Universität Siegen
Fachbereich 12
Elektrotechnik und Informatik
Fachgruppe Praktische Informatik
Hölderlinstr. 3
D-57068 Siegen

SiDiff 2.0: Vergleichsfunktionen

Pit Pietsch, Timo Kehrer

11. März 2009

Inhaltsverzeichnis

Document History

Date	Changes
17.02.09	Initial Version
02.03.09	Beschreibung zur Erstellung von Testmodellen hinzugefügt (initial)
08.03.09	New todos and comments added
11.03.09	Ein Dokument für alle Compare-Funktionen angelegt (initial)

TODOs and Comments

- Strengere Konventionen für die Bezeichnung von CompareFunctions?
  Bsp.: ParentsEqualType und ChildrenEqualViewingTypes. Beide Compare-Functionen meinen doch dasselbe, oder? Entweder mit oder ohne Viewing (ganz generell). (tk)
- ValueAdmeasue aufräumen (pp)
- Warum existiert compareStringViewingSimilarityUsingLCSIgnoringCase/...ConsideringCase aber keine vergleichbare trennung bei IndexOf? (pp)
- Wir sollten spezifizieren und überprüfen (und dokumentieren) für welche eTypes eine comparefunction anwendbar ist (daraus ergeben sich ja auch die aequivalenzklassen zum testen). die NumericAttributEquals...-funktionen haben einen derartigen mechanismus bereits hartverdrahtet implementiert. (pp)
  tk kümmert sich darum (tk)
  Zusätzliche Anmerkung (tk): Eine, wie ursprünglich angedacht, einfache Matrix (CompareFunktion X AttributTyp) ist unter Umständen nicht ausreichend. Im Prinzip müssen hier auch noch die Parameter als 3.te Dimension aufgenommen werden. Oder sollen wir eben eine solche Kompatibilitäts-Matrix auch noch für Parameter erstellen?
- Was passiert bei fehlerhaften regulaeren Ausdruecken? Gibt es die ueberhaupt, oder heisst falsch lediglich, dass der RegEx nicht der Intention des Benutzers entspricht? (tk)
- macht es sinn neben den aequvialenzklassen die im moment die ausgangsituation für die testergebnisse festlegt auch ergebnissorientierte aequivalenzklassen zu definieren? beispielsweise: erzeuge eine similarity von 0.5 mit vergleichsfunktion x,y? (pp)
- ExplicitMatch/MaximumSimilarity: brauchen wir beide, oder kann explicitMatch nicht einfach auch weg? (pp)
  ExplicitMatch wird rausgenommen. Evtl späters name-refactoring (pp/tk)
  -erledigt
  Vorschlag für Name-Refactoring: Maximum-Similarity und MinimumSimilarity evtl. nicht sofort intuitiv einleuchtend? Wie wäre es mit ExplicitMatch und ExplicitDismatch? (tk)
- Vergleich von Attributen: Wird hier schon unterstellt, dass die beiden zu vergleichenden Elemente vom selben Typ sind? Haette Auswirkungen auf die Aequivalenzklassen...(tk)
  typgleichheit ist über die dedicatedClass sichergestellt.(pp)
  -erledigt
- equalID kann als comparefunction doch eigentlich entfallen: spezialfall von compareAttributeUsingEquals? (pp)
  equalID wird rausgenommen (pp/tk)
  -erledigt
- parameter: der inhalt eines parameter-strings ist abhängig von dem typ des zu vergleichenden attributs. die verarbeitung des parameters wird in der abstracten typ oberklasse definiert. beispiel: StringAttributeUsingEquals erbt das wissen um die Verarbeitung von abstract-StringAttribute, wo festgelegt wird, wie der Parameter auszulesen ist. dieser mechanismus steht ja bei unser idee so nicht mehr zur verfügung... gleiches gilt für den boolean parameter sensitive.
  evtl. auslagern der parameterfunktionalität? eine utilklasse der man parameter + eType übergibt, die ihn entsprechend zerlegt und das Ergebniss zurückliefert? (pp)

  lösung: AbstractAttribute zerlegt parameter und behandelt alle parameter die auf dieser ebene relevant sind. weitere parameter werden in einer liste gespeichert und können in der

