B$ und $t_i^A = t_j^B$ (nach [GM08]). Für genau diese *Topics* t_i^A und t_j^B gilt es nun zu überprüfen, welche Gemeinsamkeiten und Unterschiede auf Eigenschaftsebene bestehen.
Es sei erwähnt, dass *Topics*, welche nicht als strukturell gleich identifiziert wurden, nicht weiter beachtet werden. Dass heißt, sollten zwei nicht identische *Topics* jeweils einen gleichen *Name* haben, sind diese von vornherein, unabhängig von deren *value*, per Definition der Identität für *Names* ungleich. Der *parent* des *Name* ist nämlich ein Gleichheitsattribut (s. 3.2), somit wäre der Vergleich zweier strukturell verschiedener *Topics* immer ein *Topic* ohne Attribute. Die hier genannten Operatoren sind also für strukturell äquivalente *Topics* definiert:

Vorbetrachtung

Für ein Topic t^A sei:

- SI^A - die Menge aller *Subject Identifiers* $\{si_1^A, ..., si_n^A\}$
- SL^A - die Menge aller *Subject Locators* $\{sl_1^A, ..., sl_m^A\}$
- II^A - die Menge aller *Item Identifiers* $\{ii_1^A, ..., ii_i^A\}$
- N^A - die Menge aller *Names* $\{n_1^A, ..., n_j^A\}$
- O^A - die Menge aller *Occurrences* $\{o_1, ..., o_k\}$
- V_i^A - die Menge der Variants $\{v_1^A, ..., v_l^A\}$ aller Names n_i^A aus N^A
- r^A - das reifizierte Konstrukt

Topic Differenz

Definition 3.18 (− - „Topic-Differenz“)
Die Topic-Differenz $t^A - t^B$ zweier semantisch äquivalenter *Topics* t^A und t^B sei ein Tupel f genannt *Topic* Fragment, mit den folgenden Elementen:
$f = (SI, SL, II, N, V, O, r)$, wobei:

- $SI := \{si_j \mid si_j \in SI^A \text{ und } si_j \notin SI^B\}$
- $SL := \{sl_j \mid sl_j \in SL^A \text{ und } sl_j \notin SL^B\}$
- $II := \{ii_j \mid ii_j \in II^A \text{ und } ii_j \notin II^B\}$
- $N := \{n_j \mid n_j \in N^A \text{ und } n_j \notin N^B\}$
- $V := \{v_j \mid v_j \in V_i^A \text{ und } v_j \notin V_i^B\}$
- $O := \{o_j \mid o_j \in O^A \text{ und } o_j \notin O^B\}$
- $r := \begin{cases} r^A, & \text{wenn } r^A \neq r^B \\ \text{null}, & \text{sonst} \end{cases}$

Das Ergebnis der Topic-Differenz ist ein *Topic* Fragment. Es enthält die in der obigen Definition genannten Elemente, also alle *Subject Identifier*, *Subject Locator* und *Item Identifier*, welche exklusiv, Eigenschaften des ersten *Topics* sind. Identifier sind letztendlich URIs, also kann über einen Zeichenkettenvergleich Äquivalenz und somit die Zugehörigkeit oder Nicht-Zugehörigkeit zum Ergebnis-Fragment festgestellt werden.
Occurrences werden ebenso wie Identifier behandelt. Sind sie Eigenschaft beider *Topics*, so sind sie nicht Teil des Ergebnisses. Sind sie ausschließlich Teil des ersten *Topics*, werden sie in das Ergebnis-Fragment aufgenommen.
Die Feststellung der Gleich- oder Ungleichheit von *Occurrences* beschränkt sich jedoch nicht auf einen Zeichenkettenvergleich, wie dies bei den Identifiern der Fall war. Hierzu muss die Definition der Occurrences (Def. 3.4) und deren Gleichheitsattribute betrachtet werden. Nach dieser Definition müssen die Attribute $value, type, datatype, scope, parent$ auf Gleichheit überprüft werden. $Value$ und *datatype* sind abermals Zeichenketten, *type* ein *Topic* und *scope* eine Menge von *Topics*. *Parent* ist jeweils das *Topic*, zu welchem die *Occurrences* gehören, diese sind per Definition gleich.
Für *Names* wird aufgrund der Attribute $value, type, scope, parent$ Gleichheit festgestellt. Auch hier werden nur die *Names* zum Ergebnis hinzugefügt, welche ausschließlich zum ersten *Topic* gehören. Die *Variants* der *Names* werden ähnlich zu den schon betrachteten Konstrukten behandelt. Es wird die Identität der einzelnen *Variants* geprüft und Übereinstimmungen werden identifiziert. Da die Definition der Gleichheit von

Variants (Def. 3.1) besagt, dass u.a. die *parents* der *Variants* (also deren *Names*) identisch sein müssen, ist dies ebenfalls nachzuweisen. Wird Nicht-Gleichheit oder Gleichheit festgestellt, so wird analog zu den anderen Konstrukten das *Variant* zum Ergebnis-Fragment hinzugefügt oder nicht hinzugefügt.

Beispiel

Die gerade formulierte Definition 3.18 soll nun noch einmal am Beispiel erläutert werden. (Für die Notation beachte man die Vorbetrachtung zu Beginn des Kapitels, gleiche Indizes bedeuten hier Äquivalenz der entsprechenden Konstrukte)

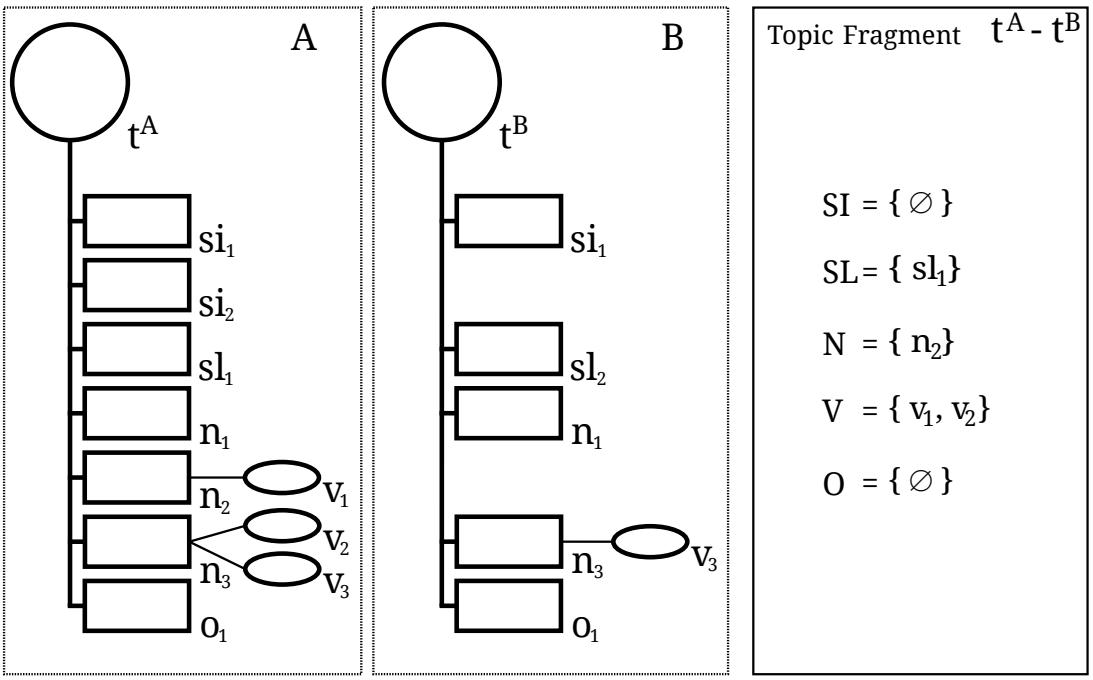

Abbildung 3.2.: Zwei Topics t^A und t^B und deren Differenz

In Abbildung 3.2 sehen wir zwei *Topics* und deren Differenz $t^A - t^B$. Voraussetzung für die Differenzbildung ist, dass beide *Topics* äquivalent sind. Definition 3.6 besagt, dass zwei *Topics* äquivalent sind, sollten sie in einem *Subject Identifier* übereinstimmen. Dies ist hier der Fall und somit darf die Topic-Differenz gebildet werden. Das Ergebnis-Fragment enthält nun alle relevanten Informationen, welche Unterschiede zwischen den beiden *Topics* bestehen.

Alle Identifier, *Names*, *Variants* und *Occurrences*, welche ausschließlich in *Topic* t^A vorhanden sind, finden sich nun im angegebenen *Topic*-Fragment wieder.
Es folgt die Definition des Topic-Schnittes. Die beiden betrachteten Topics müssen wieder strukturell äquivalent sein. Es gilt weiterhin die Vorbetrachtung vom Beginn dieses Abschnittes (Seite 34).

Topic Schnitt

Definition 3.19 ($\wedge$ - „Topic-Schnitt“)
Der Topic-Schnitt $t^A \wedge t^B$ zweier semantisch äquivalenter *Topics* t^A und t^B sei ein Tupel f genannt *Topic* Fragment, mit den folgenden Elementen:
$f = (SI, SL, II, N, V, O, r)$, wobei:

1. $SI := \{si_j \mid si_j \in SI^A \text{ und } si_j \in SI^B\}$
2. $SL := \{sl_j \mid sl_j \in SL^A \text{ und } sl_j \in SL^B\}$
3. $II := \{ii_j \mid ii_j \in II^A \text{ und } ii_j \in II^B\}$
4. $N := \{n_i \mid n_i \in N^A \text{ und } n_i \in N^B\}$
5. $V := \{v_j \mid v_j \in V_i^A \text{ und } v_j \in V_i^B\}$
6. $O := \{o_j \mid o_j \in O^A \text{ und } o_j \in O^B\}$
7. $r := \begin{cases} r^A, & \text{wenn } r^A = r^B \\ \text{null}, & \text{sonst} \end{cases}$

Intuitiv heißt das, dass alle Informationen, die beide *Topics* gemeinsam enthalten, sprich Eigenschaften von t^A *UND* t^B (deren Schnitt), im Ergebnis-Fragment enthalten sind. Es ist die Nähe zu den mengentheoretischen Operatoren zu erkennen.
Das entsprechende *ODER* der Eigenschaften des *Topics* (also deren Vereinigung) wird durch ein Mergen der beiden *Topics* erreicht. Dieser Vorgang ist ausführlich im TMDM beschrieben (s. Def. 3.17).