1 Einleitung

Um eine SiDiff-Compare-Konfiguration zu erstellen oder zu modifizieren, ist zunächst eine ausreichende Kenntnis der verfügbaren Vergleichsfunktionen unausweichlich. An dieser Stelle soll nun eine Dokumentation der im SiDiff-Kern implementierten Vergleichsfunktionen folgen.

Struktur der Dokumentation

Die Dokumentation der einzelnen Vergleichsfunktionen ist dabei in mehrere, kurze Unterabschnitte gegliedert. Dies beinhaltet eine Unterteilung in die Sektionen *Vorbedingungen*, *Semantik* und folgend *Rückgabewert*, *Parameter* und gegebenenfalls Querverweise zu anderen, ähnlichen Vergleichsfunktionen.
In den einzelnen Unterabschnitten – jedoch vorwiegend unter *Vorbedingungen* – werden Ausnahmetypen angeführt, die bei fehlerhafter Anwendung der Vergleichsfunktion bzw. möglicherweise auch fehlerhaftem Datenbestand ausgelöst werden können.
Für jede Vergleichsfunktion werden zudem Testfälle spezifiziert, um die Vergleichsfunktion in einem Black-Box Testverfahren auf ihre Korrektheit zu prüfen. Hinsichtlich der Testdaten einer Vergleichsfunktion sind zwei verschiedene Arten zu unterscheiden: Modelldaten und Parameter. Zum derzeitigen Zeitpunkt ist dieses Dokument ein lebendes Dokument, welches sich im Aufbau befindet und ständigen Änderungen unterworfen ist. Die Vorgehensweise zur Erstellung der Testdaten sollte sich dabei in zwei Schritte untergliedern: Die Identifikation verschiedener Äquivalenzklassen von Testdaten und die Wahl geeigneter Repräsentanten mit welchen die Tests durchgeführt werden sollen.

Hinweise zur Erstellung von Testmodellen finden sich in Kapitel **??**. Hinweise zur Durchführung der Unit-Tests sind in Kapitel **??** zu finden.

Sofern sich Gruppierungen der Vergleichsfunktionen nach ähnlicher Semantik treffen lassen, erfolgt zu Beginn eines jeden neuen Gruppierungs-Abschnitts eine Auflistung der Spezifikationen, die sich allgemein über diese Klasse von Funktionen aussagen lassen. Eine spezifischere Erläuterung der einzelnen Vergleichsfunktionen lässt sich darauffolgend in den sich anschließenden Unterabschnitten, in denen die Funktionen nacheinander dokumentiert werden, finden.

2 Testdatenerstellung

Im Hinblick auf die Testdaten zum Testen der Compare-Funktionen sind zwei Kategorien von Testdaten zu unterscheiden: Testmodelle und Parameter der jeweiligen Compare-Funktion. Die in diesem Abschnitt beschriebene Erstellung von Testdaten bezieht sich dabei auf die Erstellung von Testmodellen.

2.1 Testmetamodell

Für die Erstellung von Testmodellen wurde ein eigenes Metamodell, im Folgenden als Testmetamodell bezeichnet, entworfen (Bundle `org.sidiff.common.testmetamodel`). Das Testmetamodell ist dabei möglichst schlank gehalten, beinhaltet aber dennoch die für reale Metamodelle typischen Strukturen: Komponenten (Blöcke) und gerichtete Beziehungen (Linien) zwischen den Komponenten. Zudem wurde darauf geachtet, alle primitiven EMF-Datentypen zu verwenden. Abbildung 2.1 zeigt das Testmetamodell in der bekannten Ecore-Diagramm-Notation.

2.2 Erstellung von Testmetamodell-Instanzen

Für die Erstellung von Testmetamodell-Instanzen exisitieren verschiedene Möglichkeiten, von denen hier zwei näher betrachtet werden sollen:

1. Mittels des durch EMF bereit gestellten, generischen Ecore-Editors. Hat den Nachteil, dass es teilweise mühsam sein kann, die Graph-Struktur der Modelle zu erfassen. Das gilt sowohl für die Erstellung der Testdaten, als auch für die Nachvollziehbarkeit der einzelnen Testfälle.
2. Instanzen werden in UML-Notation mittels eines UML-Tools spezifiziert und über einen Konverter nach EMF transformiert.

Aus Gründen der Nachvollziehbarkeit und der leichteren Erstellung von Testmodellen werden wir die 2.te Variante anwenden. Als UML-Werkzeug ist der IBM Rational Software Modeler (RSM) in der Version 7.5 zu benutzen. Im Folgenden werden die einzelnen Schritte zur Erstellung von Testmodellen beschrieben.

2.2.1 Modellerstellung in RSM

Alle Testmodelle sind im RSM zu erstellen. Hierzu existieren zwei RSM-Projekte, TestmetamodelProfile und TestModels, welche im Verzeichnis `rsm/workspace` des Bundles `org.sidiff.common.testmodels` zu finden sind. Das Projekt TestmetamodelProfile definiert einige zur Identifikation von Komponenten benötigte Stereotypen (s. unten). Die eigentlichen Testmodelle werden im Projekt TestModels spezifiziert.

Pakethierarchie

Testmodelle werden in einer Pakethierarchie organisiert, welche folgenden Konventionen genügt (s. Abbildung 2.2.1):

- Auf oberster Ebene befindet sich das Paket, welches die JUnit-Testfälle beinhaltet. Das Paket heißt wie das jeweilige OSGI-Bundle (z.B. org.sidiff.compare.comparefunctions.emf.test)
- Auf der nächsttieferen Ebene befindet sich das Paket, welche alle Testdaten für eine spezielle Compare-Funktion Test-Suite (z.B. CompareAttributeUsingLCSTest) beinhaltet.
- Auf der nächsten Hierarchie-Ebene befinden sich Pakete für alle Testfälle. Diese Pakete werden entsprechend dem Namensmuster testcase$_{<}$NR. $>$ benannt, wobei $<$ NR. $>$ einer sequenziell hochgezählten Nummer entspricht.