3.2.3. Eigenschaften der Operatoren

Genau wie die mengentheoretischen Operatoren haben die vorgestellten Topic Maps Operatoren bestimmte Eigenschaften. Besonders wichtig ist die Kommutativität der Operatoren, da besonders hier Fehler in der Handhabung der vorgestellten Verfahren auftreten können. Kommutativität bedeutet Vertauschbarkeit der Operanden miteinander. Aus der Mengentheorie ist bekannt:
$A \cap B = B \cap A$ und
$A \cup B = B \cup A$

Letztendlich agieren die vorgestellten Operatoren auf Mengen, wobei jedes Element der Menge (wie für Mengen üblich) eine definierte Identität besitzt. Der strukturelle Topic Schnitt ($A \cap_t B$) zweier *Topic Maps* ist ein Operator, welcher auf der Menge der *Topics* der *Topic Maps* A und B operiert. Die *Topic* Tupel sind dabei die Elemente der Menge. Definition 3.15 gleicht der entsprechenden Mengenoperation. Der Beweis der Kommutativität des Topic-Schnitt Operators lässt sich also auf den Beweis der Kommutativität der Schnitt-Operatoren für Mengen zurückführen und ist somit trivial. Analog gilt dies natürlich auch für den Association-Schnitt Operator.
Die Differenz-Operatoren verhalten sich ebenfalls analog zu ihren mengentheoretischen Entsprechungen. Die Argumentation verläuft äquivalent zur vorangegangenen. Die mengentheoretische Differenz stellt eine nicht kommutative Operation dar, sprich:
$A \setminus B \neq B \setminus A$.
Dies gilt auch für die strukturelle Topic - und Association - Differenz.
Kommutativität ist auch auf Eigenschaftsebene gegeben. Ein Blick auf die Definitionen 3.6 und 3.19 zeigt, wie auch hier auf einfache Weise Kommutativität nachgewiesen werden kann. Das *Topic* Tupel besteht aus Mengen und dem Reifier. Für die Mengen sind die Schnittdefinitionen abermals analog zu der entsprechenden mengentheoretischen Operation festgelegt. Der Reifier wird beim Schnitt der beiden *Topic Maps* nach den im TMDM zum Mergen festgelegten Regeln behandelt. Dass heißt, ist eines der beiden Reified Konstrukte leer, so wird das andere genutzt. Ist das Element bei beiden leer, so bleibt es auch so. Dass jeweils andere Konstrukte reifiziert werden, ist per Definition ausgeschlossen (s. Seite 16). Dieser Fall sollte dann bei einer technischen Umsetzung gesondert behandelt werden. Ansonsten ist auch hier zu erkennen, dass der Operator kommutativ ist, also die *Topics* nach Belieben vertauscht werden können. Für die Vereinigung (also den „Merge“) der *Topics* gilt natürlich dasselbe mit analoger Begründung. Dass die Topic Differenz nicht kommutativ ist, wird anhand der vorangegangen Begründungen sehr einfach deutlich und ist deshalb hier nicht noch einmal ausführlich erörtert.
Ebenfalls interessant ist die Feststellung der Assoziativität der Operatoren. Sind Operatoren assoziativ, so kann eine Kette von Operationen in beliebiger Reihenfolge durchgeführt werden. Für den assoziativen Operator $\circ$ gilt also:
$(A \circ B) \circ C \Leftrightarrow A \circ B \circ C \Leftrightarrow A \circ (B \circ C)$

Auch hier sei wieder ein Blick auf die mengentheoretischen Geschwister der Operatoren vorweggenommen. Für Mengen A, B und C gelten folgende bekannte Gesetzmäßigkeiten:

- $(A \cup B) \cup C = A \cup B \cup C = A \cup (B \cup C)$,
- $(A \cap B) \cap C = A \cap B \cap C = A \cap (B \cap C)$,
- $(A \setminus B) \setminus C = A \setminus (B \cup C)$ und
- $A \setminus (B \setminus C) = (A \setminus B) \cup (A \cap C)$. [Wik12c]

Schnitt und Vereinigung sind also assoziativ, die Differenz jedoch nicht. Die Topic Maps Operatoren verhalten sich erneut genauso, wie ihre mengentheoretischen Entsprechungen. Dies folgt direkt aus den oben vorgestellten Erläuterungen und den Operatorgesetzmäßigkeiten für Mengen und wird hier nicht noch einmal im Einzelnen erörtert. Allgemein kann nun festgehalten werden, dass sich die Topic Maps Operatoren äquivalent zu ihren entsprechenden mengentheoretischen Operatoren verhalten. Sie sind also intuitiv verwendbar und verhalten sich erwartungsgemäß.

3.2.4. Differenz von Topic Maps

Die Topic Map Differenz lässt sich nun mit Hilfe der vorangegangen Betrachtungen, leicht definieren:

Definition 3.20 ($\setminus$ - Topic Map Differenz)
Die Topic Map Differenz zweier *Topic Maps* $A = (T_a, A_a)$ und $B = (T_b, A_b)$ ist eine *Topic Map* $M = (T, A)$, für die gilt:

- $T = \{T_a \setminus_t T_b\}$
- $A = \{A_a \setminus_a A_b\}$

Die auf *Topic Maps* definierte Differenz kumuliert nun die zuvor erfolgten Betrachtungen auf eine zentrale Operation, welche dann eine Übersicht über die strukturellen Unterschiede zweier *Topic Maps* liefert. Atomare Unterschiede sind hier nicht zu erwarten, da alle *Topics* und *Associations* im Ergebnis grundsätzlich Subjects modellieren, welche ausschließlich in *Topic Map A* vorkommen. Es ist nun also möglich, strukturelle Unterschiede von *Topic Maps* zu erkennen, jedoch ist eine zentrale Frage noch nicht geklärt. Was bedeutet es, wenn zwei *Topic Maps* gleich sind? Der nächste Abschnitt beantwortet diese Frage und es werden die vorgestellten Verfahren und deren Zusammenhänge an einem praktischen Beispiel verdeutlicht.

3.2.5. Gleichheit von Topic Maps

Nun wird die Frage nach der Gleichheit von zwei *Topic Maps* noch einmal aufgegriffen. Zu Beginn der Arbeit wurden verschiedene Ideen für die Gleichheitsbestimmung, wie z.B. das Untersuchen von kanonisch serialisierten *Topic Maps* mit Hilfe von Text-basierten Differenztools (z.B. UNIX-Diff) vorgestellt. Die Nachteile dieser Lösungen wurden ebenfalls erörtert. Durch die in der Arbeit definierten Operatoren wird die Untersuchung der Gleichheit von Topic Maps auf verschiedenen Ebenen möglich. Die Operatoren der strukturellen Ebene liefern Aussagen über das Vorhandensein von *Topics* und *Associations*. Zwei *Topic Maps*, welche dieselben Subjects und deren Verknüpfungen modellieren, können durchaus gleich sein, obwohl sich Einzelheiten (z.B. das Vorhandensein eines zweiten *Names* eines *Topics*) unterscheiden. Diese Form der Gleichheit sei **semantische Gleichheit** genannt. Sie ist folgendermaßen definiert:

> **Definition 3.21** ($=_s$ - „semantische Gleichheit“)
> Für zwei *Topic Maps* $A = (T_a, A_a)$ und $B = (T_b, A_b)$ und die leere *Topic Map* $E = (\emptyset, \emptyset)$ sind A und B semantisch gleich genau dann, wenn das Ergebnis von deren Topic Map Differenz die leere *Topic Map* ist:
> $A =_s B \Leftrightarrow A \setminus_m B = B \setminus_m A = E$

Sind zwei *Topic Maps* semantisch gleich, so sind in ihnen also dieselben (identischen) *Topics* und *Associations* vorhanden. Diese Form der Gleichheit ist auch in der Mengentheorie kein unbekannter Begriff. Die Nähe der Topic Maps Operatoren zu denen der Mengentheorie ist auch hier wieder sehr praktisch und erleichtert das Verständnis. Der Nachweis über die Gleichheit zweier Mengen A und B erfolgt klassisch auf folgende Art und Weise:
$A = B \Leftrightarrow (A \subseteq B) \wedge (B \subseteq A)$
Mit anderen Worten heißt das, dass A gleich B ist, genau dann, wenn alle Elemente von A in B enthalten sind und alle Elemente aus B in A enthalten sind, sprich wenn die Differenz der Mengen A und B und der Mengen B und A leer ist. Dass heißt, es gibt keine Elemente in A bzw. B, welche nicht Element von B bzw. A sind. Also:
$A = B \Leftrightarrow A \setminus B = B \setminus A = \emptyset$
Hier ist nun Definition 3.21 eindeutig wiederzuerkennen.

Beispiel

Man stelle sich zwei verschiedene Fußballfans derselben Mannschaft vor, die Daten über Spieler und Taktiken sammeln. Um das Beispiel einfach zu halten, modelliert jeder der beiden Fans jeden der Stammspieler und die Spielerpositionen als *Topic* und erstellt *Associations* zwischen dem jeweiligen Spieler und der Position. Da beide Fans dieselben URIs als *Subject Identifier* nutzen (Wikipedia Seiten der Spieler und Positionen), lässt sich jeweils feststellen, welche *Topics* identisch sind (den selben Spieler repräsentieren). Haben die *Topics* nun auch noch die selben Eigenschaften, also die selben *Names*, *Variants* und *Occurrences*, so sind die beiden *Topic Maps* vollständig gleich. Dies bedeutet, dass in keiner der beiden *Topic Maps* mehr oder weniger Informationen, wie z.B. die Größe oder das Gewicht von Spielern oder die durchschnittliche Anzahl an Toren, die auf einer Position geschossen werden oder aber auch Informationen wie Spitznamen o.ä., zu finden sind. Das Zusammenführen der beiden Wissensbasen bringt also keinen Mehrwert gegenüber einer einzelnen existierenden *Topic Map*.
Sind *Topic Maps* somit strukturell gleich und haben deren *Topics* dieselben Attribute, so sind sie „vollständig gleich“

Definition 3.22 ($=_v$ - vollständige Gleichheit)
Zwei *Topic Maps* A mit den *Topics* $t_a = (SI_a, SL_a, II_a, N_a, V_a, O_a)$ und B mit den *Topics* $t_b = (SI_b, SL_b, II_b, N_b, V_b, O_b)$ sind vollständig gleich, also $A =_v B$ genau dann, wenn:

- $A =_s B$
- $\forall t_a, t_b \mid t_a = t_b$ gilt:
  $SI_a = SI_b$
  $SL_a = SL_b$
  $II_a = II_b$
  $N_a = N_b$
  $V_a = V_b$
  $O_a = O_b$

Die eingangs erwähnten Ebenen der Gleichheit beziehen sich auf die hier definierte semantische und vollständige Gleichheit. Auf der nun geschaffenen Grundlage lassen sich qualitativ unterschiedliche Aussagen über die Gleichheit von *Topic Maps* treffen. Einerseits kann völlig abstrahiert von einzelnen Eigenschaften der *Topics* semantische Gleichheit bestimmt werden, andererseits besagt die vollständige Gleichheit, dass alle Informationen, die in der einen *Topic Map* vorhanden sind, auch in der zweiten zu finden sind.

Abschließend noch einmal zurück zu obigem Beispiel:

Ein dritter Fan desselben Vereins sammelt ebenfalls Daten im Topic Maps Format. Er kann nun auf einfache Weise bestimmen, welche Subjects (Spieler, Positionen) in der ersten und in seiner *Topic Map* korrespondieren (struktureller Topic- und Association-Schnitt) und welche er alleine, bzw. welche er noch nicht aufgenommen hat (struktureller Topic- und Association-Differenz). Es ist nun auch einfach für ihn möglich nachzuprüfen, welche Daten zu den korrespondierenden Subjects in welchen *Topic Maps* vorhanden sind (*Topic*-Schnitt und -Differenz). Auch das Zusammenführen seiner *Topic Map* mit den anderen ist möglich (Vereinigung).

Die vorgestellten Operatoren sind ein sehr mächtiges Werkzeug zum Vergleich und Abgleich von *Topic Maps*. Es können einfach und intuitiv Unterschiede und Gemeinsamkeiten herausgestellt und verarbeitet werden. Im folgenden Kapitel wird ein prototypisches Framework vorgestellt, welches die oben definierten Operatoren umsetzt.

4. TMDiff Framework - Entwurf und Implementierung der Operatoren

Die im vorigen Kapitel vorgestellten Operatoren für *Topic Maps* sind im Framework `TMDiff` umgesetzt. Es soll hier gezeigt werden, dass eine praktische Umsetzung der Operatoren möglich ist und diese effizient und korrekt funktioniert.
Es werden die entwickelten Algorithmen für die einzelnen Operationen und deren Vorteile gegenüber einer naiven Implementierung erläutert. Im Anschluss werden das Framework an sich und dessen Testkonzept vorgestellt. Systematische Tests stellen korrektes Verhalten der Implementierung gegenüber der durch die Tests gegebenen Spezifikation sicher. Es ist Korrektheit für eine gewisse Bandbreite an typischen Anwendungsfällen gesichert.
`TMDiff` ist ein Framework, welches als solches einfach eingebunden werden kann. Außerdem kann es als eigenständiges Programm verwendet oder als Modul in jede beliebige Infrastruktur integriert werden. Wie `TMDiff` zu verwenden ist, wird in diesem Kapitel erläutert. Abschließend wird eine Analyse der Performance anhand ausgewählter *Topic Maps* durchgeführt, was die Effizienz der Implementierung experimentell bestätigen soll.