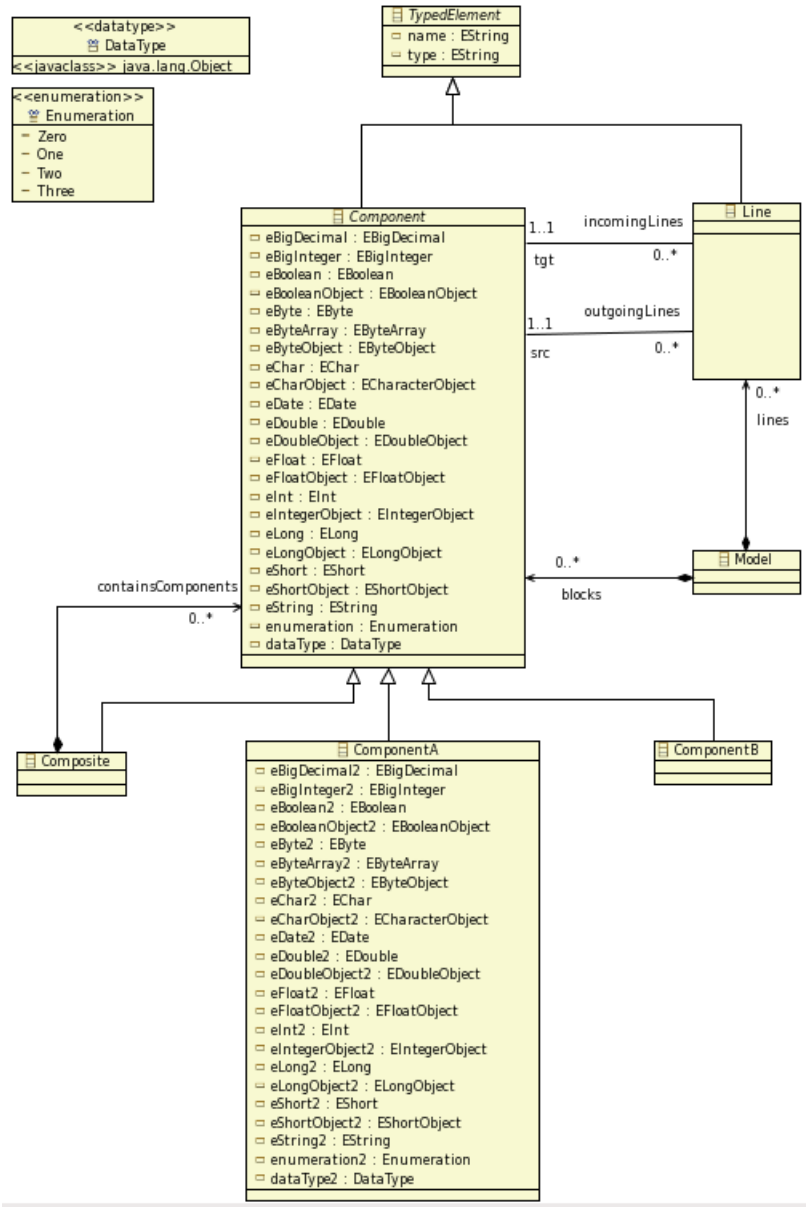

Abb. 2.1: Testmetamodell

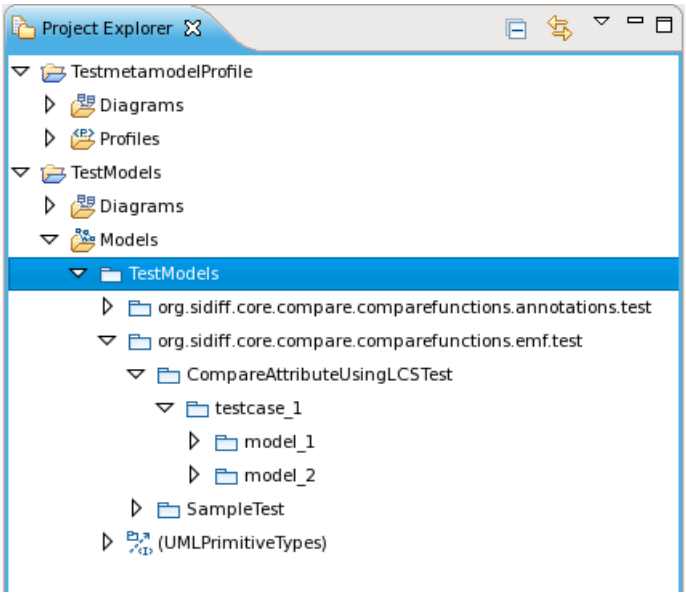

Abb. 2.2: Exemplarische Paket-Hierarchie

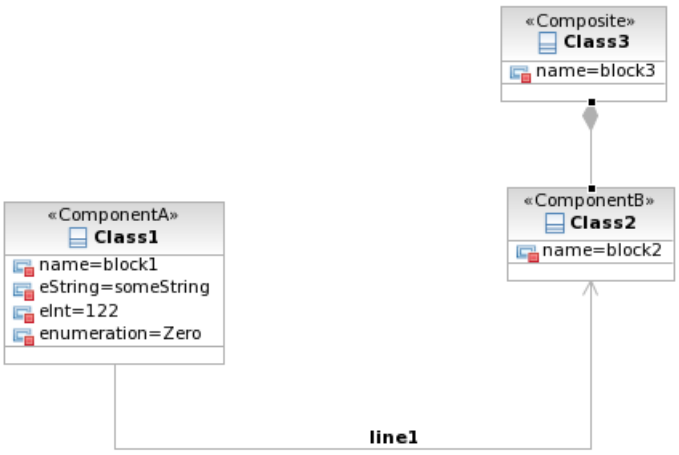

Abb. 2.3: Exemplarisches Testmodell in UML-Klassendiagramm Notation

- Für jeden Testfall werden schließlich die Eingabemodelle (in der Regel zwei) spezifiziert, welche sich in Unterpakten model-1 bzw. model-2 befinden.

Spezifikation der eigentlichen Testmodelle

Die eigentlichen Testmodelle werden in den oben beschriebenen Paketen model-1 bzw. model-2 spezifiziert. Zur grafischen Visualisierung ist ein Klassendiagramm zu verwenden. Abbildung 2.2.1 zeigt ein Exemplarisches Testmodell in Klassendiagramm-Notation, Abbildung 2.2.1 die dazugehörige Modellstruktur, wie sie im RSM im Projekt-Explorer dargestellt wird.

Die im Testmetamodell definierten Metaklassen werden durch folgende UML-Konstrukte umgesetzt:

- ComponentA: Entspricht einer Klasse mit dem Stereotyp ComponentA (welcher in TestmetamodelProfile als Erweiterung der UML-Metaklasse Class definiert ist).