4.1. Algorithmus

4.1.1. Laufzeitabschätzungen

Lang definiert Zeitkomplexität folgendermaßen:

> Um den Algorithmus unabhängig von der konkreten Eingabe bewerten zu können, betrachtet man die Zeitkomplexität. Die Zeitkomplexität ist eine Funktion $T(n)$ in Abhängigkeit von der Problemgröße n. Der Wert von $T(n)$ ist die Laufzeit des Algorithmus im schlechtesten Fall (worst case), d.h. die Anzahl der Schritte, die bei einer beliebigen Eingabe höchstens ausgeführt werden. [Lan11]

Für eine sehr große Problemgröße n, lässt sich nun eine Schrankenfunktion $S(n)$ finden, für die gilt:
$T(n)$ wächst nicht wesentlich schneller als $S(n)$. Mit Hilfe der Landau-Symbole [Wik12b], insbesondere der Groß-O-Notation [Wik12a], lassen sich Algorithmen anhand ihrer Schrankenfunktion hierarchisch klassifizieren. Die durch die Landau-Symbole gegebenen charakteristischen Schrankenfunktionen definieren obere Schranken für die Laufzeiten der betrachteten Algorithmen. Dass heißt, je geringer die kleinste obere Schranke der Laufzeit eines Algorithmus ist, desto schneller bearbeitet der gegebene Algorithmus das vorliegende Problem (für große Problemgrößen n). Diese Hierarchie der oberen Schranken ist in Abbildung 4.1 dargestellt.

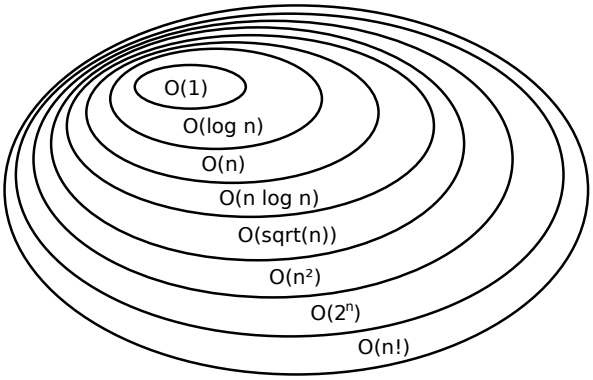

Abbildung 4.1.: Laufzeitklassen von Algorithmen mit Eingabegröße n

Das Ziel des Algorithmenentwurfes soll es sein, eine möglichst niedrige Laufzeit für große Eingaben zu erzielen, also in eine möglichst kleine?? Klasse aus Abb. 4.1 zu fallen. Es wird gezeigt (Abschn. 4.1.3), dass die obere Schranke des entworfenen Algorithmus durch eine Sortieroperation bestimmt ist, welche in der Klasse $O(n\ log\ n)$ liegt [JAV]. In den folgenden Abschnitten werden die Algorithmen zur Berechnung der Operationen und deren Laufzeitklassifizierung beschrieben.

4.1.2. Vorbetrachtung

Topic Maps finden vielseitige Anwendung und können nicht nur zur Modellierung von Ontologien als Metadaten verwendet werden, sondern auch zum Speichern von Instanzdaten. Hierbei treten häufig Performanceprobleme auf. Vergleichbar ist die Situation mit der relationalen Datenhaltung. Einem relativ kleinen Schema, welches zur Strukturbeschreibung erstellt wird, stehen viele Instanzdaten gegenüber, auf die über die gegebene Struktur zugegriffen werden soll. Auch hier sind effiziente Zugriffstechniken,

wie Indexstrukturen, Caching, optimierte Join-Bearbeitung, usw. (vgl. [HR01]) vonnöten, um akzeptable Antwortzeiten zu gewährleisten. Es sollen mit den definierten Operatoren solche, unter Umständen große (viele *Topics* und *Associations*), *Topic Maps* verglichen werden können. Deshalb spielen bei der Implementierung Performanceaspekte eine Rolle. Bei der Bestimmung der Performance eines Algorithmus ist dessen Laufzeit (s. Abschn. 4.1.1) ein kritisches Bewertungsmaß. Hat ein Algorithmus beispielsweise lineare oder logarithmisch-lineare Laufzeit, so wächst diese in einem akzeptablen Maße zur Größe der Eingabe (im konkreten Fall die Anzahl der *Topics* und *Associations*). Die folgend vorgestellten Vorgehensweisen sind stark auf das effiziente Berechnen der strukturellen Operatoren ausgerichtet. Es wird davon ausgegangen, dass die Anzahl der *Occurrences*, *Names* und *Variants* im Verhältnis zu den *Topics* eine Konstante ist. Es wird also davon ausgegangen, dass *Topics* relativ wenige Instanzen der genannten Konstrukte als Attribute besitzen. Bis auf Spezialfälle ist dies keine beschränkende Annahme und somit ist der kritische Performancefaktor nicht das Berechnen der atomaren Topic-Differenzen und -Schnitte, sondern die Berechnung der strukturellen Operationen. Es werden die einzelnen Schritte der Algorithmen und deren Laufzeitabschätzung aufgezeigt und erläutert.

4.1.3. Algorithmus

Zunächst wird ein einfacher Algorithmus beschrieben, welcher korrekt funktioniert, jedoch aufgrund seiner Laufzeit nicht das Kriterium der Effizienz erfüllt. Dies wird in den folgenden Abschnitten noch genauer erläutert werden. Für den Algorithmus werden die zwei Operanden bzw. Eingabe-*Topic Maps* A und B vorausgesetzt. Anhand derer wird das Vorgehen erläutert und der Algorithmus beschrieben. Im Anschluss wird der im Framework implementierte Algorithmus vorgestellt. Im nächsten Kapitel erfolgt dann ein Vergleich der Performance beider Vorgehensweisen, um nachzuweisen, wie effizient die vorgestellte Lösung ist.

Naiver Algorithmus

Das vorliegende Problem lässt sich prinzipiell einfach lösen. Für die strukturellen Topic-Operationen muss jedes *Topic* aus A mit jedem *Topic* aus B verglichen werden. Analog kann auch die Umsetzung der Association-Operationen erfolgen.

Naiver Algorithmus für das Berechnen des strukturellen Topic-Schnittes:

```
1: for jedes Topic t_a in A do
2:   for jedes Topic t_b in B do
3:     if t_a = t_b then
4:       result.add(t_a)
5:     end if
6:   end for
7: end for
8: return result
```

Der strukturelle Association-Schnitt wird analog zu diesem Verfahren berechnet und für die Differenzen muss jedes *Topic* bzw. jede *Association* hinzugefügt werden, für die kein gleiches Konstrukt in B gefunden wurde. Die Eingabegröße ist hier jeweils die Anzahl der *Topics* bzw. *Associations*. Somit hat diese Vorgehensweise eine quadratische Laufzeit, liegt also in $O(n^2)$. Um Skalierbarkeit des Verfahrens sicherzustellen ist quadratische Laufzeit unbedingt zu vermeiden, da nun die Laufzeit des Algorithmus in einem großen Verhältnis zur Eingabegröße steigt und sich daraus für große *Topic Maps* inakzeptable Berechnungszeiten der einzelnen Operationen ergeben. Deshalb werden in den nächsten Abschnitten optimierte Verfahren vorgestellt und deren Effizienz erläutert.

TMDiff-Algorithmen

Strukturelle Topic-Differenz

Zunächst werden alle *Topics* der jeweiligen *Topic Maps* A und B mit ihren Identifiern betrachtet. Die Identifier werden für A und B einzeln aufgelistet und lexikographisch aufsteigend sortiert. *Topics* können mehr als einen Identifier haben. Somit ist es möglich, dass in der sortierten Liste mehr Identifier existieren als *Topics*. Zu jedem Identifier wird die Zuordnung zum entsprechenden *Topic* erhalten. Es entstehen zwei Listen, eine für *Topic Map* A und eine andere für *Topic Map* B. Nun wird parallel durch beide Listen iteriert, indem immer ein Vergleich des aktuellen Identifiers aus Liste A mit dem Identifier aus Liste B erfolgt. Es wird in der Liste mit dem lexikographisch vorangehenden (in lexikographischer Sortierung kleinerem) Identifier fortgeschritten. Sollten beide Identifier identisch sein, so wird ein Mapping zwischen den jeweiligen *Topics* erstellt und gespeichert. Am Ende der Iteration existieren Mappings von *Topics* aus A zu *Topics* aus B. *Topics*, welche in beiden *Topic Maps* vorkommen (für die

ein Mapping existiert), sind nun nicht Teil des Ergebnisses und werden nicht weiter beachtet. Es werden also diejenigen *Topics* aus A zur Ergebnismenge hinzugefügt, für die kein Mapping existiert, welche also in B kein Äquivalent haben. Dies entspricht der Spezifikation der Operation.

Der Algorithmus in Pseudocode:

1: generiere eine lexikographisch sortierte Liste ids_A (bzw. ids_B), die alle Identifier der *Topics* aus *Topic Map* A (bzw. B) enthält
2: **while** ids_A nicht leer && ids_B nicht leer **do**
3: **if** ids_A.first() $<$ ids_B.first() **then**
4: entferne ersten Identifier aus ids_A
5: **else if** ids_A.first() == ids_B.first() **then**
6: generiere Mapping zwischen den *Topics* mit den Identifiern ids_A.first() und ids_B.first
7: entferne ids_A.first() und ids_B.first
8: **else if** ids_A.first() $>$ ids_B.first() **then**
9: entferne ersten Identifier aus ids_B
10: **end if**
11: **end while**
12: **return** alle *Topics* aus A, für die kein Mapping existiert

Die Laufzeitabschätzung für den vorgestellten Algorithmus:

#	**Beschreibung**	**Laufzeitklasse**
1	jeweils lexikographisches Sortieren aller Identifier der Topics beider Maps	$O(n\ log\ n)$
2	paralleles Iterieren der sortierten Listen und Generieren der Mappings	$O(n)$
3	Generieren der Ergebnismenge anhand der Mappings	$O(n)$
	Laufzeitklasse gesamt	**O(n log n)**

Tabelle 4.1.: Laufzeitabschätzung der Topic Differenz/Schnitt-Berechnung

Die eigentliche obere Schranke, welche die Laufzeit des Algorithmus begrenzt, ist der Sortiervorgang beim Generieren der Listen. Der Teil des Algorithmus, welcher Listen sortiert, liegt in $O(m\ log\ m)$ ([JAV]), wobei m die Anzahl der zu sortierenden Elemente, also der Identifier, ist. Es kann davon ausgegangen werden, dass sich die Anzahl der

Identifier als Konstante verhält. *Topics* haben gewöhnlicherweise wenig Identifier. Somit liegt die Laufzeit für das Sortieren weiterhin in $O(n\ log\ n)$, wobei n die Anzahl der *Topics* ist. Die anderen Schritte durchlaufen die vorher sortierten Listen einmalig und es müssen höchstens $|ids_A| + |ids_B|$ (jeweils Anzahl der Identifier aller *Topics* in den Eingabe-*Topic Maps*) Vergleiche durchgeführt werden. Im Vergleich zum naiven Algorithmus zu Beginn des Abschnittes, hat sich beim TMDiff-Algorithmus die Laufzeit auf die Klasse $O(n\ log\ n)$ verringert.

Struktureller Topic-Schnitt

Der Algorithmus für die Berechnung des Topic-Schnittes ähnelt dem für die Berechnung der Differenz. Der Unterschied in den Operationen (vgl. Kap. 3.2.2) besteht darin, dass nicht die für *Topic Map A* exklusiven, sondern die in A und B vorhandenen *Topics* im Ergebnis enthalten sind. Der Algorithmus für die strukturelle Topic Differenz muss nur geringfügig abgeändert werden, um den Schnitt an Stelle der Differenz zu berechnen. Beim Generieren der Ergebnisliste werden am Ende jene *Topics* ins Ergebnis übernommen, für die ein Mapping existiert. Die Zusammenfassung und Laufzeitabschätzungen sind ebenfalls Tabelle 4.1 zu entnehmen.

Strukturelle Association-Differenz

Das vorangehend vorgestellte Prinzip des Ordnens der Konstrukte ist für *Associations* nicht adaptierbar, da für sie keine Identität definiert ist, welche in irgendeiner Form einer Ordnung unterliegt. Ein Sortieren der *Associations* ist also nicht möglich und somit ein anderes Verfahren für die Berechnung nötig. Wie in Def. 3.5 festgelegt, hängt die Identität einer *Association* von deren *type*, *scope* und *roles* ab. Die Identität einer *role* hängt wiederum von deren *player*, *type* und *scope* ab. Ist eines dieser *Topics* einer *Association* aus *Topic Map A* nicht in B vorhanden, so ist dieses eine hinreichende Bedingung für das Nicht-Vorhandensein der *Association* in B.
Dieses Prinzip wird nun ausgenutzt um die Anzahl der möglichen, gleichen *Associations* auf ein Minimum zu reduzieren bzw. das Vorhandensein direkt auszuschließen. Es wird, wie bei den Topic-Operatoren, ein Mapping der *Topics* aus A zu ihren äquivalenten *Topics* aus B erstellt. Nun kann dieses Mapping genutzt werden, um für jede *Association* aus A eine Menge von möglichen *Associations* zu erstellen, welche identisch zur jeweiligen, gegebenen *Association* sind. Diese Menge von *Associations* bestehen im Normalfall aus keiner oder einer (der tatsächlich gleichen) *Association*. So wird auch für die Associations-Operatoren Skalierbarkeit gewährleistet.

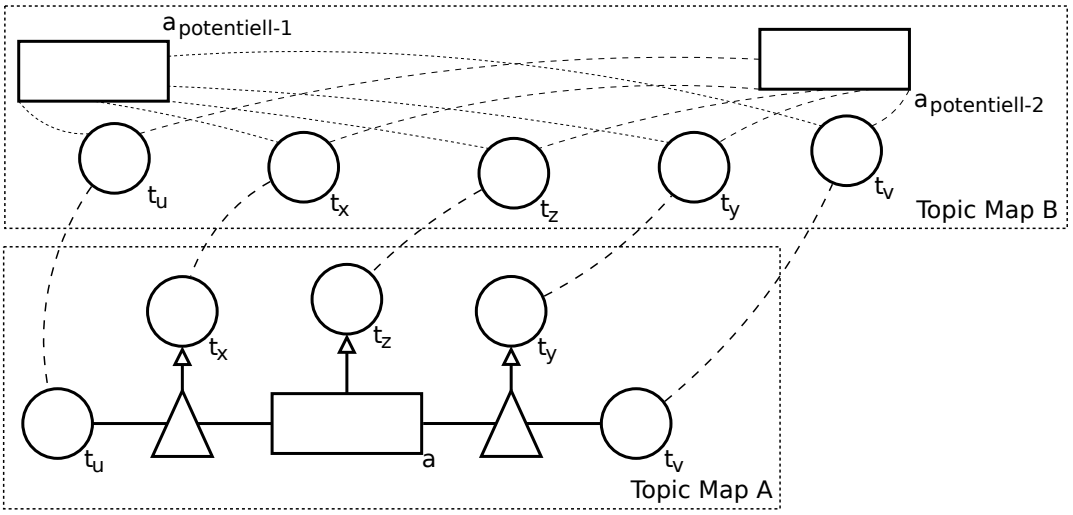

Abbildung 4.2.: Mappings der *Topics* einer *Association* zu den entsprechenden *Topics* in einer zweiten Map

Wie in Abbildung 4.2 dargestellt, werden zu den *Topics*, welche zu a gehören, über das erstellte Mapping alle gleichen *Topics* in der anderen *Topic Map* herausgesucht. Nun werden für den *Association* Typ t_z alle *Associations* in B herausgesucht, die t_z als Typ haben. Gleiches wird für die *role*-Typen und *player* durchgeführt. Sollte es für ein *Topic* aus A mehrere gleiche *Topics* in B geben (in der Abbildung nicht dargestellt), so entstehen Mengen von *Associations*. Diese Mengen werden nun vereinigt und es bleibt eine Menge möglicherweise gleicher *Associations* zurück (in Abb. 4.2: $\{a_{potentiell-1}$, $a_{potentiell-2}\}$). Der letzte Schritt ist dann die konkrete Prüfung auf Gleichheit der *Association* aus A mit den *Associations* in der Menge der möglichen *Associations* aus B. Anzumerken ist hier, dass der Einfachheit halber der *scope* der *Associations* und *roles* in Abb. 4.2 jeweils nicht dargestellt wurden. Ist also festgestellt worden, dass für a eine gleiche *Association* in B existiert, so ist a nicht Teil des Ergebnisses der strukturellen Association-Differenz. Wird kein gleiches Konstrukt gefunden, ist a der Ergebnismenge hinzuzufügen.

Der Algorithmus im Pseudocode:

1: generiere Mappings aller *Topics* aus Topic Map A zu semantisch äquivalenten *Topics* aus *Topic Map B*
2: **for** jede *Association* a in *Topic Map A* **do**
3: erstelle Menge $\{a_1^*, a_2^*, a_3^*, ...\}$ von *Associations*, welche aufgrund der *Topics* potentiell äquivalent zu a sind
4: **for** jede *Association* a^* in $\{a_1^*, a_2^*, a_3^*, ...\}$ **do**
5: **if** $a = a^*$ **then**
6: $assocs.add(a)$
7: **end if**
8: **end for**
9: **end for**
10: **return** $\{a \in A | a \notin assoc\}$

Eine kurze Zusammenfassung und die Laufzeitabschätzung als Übersicht findet sich in Tabelle 4.2, wobei n die Anzahl der *Topics* und m die Anzahl der *Associations* ist. Für Schritt 3 ist die Laufzeit im average case ([ac]) angegeben, also die Laufzeit, welche im „Normalfall“ zu erwarten ist.

#	**Beschreibung**	**Laufzeitklasse**
1	Generieren der Mappings (Sortierung)	$O(n\ log\ n)$
2	Heraussuchen der möglichen *Associations* $\{a_1^*, a_2^*, a_3^*, ...\}$ für jede *Association* a in *Topic Map A*	$O(m)$
3	Für alle *Associations* a: Prüfe ob $a \in \{a_1^*, a_2^*, a_3^*, ...\}$	$O(m)$ [ac]
	Laufzeitklasse gesamt	**O(m+n log n)**

Tabelle 4.2.: Laufzeitabschätzung der Association Differenz/Schnitt-Berechnung

Zu beachten ist an dieser Stelle, dass sich der Algorithmus im schlechtesten Fall quadratisch verhält. Dieser Fall tritt ein, wenn für jede *Association* in *Topic Map A* jeweils die gesamte Menge von *Associations* als Menge potentiell gleicher *Associations* zurückgegeben werden würde. Dieser Fall wäre jedoch eine sehr ungewöhnliche Konstruktion und spielt für keine anwendungsorientierte *Topic Map* eine Rolle.

Struktureller Association-Schnitt

Die Berechnung des strukturellen Association-Schnittes ist ähnlich zur Berechnung der Association-Differenz. Der Unterschied in den Operationen (vgl. Kap. 3.2.2) besteht darin, dass nicht die in *Topic Map A* exklusiv, sondern die in A und B enthaltenen *Associations* im Ergebnis vorhanden sein sollen. Also entspricht das Vorgehen im Prinzip dem o.g. Vorgehen mit dem Unterschied, dass jetzt jene *Associations* zurückgegeben werden, für die semantische Äquivalenz festgestellt wurde. Zusammenfassung und die Laufzeitabschätzung sind ebenfalls Tabelle 4.2 zu entnehmen.

4.2. Design des Frameworks

In diesem Abschnitt wird der Aufbau des Programmes `TMDiff`, welches die im vorigen Kapitel definierten Operatoren für Topic Maps umsetzt, beschrieben. Es wird der Aufbau des Frameworks erklärt, wie Korrektheit der Implementierung sichergestellt wird und wie das Framework eingesetzt bzw. benutzt werden kann. Zunächst werden die Anforderungen dargestellt und danach die Umsetzung erläutert.

4.2.1. Anforderungen

Wie in den vorigen Abschnitten schon genannt, liegt der Fokus der Implementierung auf Korrektheit und Effizienz der Umsetzung. Korrektheit kann dabei bis zu einem gewissen Grad durch ein strukturiertes Testkonzept gewährleistet werden. Im Abschnitt 4.3 wird das Testkonzept im Einzelnen vorgestellt. Softwaretests können zwar grundsätzlich nicht jeden Fehler verhindern, garantieren jedoch korrektes Verhalten gegenüber den durchgeführten Tests und damit für ein gewisses Spektrum an Anwendungsfällen. Außerdem unterstützen Tests das Erstellen modularer Software, da einzelne Funktionen und Komponenten getestet werden und diese Teile dann austausch- und leichter wartbar sind.
Effizienz im Sinne der Laufzeit wurde bereits im vorigen Abschnitt diskutiert. Es wurde gezeigt, dass die Algorithmen, welche jeweils zur Umsetzung der Operatoren erklärt wurden, in der logarithmisch-linearen Laufzeitklasse liegen und somit besser sind, als naives Vergleichen jedes *Topics* aus *Topic Map A*, mit jedem *Topic* aus *Topic Map B*.
Software sollte unabhängig von Programmiersprachen und Frameworks verwendbar sein. Das zwingende Festlegen der Anwender auf bestimmte Techniken reduziert die Wiederverwendbarkeit und Handhabbarkeit der Software. Für `TMDiff` wird ein Java-Interface beschrieben. Außerdem ist die Nutzung als eigenständiges Programm möglich. Es werden Austauschformate definiert und erläutert. Hiermit können Anwender unabhängig von deren System-Architektur und Umgebung auf die Software zugreifen und sie nutzen.

4.2.2. Programmaufbau

In diesem Abschnitt soll grob der Aufbau des Prototypen für die Berechnung der Operatoren erläutert werden. Dabei wird zunächst das zentrale Interface beschrieben und dann auf die Implementierung eingegangen.

Voraussetzungen

Die Implementierung basiert auf der am Topic Maps Lab [TML11] entwickelten Topic Maps Engine „MaJorToM“. „MaJorToM“ steht für „Merging Java Topic Maps Engine“. Das Nutzen von „MaJorToM“ hat einige Vorteile gegenüber anderen Engines. Sie ist quelloffen und implementiert strikt das Topic Maps Application Programming Interface (TMAPI) [TMA12b]. Das TMAPI ist ein in der Topic Maps Welt allgemein anerkannter Standard und sichert somit auch die Austauschbarkeit der Backend-Implementierung und damit die Wieder- und Weiterverwendbarkeit des gesamten Frameworks. „MaJorToM“ ist softwaretechnisch und im tatsächlichen Einsatz (Maiana [MAI11]) getestet und frei verfügbar. Außerdem hat „MaJorToM“ noch einen weiteren Vorteil gegenüber anderen Engines. Das im TMDM spezifizierte Mergen ist implementiert und selbst innerhalb einer *Topic Map* werden Äquivalenzen automatisch erkannt und aufgelöst. Dadurch, dass die Engine und somit das Backend in Java umgesetzt sind, bietet sich Java als Programmiersprache für die Umsetzung des Frameworks an. Außerdem existieren zahlreiche Bibliotheken für Tests, komplexe Datenstrukturen und Kommunikation, welche das Entwickeln erleichtern.

Interface TMDiff

Im Zentrum des Frameworks steht das Interface `TMDiff`. Das Interface enthält sieben Methoden, welche die hier vorgestellten Operatoren struktureller Assoziationen- und Topic-Schnitt, sowie Differenz und atomarer Topic-Schnitt und -Differenz und zusätzlich noch die im TMDM spezifizierte Merge-Operation abbildet.

```
1  public interface TMDiff {
2
3    public Set<Topic> semanticTopicDiff(TopicMap tm1,
4        TopicMap tm2);
5    public Set<Topic> semanticTopicIntersect(TopicMap tm1,
6        TopicMap tm2);
7    public Set<Association> semanticAssociationDiff(
8        TopicMap tm1, TopicMap tm2);
9    public Set<Association> semanticAssociationIntersect(
10       TopicMap tm1, TopicMap tm2);
11   public void merge(TopicMap tm1, TopicMap tm2);
12   public Fragment topicDiff(Topic t1, Topic t2);
13   public Fragment topicIntersect(Topic t1, Topic t2);
14
15 }
```

Die ersten vier Methoden bilden die strukturellen Operationen ab und erwarten als Übergabeparameter zwei TMAPI-*Topic Maps*, welche jeweils eine TMDM *Topic Map* repräsentieren. Das Ergebnis sind Mengen von *Topics* bzw. *Associations*, wie in Abschnitt 3.2.2 spezifiziert.
Die Zusammenführung der beiden *Topic Maps* kann mit Hilfe der Merge-Operation vollzogen werden. Dies entspricht der Vereinigung der beiden *Topic Maps*, was auch schon vorher erläutert wurde.
Die letzten beiden Methoden bilden die Operationen auf Eigenschaftsebene ab. Auch hier wurde die Funktionalität ausführlich erläutert. Der Rückgabewert ist ein Fragment-Objekt, welches alle Elemente des *Topic*-Tupels und die entsprechenden Zugriffsmethoden bereithält.

4.3. Tests

Mit Hilfe des Testkonzeptes soll das korrekte Arbeiten der Methoden sichergestellt werden. Das gewünschte Verhalten der Operatoren ist im vorigen Kapitel beschrieben worden. Es werden nun für jede Methode eine Reihe von Tests implementiert, die das Verhalten der Methode gegenüber der Spezifikation des entsprechenden Operators testet und so Korrektheit gegenüber dieser Spezifikation gewährleistet.
Es wird hierfür das Testframework *JUnit* [JUN12] verwendet. *JUnit* ermöglicht es auf einfache Art und Weise einzelne Softwaremodule, deren Klassen und Methoden zu testen. Dabei fördert es gleichzeitig die Software selbst, modularer zu entwickeln. Fehlerisolation, Austauschbarkeit und Verständlichkeit sind Vorteile, die sich unmittelbar daraus ergeben.

4.3.1. Unit Tests

Im Framework `TMDiff` existiert ein eigener Ordner `src/test` für die Testdateien. Neben Tests von einzelnen, kritischen Methoden (im `utils` Paket) befinden sich hier die sechs Testklassen für die einzelnen Operatoren (die Vereinigung wird nicht getestet, da diese Funktionalität von der Engine bereitgestellt wird und dort getestet ist). Beim Unit-Testing wird pro Testfall ein Modul oder Teil der Software getestet.
Beispielsweise wird die Methode zum Feststellen der Gleichheit von *Topics* an mehreren Stellen benötigt. Nun wird hierfür ein Test geschrieben, welcher das Verhalten der Methode spezifiziert. Daraufhin wird eine Methode implementiert, welche ein definiertes Ein- und Ausgabeverhalten aufweist. Die Methode ist für sich gekapselt. Dass heißt, Fehler können leicht isoliert werden. Außerdem spezifizieren die Tests das Verhalten der Methode. Sie ist somit leicht verständlich und austauschbar, sollte sich die Spezifikation des Verhaltens ändern. So wurde beispielsweise in Abschnitt 3.1 diskutiert, wie die Identität von *Topics* festzulegen ist. Sollte nun eine andere Form der Identität festgelegt werden, muss nur eine Methode ausgetauscht und deren Tests angepasst werden. Der Rest der Software bleibt unberührt.

Beispiel

Das Testkonzept soll am Beispiel des strukturellen Topic-Schnittes erläutert werden. Der strukturelle Topic-Schnitt Operator $\cap_t$ ist in `TMDiff` durch die Funktion:
`public Set<Topic> semanticTopicIntersect(TopicMap tm1, TopicMap tm2);`
abgebildet. Es werden zwei *Topic Maps* an die Methode übergeben und es soll eine Menge von *Topics* zurückgeliefert werden, welche die *Topics* enthält, die Teil beider *Topic Maps* sind. Genau dieses Verhalten wird nun getestet. Abbildung 4.3 verdeutlicht, welche Fälle dabei zu beachten sind.

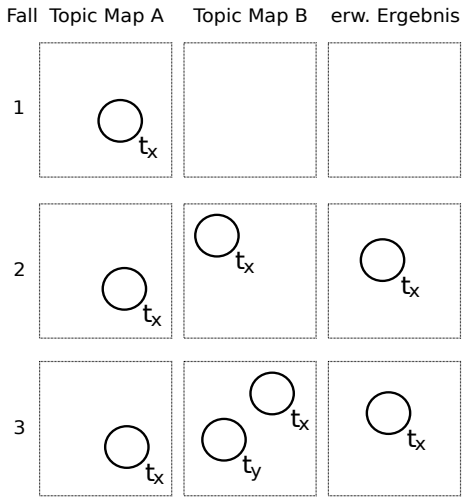

Abbildung 4.3.: Testsuite für den strukturellen Topic-Schnitt mit Ausgangsmaps und Ergebnis

Es müssen Tests für folgende Fälle entwickelt werden:
Es gibt in B keine gleichen *Topics* zu dem *Topic* in A, es gibt ein *Topic*, welches gleich ist und es gibt zusätzlich *Topics* in B, welche nicht gleich zu einem *Topic* in A sind, welche also nicht im Ergebnis auftauchen dürfen. Außerdem sollte zusätzlich die Symmetrie der Operation getestet werden, indem man die Operanden vertauscht. Für die anderen strukturellen Operatoren wird analog vorgegangen. Welches Ergebnis jeweils zu erwarten ist, wurde bereits ausführlich diskutiert.
Die atomaren Topic-Operatoren müssen ebenfalls getestet werden. Hier werden jeweils die Identifier (*Subject Identifier*, *Item Identifier*, *Subject Locator*) und die Konstrukte *Names*, *Variants* und *Occurrences* miteinander verglichen. Diese Vergleiche sind Mengenoperationen, da immer eine Menge des jeweiligen Konstrukts mit dem *Topic* assoziiert

ist. Außerdem wird auch das korrekte Behandeln der Reifier getestet. Auch hier müssen wieder die oben genannten Fälle für die entsprechenden Mengen unterschieden werden.

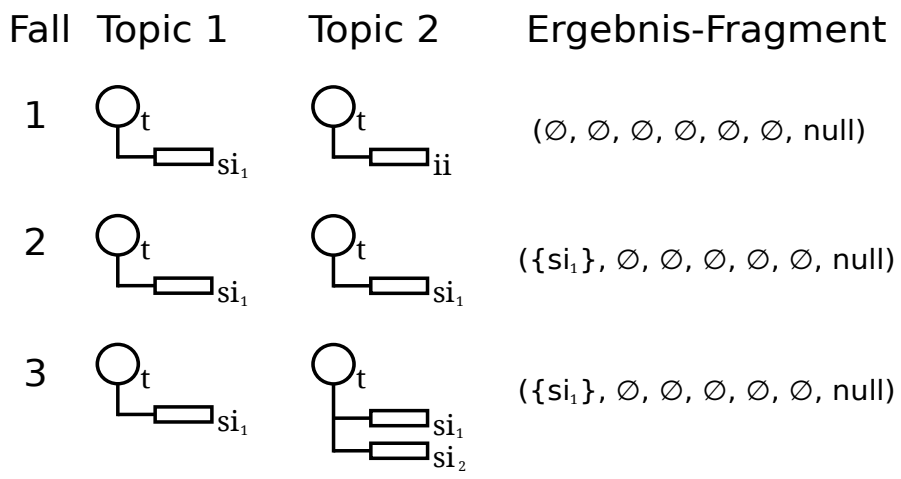

Abbildung 4.4.: Beispiel für das Testen des atomaren Topic-Schnittes anhand der *Subject Identifier* (si - *Subject Identifier*, ii - *Item Identifier*)

Das Verfahren, welches für die *Subject Identifier* dargestellt ist, wird dann auch für die anderen Attribute des *Topics* angewandt und die Tests dementsprechend gestaltet. Beim Vertauschen der Operanden ist ebenfalls das gleiche Ergebnis-Fragment zu erwarten, da die atomare Topic-Schnitt Operation symmetrisch ist.
Analog hierzu werden die Tests für die atomare Topic-Differenz gestaltet. Was die jeweiligen Ergebnisse der Testfälle sind, ist schon bei der Beschreibung der Operationen (Abschn. 3.2) erläutert worden und wird daher hier nicht wiederholt.

4.3.2. Integrationstest

Das Ziel des Integrationstests ist es, die Operationen für komplexe *Topic Maps* zu testen. Komplex bedeutet, dass sie viele *Topics* beinhalten, die wiederum über viele *Associations* in Verbindung stehen. Es wird z.B. getestet, ob keine Speicherüberläufe für typische Anwendungsszenarien entstehen oder die Software aufgrund komplexer Zusammenhänge von *Topics* und *Associations* unter Umständen in Endlosschleifen gerät oder ähnliches. Außerdem soll gewährleistet werden, dass auch bei hochgradig miteinander vernetzten Datensätzen kein unerwartetes Verhalten entsteht und die Implementierung noch der gewünschten Spezifikation entspricht.

4.3.3. Performancetest

Der Performancetest soll die absolute und relative Laufzeit der in `TMDiff` implementierten Algorithmen testen. Dazu wird zunächst die absolute Laufzeit des Algorithmus anhand einer Referenz-*Topic Map* bestimmt und dann mit der Laufzeit der naiven Implementierung verglichen. Leider gibt es zur Zeit keine so ähnlichen Arbeiten, dass ein aussagekräftiger Vergleich durchgeführt werden kann, also bleibt nur der Vergleich mit einer Implementierung, welche die selbe Spezifikation erfüllt, jedoch „offensichtlich“ umgesetzt ist. Eine offensichtliche oder naive Umsetzung des Algorithmus spiegelt sich in der Art und Weise der Beschreibung des Algorithmus und der Anzahl der benötigten Codezeilen (LOC) wieder. Sie ist häufig die erste Herangehensweise für die Problemlösung und lässt sich einfach und kurz beschreiben. Häufig ist der so entstandene naive Algorithmus jedoch nicht die performanteste Lösung für das gegebene Problem.
Der Algorithmus für die naive Implementierung wurde bereits in Abschnitt 4.1.3 erläutert. Die Ergebnisse der Performancetests sind in Abschnitt 4.5 dargestellt.

4.4. Benutzung

In diesem Abschnitt soll beschrieben werden, wie `TMDiff` in eine bestehende Softwareinfrastruktur eingebettet und genutzt werden kann. Zunächst wird erklärt, wie sich das Framework an sich direkt als Java-Modul einbinden lässt. Danach wird erläutert, wie `TMDiff` mit Hilfe von spezifizierten Austauschformaten als Komponente in andere Infrastrukturen eingebunden werden kann.

4.4.1. Direkte Einbindung des Java-Frameworks

Um das Framework direkt in ein eigenes Java Projekt einzubinden, muss zunächst das fertig gebaute JAR unter [Fel12b] heruntergeladen werden. Danach muss das geladene Paket in den Build-Path des Projektes übernommen werden. Nun lässt sich direkt das in Abschn. 4.2.2 definierte Interface nutzen. Die Methoden des Interfaces erwarten TMAPI konforme *Topic Maps* bzw. *Topics*.
Topic Maps, die mit Hilfe einer TMAPI konformen Engine erstellt worden sind, können einfach an die Interface Methoden übergeben werden. Sollte die *Topic Map* jedoch serialisiert (z.B. als XTM Datei [GM06]) vorliegen, kann diese in eine TMAPI konforme *Topic Map* geparsed werden. Hierfür bietet sich beispielsweise TMAPIX [TMA12a] an. Dazu ist ein Beispiel im Anhang (A.1) angegeben.

4.4.2. Als Programm oder Modul

Um `TMDiff` direkt über eine Konsole nutzen zu können, muss das fertig gebaute JAR mit allen Abhängigkeiten heruntergeladen werden (unter [Fel12a]). Außerdem muss Java (Version 6) auf dem ausführenden Rechner installiert sein (siehe [JAV12]). Sind diese Voraussetzungen erfüllt, wechselt man in der Konsole zu dem Verzeichnis, in dem sich das heruntergeladene Archiv befindet und startet das Programm folgendermaßen:
`java -jar TMDiffComplete <option> <file1> <file2>`
Die beiden (`<file1>`, `<file2>`) File-Parameter müssen als Pfade zu den entsprechenden XTM-Topic-Map-Dateien angegeben werden. Dabei ist `<file1>` der erste Operand und `<file2>` der zweite. Der `<option>`-Parameter kann einer oder mehrere der folgenden sein:

- `-topicDiff` (strukturelle Topic-Differenz)
- `-topicIntersect` (struktureller Topic-Schnitt)
- `-associationDiff` (strukturelle Association-Differenz)
- `-associationIntersect` (struktureller Association-Schnitt)
- `-atomicDiff` (atomare Topic-Differenz)
- `-atomicIntersect` (atomarer Topic-Schnitt)
- `-d <path>` (Pfad zu den Ergebnisdateien)

Bei Angabe einer der ersten vier Parameter wird die jeweils aufgeführte strukturelle Operation durchgeführt und ein Ergebnis in Form einer XTM-Topic-Map-Datei zurückgeliefert. Wie das Ergebnis zu interpretieren ist, wird im nächsten Abschnitt erklärt.
Werden der fünfte bzw. sechste Parameter gesetzt, so wird für alle Topics, welche sich im strukturellen Topic-Schnitt befinden, jeweils die atomare Topic-Differenz bzw. der atomare Topic-Schnitt berechnet und das Ergebnis in Form einer XML-Datei zurückgegeben. Die Spezifikation des Rückgabeformates ist ebenfalls im nächsten Abschnitt erläutert. Der letzte Parameter erfordert die Angabe eines Wertes (`<path>`). Hiermit kann ein Ordner angegeben werden, in welchem die Ergebnisdateien der durchgeführten Operationen abgelegt werden. Wird dieser Parameter nicht gesetzt, so werden die Dateien in das gleiche Verzeichnis geschrieben, in der sich auch das ausgeführte JAR befindet.

Spezifikation der Austauschformate

Strukturelle Operationen

Die Ergebnisse der strukturellen Operationen für das Konsolenprogramm *TMDiff*, sind jeweils der XTM2-Spezifikation entsprechende XML-Dateien. Dass heißt, sie können genauso deserialisiert werden, wie dies auch mit den ursprünglichen XTM-Topic-Map-Dateien der Fall ist. Die strukturellen Topic-Operatoren liefern XTM-Topic-Maps mit einer Menge von Topic-Stubs. Topic-Stubs sind hier *Topics*, welche mindestens einen Identifier haben, über den das *Topic* im ersten Operanden referenziert werden kann. Ist also ein *Topic* in der Ergebnismenge der jeweiligen Operation enthalten, so ist für dieses *Topic* ein Topic-Stub in der Ergebnis-*Topic Map* vorhanden. Über diesen Topic-Stub kann nun wieder das ursprüngliche *Topic* gefunden werden. Es werden nur Topic-Stubs und nicht komplette *Topics* hinzugefügt, da das Hinzufügen eines *Topics* zu einer *Topic Map* häufig das Hinzufügen mehrerer anderer *Topics* nach sich zieht. So müssten beispielsweise die Typen der *Occurrences* eines *Topics* ebenfalls zur Ergebnis-*Topic Map* hinzugefügt werden, was an dieser Stelle jedoch das Ergebnis verfälschen würde. Mit dem Hinzufügen von bloßen Topic-Stubs, die nur Identifier für die Referenz auf das Ursrpungs-*Topic* enthalten, wird dieses Problem umgangen.
Das Ergebnis der strukturellen Association-Operationen sind ebenfalls valide XTM2-Topic-Map-Dateien, welche jeweils eine Menge von *Associations* enthalten, die der Spezifikation der jeweiligen Operation entspricht.

Atomare Operationen

Die atomaren Topic-Operatoren liefern als Ergebnis jeweils XML-Dateien, welche dem in Anhang A.3 gezeigten Schema entsprechen. Jedes Konstrukt, welches in der Ergebnis-XML-Datei auftaucht, ist eine komplette Referenz auf das entsprechende Konstrukt im ersten Operanden der Operation. Das heißt, die Informationen zur Identität der Konstrukte sind vollständig vorhanden. Im Wurzelelement der XML-Datei befinden sich mehrere `<topic>`-Elemente, welche jeweils ein Topic-Fragment repräsentieren, also das Ergebnis einer atomaren Topic-Operation darstellen. (Man beachte, dass die ausgewählte Operation für alle *Topics* in der ersten *Topic Map* mit ihren jeweils äquivalenten *Topics* in der zweiten *Topic Map* durchgeführt wird.) Ein Ausschnitt einer solchen Ergebnis-XML-Datei ist im Anhang A.2 gegeben. Das `<topic>`-Element enthält genau ein `<reference>`-Element, welches ein Attribut (`si` für *Subject Identifier*, `sl` für *Subject Locator*, `ii` für *Item Identifier*) hat, dessen Wert auf ein *Topic* in der Ausgangs-*Topic Map* zeigt. Danach folgen jeweils Mengen von `<si>` (*Subject Identifier*), `<sl>` (*Subject Locator*), `<ii>` (*Item Identifier*), `<name>` (*Name*), `<variant>` (*Variant*) und `<occurrence>` (*Occurrence*) Ele-

menten. Jedes dieser Elemente ist über dessen Identität auffindbar. Die `<parent>`-, `<type>`- und `<scope>`-Elemente zeigen über den im Attribut definierten Identifier auf die entsprechenden *Topics* und die `<value>`- bzw. `<datatype>`-Elemente enthalten die Zeichenkette, welche auch im entsprechenden Attribut des *Topics* zu finden ist.

4.5. Bewertung

Dieser Abschnitt soll die theoretischen Überlegungen zur Leistungssteigerung des entworfenen Algorithmus anhand ausgewählter Anwendungsszenarien bestätigen. Wie schon in Abschnitt 2.7 aufgezeigt, existiert keine gleichwertige Umsetzung, mit Hilfe derer ein Vergleich der Performance des Frameworks möglich wäre. Daher wird der in Abschnitt 4.1.3 vorgestellte naive Algorithmus als Referenz implementiert und der relative Laufzeitvergleich mit der `TMDiff`-Implementierung als Referenz vollzogen.
Die genutzten Testdaten sind zum einen Steve Peppers Italien Opera *Topic Map* („Opera“ - zu finden unter [Pep11]) und zum anderen Uta Schulzes ToyTM *Topic Map* („ToyTM“ - zu finden unter [Sch11]). Opera wurde als Test für die Ontopia Topic Maps Engine [Ont12] erstellt und enthält eine Vielzahl verschiedener Konstruktionen. Sie enthält viele *Associations* und *Topics* und bietet sich somit für einen Test der Performance und Korrektheit von `TMDiff` an. ToyTM enthält im Gegensatz dazu weniger *Topics* und *Associations*, ist allerdings auch vielseitig und gut für Tests geeignet.
Bei den Performancetests sind drei Anwendungsszenarien von Bedeutung, die im tatsächlichen Einsatz typisch sind:

1. Vergleich komplett unterschiedlicher *Topic Maps*
2. Vergleich vollständig gleicher *Topic Maps*
3. Vergleich geringfügig unterschiedlicher *Topic Maps*

Für das erste Szenario werden die Operationen für Opera und ToyTM durchgeführt (es gibt keine gleichen Konstrukte in beiden *Topic Maps*). Beim zweiten Szenario werden die Operationen mit zwei vollständig gleichen *Topic Maps* durchgeführt. Dazu werden jeweils Opera und ToyTM mit sich selbst verglichen. Für das letzte Szenario wird Opera leicht abgeändert. Es werden zufällig einige wenige *Associations* und *Topics* gelöscht. Die so entstandene *Topic Map* Opera(mod) wird nun mit dem Original verglichen. Es werden also zwei ähnliche *Topic Maps* auf Gemeinsamkeiten und Unterschiede hin untersucht.
Für alle durchgeführten Operationen ist eine erhebliche Laufzeitverbesserung des TMDiff-Algorithmus gegenüber dem entsprechenden naiven Algorithmus zu erwarten.

Um welchen Faktor sich die Laufzeit verbessert, ist auch von der Eingabegröße abhängig. Da die vorgestellten Algorithmen (naiv und TMDiff) sich in ihrer Laufzeitklasse unterscheiden, ist zu erwarten, dass der Faktor der Verbesserung bei größerer Eingaben ebenfalls größer wird. So sollte der Vergleich von ToyTM mit sich selbst eine kleinere Laufzeitverbesserung erzielen, als der Vergleich von Opera mit sich selbst. Bei der strukturellen Topic Differenz (**TD**) sollte die Reihenfolge der Operanden keine signifikante Rolle spielen, da jeweils beide Identifier-Listen parallel iteriert werden. Für den strukturellen Topic-Schnitt (**TS**) sollte dies ebenso zutreffen. Für die strukturelle Association-Differenz (**AD**) sind die längsten Laufzeiten zu erwarten, da hier für jede potentiell gleiche *Association* ein Vergleich durchgeführt werden muss. Sollte beim strukturellen Association-Schnitt schon für die erste potentiell gleiche *Association* eine Übereinstimmung gefunden werden, so ist diese dann Teil des Ergebnisses. Bei der **AD** sind erwartungsgemäß mehr Vergleiche nötig. Außerdem hängen die Association-Operationen stark von der Eingabe ab. Sollten die *Topic Maps* sich wenig gleichen (Szenario 3), so tritt das Ausschließen der Möglichkeit einer Übereinstimmung sehr häufig auf und die Berechnung ist an dieser Stelle schnell beendet. Im Folgenden werden die gemittelten Laufzeiten der Operationen über jeweils zehn Versuche in Millisekunden und der Verbesserungsfaktor der Laufzeit von `TMDiff` im Vergleich zum naiven Algorithmus angegeben.

Szenario 1

Opera ∘ ToyTm

∘	**TD**	**TS**	**AD**	**AS**
naiver Alg.	1.198,1	1.207,4	6.969,2	1.222,4
TMDiff-Alg.	65,2	64,4	155,9	63,2
Faktor	18.38	18,75	44,70	19,34

Tabelle 4.3.: Szenario 1: Zeit in ms für die Operationen anhand von Opera und ToyTM

ToyTM ∘ Opera

∘	**TD**	**TS**	**AD**	**AS**
naiver Alg.	1.181,4	1.180,3	7.382,9	1.195,4
TMDiff-Alg.	58,5	57,1	64,2	59,6
Faktor	20,19	20,67	115,00	20,06

Tabelle 4.4.: Szenario 1: Zeit in ms für das Berechnen der Operationen für ToyTM und Opera

Die bestimmten Laufzeiten für das erste Szenario entsprechen dem Erwartungsbild. Die größte Zeiteinsparung ergibt sich für die **AD**. Hier ist allerdings entscheidend, in welcher Reihenfolge die Operanden eingesetzt werden. Diese Operation iteriert über alle *Associations* aus der ersten *Topic Map*. Da Opera mehr *Associations* als ToyTM enthält, dauert die Berechnung der Operation in der ersten Variante auch länger.
Erwartungsgemäß sollten sich die Laufzeiten für **TD** und **TS** beim TMDiff-Algorithmus nicht unterscheiden. Der hier jedoch vorhandene Unterschied basiert auf dem Vorgehen für jedes *Topic* in der ersten *Topic Map*, im Mapping nach einem gleichen *Topic* zu suchen. Somit ist auch hier wieder der erste Operand mit dessen Anzahl an *Topics* entscheidend für die Laufzeit. Dieser Umstand verbirgt ein geringfügiges Verbesserungspotential, denn es könnte direkt das Mapping für die Auswahl der *Topics* genutzt werden.

Szenario 2

Opera ∘ Opera

∘	**TD**	**TS**	**AD**	**AS**
naiver Alg.	15.105,3	15.206,6	337.254,7	15.132,5
TMDiff-Alg.	102,5	101,8	35.616,3	103,7
Faktor	147,37	149,38	9,47	145,93

Tabelle 4.5.: Szenario 2: Zeit in ms für das Berechnen der Operationen für Opera

ToyTM ∘ ToyTM

∘	**TD**	**TS**	**AD**	**AS**
naiver Alg.	127,7	127,5	377,4	123,8
TMDiff-Alg.	18,7	18,6	113,6	18,7
Faktor	6,83	6,85	3,32	6,62

Tabelle 4.6.: Szenario 2: Zeit in ms für das Berechnen der Operationen für ToyTM

Hier bestätigt sich ganz eindeutig die eingangs formulierte Vermutung, dass der Verbesserungsfaktor stark von der Eingabegröße abhängt. Die beiden Tabellen bestätigen nun auch experimentell die Verbesserung der Laufzeitklasse der TMDiff-Algorithmen. Während für zwei gleiche, kleine *Topic Maps* geringe Verbesserungsfaktoren vorliegen (zwischen drei und sieben), liegt für zwei größere *Topic Maps* der Faktor höher (zwischen neun und 150). **TD** und **TS** liefern erwartungsgemäß gute Verbesserungsfaktoren. Da für **AS** bei zwei gleichen Eingabe-*Topic Maps* sofort eine entsprechende *Association* gefunden wird, ist diese Berechnung ebenfalls um einen ähnlichen Faktor schneller als **TD** und **TS**.

Szenario 3

Opera ∘ Opera(mod)

∘	**TD**	**TS**	**AD**	**AS**
naiver Alg.	15.890,8	15.772,4	451.111,4	17.506,9
TMDiff-Alg.	103,7	105,1	45.603,6	118,7
Faktor	153,24	150,07	9,89	147,49

Tabelle 4.7.: Szenario 3: Zeit in ms für das Berechnen der Operationen für Italien Opera und Italien Opera(modifiziert)

Opera(mod) ∘ Opera

∘	**TD**	**TS**	**AD**	**AS**
naiver Alg.	15.588,5	15.194,7	459.579,8	16.249,2
TMDiff-Alg.	103,6	106,2	52.640	112,1
Faktor	150,47	143,08	8,73	144,95

Tabelle 4.8.: Szenario 3: Zeit in ms für das Berechnen der Operationen für Italien Opera(modifiziert) und Italien Opera

Auch im dritten Szenario sind merklich Verbesserungen zu verzeichnen. Die besten Verbesserungsfaktoren haben die Topic-Operatoren und der **AS**. Bei der **AD** müssen wieder mehr *Topics* und *Associations* betrachtet werden, um sicherzugehen, dass keine gleiche *Association* in der zweiten *Topic Map* vorhanden ist.

Zusammenfassend lässt sich sagen, dass die entworfenen Algorithmen effizient und korrekt arbeiten. `TMDiff` lässt sich einfach in bestehende Java-Anwendungen über das Interface und für alle anderen Anwendungen über die definierten Austauschformate integrieren. Es sind skalierende Algorithmen entwickelt worden, welche den praktischen Einsatz der Operationen auch für große *Topic Maps* ermöglichen.
Im nächsten Kapitel erfolgt die Zusammenfassung der Arbeit und deren Ergebnisse und es wird ein Ausblick gegeben, welche Anwendungs- und Forschungsmöglichkeiten sich aus den entwickelten Operationen und deren Umsetzung ermöglichen.

5. Ausblick und Zusammenfassung

In diesem Kapitel wird die Arbeit zusammengefasst. Es wird ein Ausblick für zukünftige Arbeiten gegeben, welche auf der hier erstellten Basis aufbauen können. Anschließend wird diskutiert, ob die zu Beginn gesteckten Ziele erreicht und die Fragestellung beantwortet wurde.

5.1. Ausblick

Das effektive Erkennen von strukturellen und atomaren Unterschieden von Wissensbasen in Form von Topic Maps bietet eine Vielzahl von direkten und indirekten Anwendungsmöglichkeiten. Die offensichtlichste Verwendungsmöglichkeit besteht im direkten Anwenden der hier vorgestellten Implementierung. Es können nun auf einfache Art und Weise Unterschiede und Gemeinsamkeiten aufgezeigt werden. Aufbauend hierauf ist es nun möglich, statistische Analysen durchzuführen. Beispielsweise könnte ein Maß der Unterschiedlichkeit von zwei *Topic Maps* eingeführt werden. Dadurch ließen sich *Topic Maps* nach Ähnlichkeit clustern. Eine solche Clusterung könnte weitergehende Analysen erleichtern, da jetzt Daten in einer Referenz-*Topic Map* mit Daten von ähnlichen *Topic Maps* (aus einer großen Menge von *Topic Maps*) verglichen werden können.

Eine weitere Anwendungsmöglichkeit wäre das Versionsmanagement von *Topic Maps*. Wird eine *Topic Map* dezentral verändert und sollen diese Veränderungen nun zentral zusammengeführt werden, so muss nur die Differenz der *Topic Maps* und der jeweiligen *Topics* berechnet werden. Hieraus können nun einfach Einfüge- und Lösch-Operationen generiert werden. Dieses System funktioniert auch dann, wenn mehrere Parteien eine *Topic Map* dezentral bearbeiten. Einfache TMQL-Statements würden beispielsweise nicht ausreichen, sobald Konflikte in den Änderungen der einzelnen Parteien auftreten (z.B., wenn zwei Parteien das gleiche *Topic* löschen). Beim Durchführen der Operationen werden solche Probleme erkannt und es wird eine Konfliktauflösung möglich.

5.2. Zusammenfassung

Das Ziel der Arbeit war es ein Verfahren zu entwickeln, um Unterschiede und Gemeinsamkeiten von *Topic Maps* erkennen und effizient berechnen zu können. Dazu erfolgte eine mathematische Formalisierung der *Topic Maps* und deren Konstrukte. So konnten Operatoren definiert werden, welche das Erkennen von Unterschieden und Gemeinsamkeiten einfach ermöglichen. Die definierten Operatoren sind ähnlich zu den ebenfalls vorgestellten mengentheoretischen Operatoren vielseitig verwendbar und liefern Ergebnisse, die mit der hier gegebenen Formalisierung der Konstrukte des TMDM konsistent sind. Sie generieren also ein Ergebnis, dessen Elemente einer konkret definierten Semantik unterliegen, nämlich derselben Semantik, die über das TMDM für die Operanden (*Topic Maps* bzw. *Topics*) der Operationen festgelegt ist. Dies unterscheidet die Arbeit auch in großem Maße von bereits bestehenden Umsetzungen, welche Unterschiede ohne semantischen Mehrwert herausfiltern [Kiv11].

Anschließend wurde ein prototypisches Framework entwickelt, welches die spezifizierten Operationen berechnet. Die Prämissen der Umsetzung waren Korrektheit gegenüber der genannten Spezifikation (Definition der Operatoren Kap. 3.2.2) und Effizienz im Sinne der Laufzeit (s. Abschn. 4.2.1). Das korrekte Verhalten der umgesetzten Operationen wurde durch ein strukturiertes Testkonzept sichergestellt. Der Algorithmus für die Berechnung und dessen Laufzeit wurden ausführlich im Abschnitt 4.1.3 analysiert und vorgestellt.

Abschnitt 4.5 verdeutlicht noch einmal die auf die Performance bezogenen Vorteile der angewandten Algorithmen gegenüber der vorgestellten naiven Implementierung (Abschn. 4.1.3). Es wurden vorher theoretische Laufzeituntersuchungen durchgeführt. Die Laufzeiten für das Berechnen der Operationen für die ausgewählten Beispiel - *Topic Maps* bestätigen die vorangegangenen Untersuchungen.

Zusammenfassend lässt sich sagen, dass die zu Beginn der Arbeit formulierten Fragen beantwortet und mit Hilfe der vorgestellten formal definierten Operatoren ein Werkzeug konzipiert wurde, welches über das eigentliche Ziel der Arbeit hinaus (s. Abschn. 5.1) einen großen Mehrwert für das Arbeiten mit mehreren *Topic Maps* bietet und neue Anwendungsmöglichkeiten eröffnet. Mit dem vorgestellten Framework `TMDiff` ist eine effiziente und leicht benutz- und erweiterbare Umsetzung der Operatoren gelungen.

Literaturverzeichnis

[Bar07] Robert Barta. Towards a Formal TMQL Semantics. In Lutz Maicher, Alexander Sigel, and Lars Garshol, editors, *Leveraging the Semantics of Topic Maps*, volume 4438. Springer Berlin / Heidelberg, 2007.

[BIO] GUID and Life Sciences Identifiers Applicability Statements. `http://www.tdwg.org/standards/150/`.

[CD10] Anthony B. Coates and Daniel Dui. "Full Impact" Schema Differencing. *XML Prague Conference Proceedings*, pages 65–86, 2010.

[Fel12a] Stephan Felke. TMDiff - Framework for calculating a diff of two topic maps - Console Application. `http://code.google.com/p/topicmapdiff/downloads/detail?name=TMDiff-0.0.1-SNAPSHOT-jar-with-dependencies.jar`, 2012. zuletzt abgerufen am 14.03.2012.

[Fel12b] Stephan Felke. TMDiff - Framework for calculating a diff of two topic maps - Framework. `http://code.google.com/p/topicmapdiff/downloads/detail?name=TMDiff-0.0.1-SNAPSHOT.jar`, 2012. zuletzt abgerufen am 14.03.2012.

[Gar02] Lars Marius Garshol. What Are Topic Maps? `http://www.xml.com/pub/a/2002/09/11/topicmaps.html`, 2002. XML.com.

[Gar04] Lars Marius Garshol. Living with topic maps and RDF. `http://www.ontopia.net/topicmaps/materials/tmrdf.html`, 2004.

[Gar06] Lars Marius Garshol. tolog – A Topic Maps Query Language. *Lecture Notes in Computer Science*, 3873:183–196, 2006.

[Gar08] Lars Marius Garshol. TMCL and OWL. *TMRA 08 Proceedings*, 2008.

[GM06] Lars Marius Garshol and Graham Moore. ISO13250 Topic Maps — XML Syntax. `http://www.isotopicmaps.org/sam/sam-xtm/`, 2006.

[GM08] Lars Marius Garshol and Graham Moore. ISO13250-2 Topic Maps — Data Model. http://www.isotopicmaps.org/sam/sam-model/, 2008.

[Hau65] Felix Hausdorff. *Grundzüge der Mengenlehre.* Chelsea Publ. Co., 1914/1949/1965.

[Hau73] Dieter Haupt. *Mengenlehre leicht verständlich*, volume 6, chapter 4, pages 39–61. VEB Fachbuchverlag Leipzig, 1973.

[Hob49] Thomas Hobbes. *Grundzüge der Philosophie. Erster Teil: Lehre vom Körper. Übersetzt von Max Frischeisen-Köhler*, chapter 11, pages 111–116. Philosophische Bibliothek, Bd. 157, Leipzig, 1949.

[HR01] Theo Härder and Erhard Rahm. *Datenbanksysteme: Konzepte und Techniken der Implementierung.* Springer Verlag Berlin Heidelberg New York, 2001.

[JAV] JAVA Collection API Dokumentation. http://download.oracle.com/javase/1.4.2/docs/api/java/util/Collections.html. zuletzt abgerufen am 02.09.2011.

[JAV12] Java - Oracle. http://www.java.com, 2012. zuletzt abgerufen am 14.03.2012.

[JUN12] JUnit. http://www.junit.org/, 2012. zuletzt abgerufen am 12.03.2012.

[KFKO02] Michel Klein, Dieter Fensel, Atanas Kiryakov, and Damyan Ognyanov. Ontology Versioning and Change Detection on the Web. *Lecture Notes in Computer Science*, 2473:247–259, 2002.

[Kiv11] Aki Kivela. Wandora. http://www.wandora.org/wandora/wiki/index.php?title=Main_Page, 2011.

[Kro10] Sven Krosse. Topic Maps Query Language Session 7. http://www.slideshare.net/LEler/tmql-session7, 2010.

[Lan11] Hans Werner Lang. Algorithmen. http://www.iti.fh-flensburg.de/lang/algorithmen/asymp.htm, 2011. zuletzt abgerufen am 04.03.2012.

[Mai08] Lutz Maicher. Einführung in Topic Maps und Subject-centric Information Architecure, 2008.

[MAI11] Maiana. http://maiana.topicmapslab.de/, 2011. zuletzt abgerufen am 13.12.2011.

[Mil98] Eric Miller. An Introduction to the Resource Description Framework. *D-Lib Magazine*, 1998.

[Net11] NetworkedPlanet. Subj3ct, 2011. zuletzt abgerufen am 30. Dezember 2011.

[New01] Steve Newcomb. XML Topic Maps: Finding Aids for the Web. *IEEE Multimedia*, 2001.

[NK04] Natalya F. Noy and Michel Klein. Ontology Evolution: Not the Same as Schema Evolution. *Knowledge and Information Systems*, 6:428–440, 2004.

[Ont12] Ontopia. Ontopia Topic Maps Engine. `http://www.ontopia.net/index.jsp`, 2012. zuletzt abgerufen am 15.03.2012.

[Pep00] Steve Pepper. The TAO of Topic Maps. *XML Europe*, 2000.

[Pep11] Steve Peppper. Opera - Italien Opera Topic Map. `http://www.ontopia.net/omnigator/models/topicmap_complete.jsp?tm=opera.ltm`, 2011. zuletzt abgerufen am 15.12.2011.

[Sch11] Uta Schulze. ToyTM - Toy Topic Map. `http://maiana.topicmapslab.de/u/uta/tm/toytm`, 2011. zuletzt abgerufen am 15.12.2011.

[Sei11] Daniel Seifarth. Identitäten und Identifikatoren in Ökologischen Daten mit Topic Maps. Diploma thesis, Universität Leipzig, Leipzig, 2011.

[Sig04] Alexander Sigel. Wissensorganisation, Topic Maps und Ontology Engineering: Die Verbindung bewährter Begriffsstrukturen mit aktueller XML-Technologie. In Christoph Lehner, H. Peter Ohly, and Gerhard Rahmstorf, editors, *Wissensorganisation und Edutainment: Wissen im Spannungsfeld von Gesellschaft, Gestaltung und Industrie*, volume 7, pages 185–193. ERGON Verlag Würzburg, 2004.

[Tho09] Dr. Andreas Thor. Vorlesung Datenintegration Vorlesungseinheit Verteilung, Autonomie und Heterogenität. `http://dbs.uni-leipzig.de/de/stud/ss2009/dataint`, 2009.

[TMA12a] tmapix - Utilities for TMAPI compatible Topic Maps engines. `http://code.google.com/p/tmapix/`, 2012. zuletzt abgerufen am 12.03.2012.

[TMA12b] Topic Maps API. `http://tmapi.org/2.0/`, 2012. zuletzt abgerufen am 12.03.2012.

[TML11] Topic Maps Lab, Universität Leipzig. http://www.topicmapslab.de/, 2011. zuletzt abgerufen am 12.03.2012.

[Ver04] Cristina Vertan. Resource Description Framework (RDF), 2004.

[Wik11a] Wikipedia. Hypergraph — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=Hypergraph&oldid=476364204, 2011. zuletzt abgerufen am 24.11.2011.

[Wik11b] Wikipedia. Identität (Logik) — Wikipedia, Die freie Enzyklopädie. http://de.wikipedia.org/w/index.php?title=Identit%C3%A4t_(Logik)&oldid=97570335, 2011. zuletzt abgerufen am 30. Dezember 2011.

[Wik11c] Wikipedia. Uniform Resource Identifier — Wikipedia, Die freie Enzyklopädie. http://de.wikipedia.org/w/index.php?title=Uniform_Resource_Identifier&oldid=97290923, 2011. zuletzt abgerufen am 05.12.2011.

[Wik12a] Wikipedia. Big O notation — Wikipedia, The Free Encyclopedia, 2012. zuletzt abgerufen am 11. März 2012.

[Wik12b] Wikipedia. Landau-Symbole — Wikipedia, Die freie Enzyklopädie, 2012. zuletzt abgerufen am 11. März 2012.

[Wik12c] Wikipedia. Mengenlehre — Wikipedia, Die freie Enzyklopädie. http://de.wikipedia.org/w/index.php?title=Mengenlehre&oldid=96767345, 2012. zuletzt abgerufen am 14.01.2012.

A. Anhänge

A.1. Minimalbeispiel

```
1  public void topicDiff() throws Exception{
2    //the base Locators the new maps should have
3    String baseLocator1 = "base1";
4    String baseLocator2 = "base2";
5
6    //the file locations of the topic maps
7    String fileLocation1 = "xtm-topicmap-1";
8    String fileLocation2 = "xtm-topicmap-2";
9
10   TopicMapSystemFactory fact = TopicMapSystemFactory.newInstance();
11   TopicMapSystem sys = fact.newTopicMapSystem();
12
13   //create empty maps:
14   TopicMap map1 = sys.createTopicMap(baseLocator1);
15   TopicMap map2 = sys.createTopicMap(baseLocator2);
16
17   //read and paste the xtm-topic-map-files into the empty maps
18   XTMTopicMapReader xtmr1 = new XTMTopicMapReader(map1 , new
         File(fileLocation1));
19   XTMTopicMapReader xtmr2 = new XTMTopicMapReader(map2 , new
         File(fileLocation2));
20   xtmr1.read();
21   xtmr2.read();
22
23   //call TMDiff for the result of any operation
24   TMDiff diff = new TMDiffImpl();
25   Set<Topic> resultTopics = diff.semanticTopicDiff(map1, map2);
26 }
```

A.2. Ausschnitt eines Ergebnisses eines atomaren Topic-Diff

```
<result>
  ...
  <topic>
    <reference si="http://en.wikipedia.org/wiki/Belgium"/>
    <si>http://en.wikipedia.org/wiki/Belgium</si>
    <si>http://dbpedia.org/resource/Belgium</si>
    <name>
      <parent si="http://en.wikipedia.org/wiki/Belgium"/>
      <type si="http://psi.topicmaps.org/iso13250/model/topic-name"/>
      <value>Kingdom of Belgium</value>
      <scope si="http://psi.oasis-open.org/iso/639/#eng"/>
    </name>
    ...
    <variant>
      <value>Belgien</value>
      <scope si="http://en.wikipedia.org/wiki/short_name"/>
      <scope si="http://psi.oasis-open.org/iso/639/#deu"/>
      <datatype>http://www.w3.org/2001/XMLSchema#string</datatype>
      <name>
        <parent si="http://en.wikipedia.org/wiki/Belgium"/>
        <type
            si="http://psi.topicmaps.org/iso13250/model/topic-name"/>
        <value>Koenigreich Belgien</value>
        <scope si="http://psi.oasis-open.org/iso/639/#deu"/>
      </name>
    </variant>
    ...
    <occurrence>
      <parent si="http://en.wikipedia.org/wiki/Belgium"/>
      <type si="http://en.wikipedia.org/wiki/Population"/>
      <value>10665867</value>
      <scope si="http://en.wikipedia.org/wiki/Estimation"/>
      <scope si="http://en.wikipedia.org/wiki/2008"/>
      <datatype>http://www.w3.org/2001/XMLSchema#long</datatype>
    </occurrence>
    ...
  </topic>
  ...
</result>
```

A.3. Schema für das Ergebnis der atomaren Topic-Operatoren

```
<!ELEMENT result>
<!ELEMENT result (topic*)>

<!ELEMENT topic (reference)>
<!ELEMENT topic (si*)>
<!ELEMENT topic (sl*)>
<!ELEMENT topic (ii*)>
<!ELEMENT topic (name*)>
<!ELEMENT topic (variant*)>
<!ELEMENT topic (occurrence*)>

<!ELEMENT name (parent)>
<!ELEMENT name (type)>
<!ELEMENT name (value)>
<!ELEMENT name (scope*)>

<!ELEMENT variant (value)>
<!ELEMENT variant (datatype)>
<!ELEMENT variant (name)>
<!ELEMENT variant (scope*)>

<!ELEMENT occurrence (parent)>
<!ELEMENT occurrence (type)>
<!ELEMENT occurrence (value)>
<!ELEMENT occurrence (datatype)>
<!ELEMENT occurrence (scope*)>

<!ELEMENT value (#CDATA)>
<!ELEMENT datatype (#CDATA)>

<!ATTLIST parent   si CDATA sl CDATA ii CDATA >
<!ATTLIST type     si CDATA sl CDATA ii CDATA >
<!ATTLIST scope    si CDATA sl CDATA ii CDATA >
<!ATTLIST reference    si CDATA sl CDATA ii CDATA >
```

Selbständigkeitserklärung

Der Verfasser erklärt an Eides statt, dass er die vorliegende Arbeit selbständig, ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt hat. Die aus fremden Quellen (einschließlich elektronischer Quellen) direkt oder indirekt übernommenen Gedanken sind ausnahmslos als solche kenntlich gemacht. Die Arbeit ist in gleicher oder ähnlicher Form oder auszugsweise im Rahmen einer anderen Prüfung noch nicht vorgelegt worden.

Leipzig, 28. März 2012 Stephan Felke