- ComponentB: Entspricht einer Klasse mit dem Stereotyp ComponentB (welcher in TestmetamodelProfile als Erweiterung der UML-Metaklasse Class definiert ist).

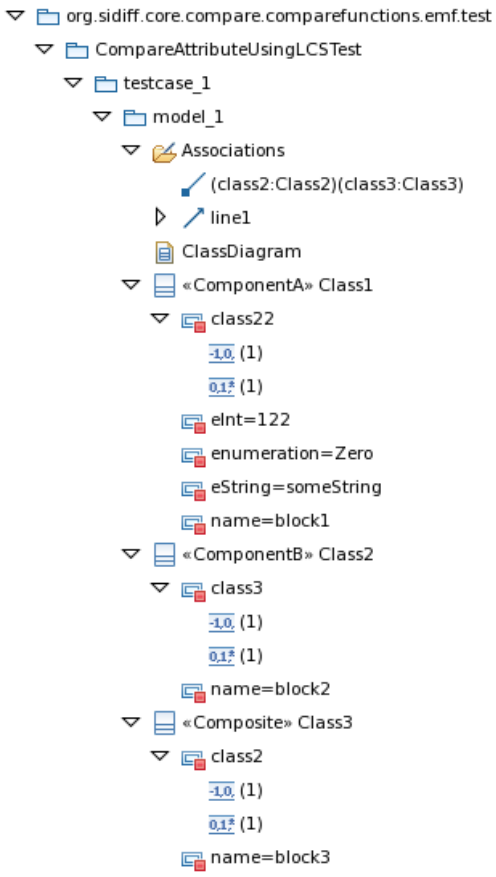

Abb. 2.4: Exemplarische Modellstruktur

- Composite: Entspricht einer Klasse mit dem Stereotyp Composite (welcher in TestmetamodelProfile als Erweiterung der UML-Metaklasse Class definiert ist). Die Beziehung containsComponent ist durch eine Kompositionsbeziehung umzusetzen.
- Line: Lines werden durch gerichtete Assoziationen abgebildet. Rollennamen und Kardinalitäten brauchen nicht spezifiziert zu werden.

Einschränkungen und Konventionen

Folgende Einschränkungen und Konventionen sind zu beachten:

- Attribute: Die Zuordnung von durch das Testmetamodell definierten Attributen und Attributwerten geschieht über die Benennung eines UML-Attributs nach folgendem Muster: $< attr-name >=< attr-value >$, wobei $< attr-name >$ dem Namen eines Attributs im Testmetamodell und $< attr-value >$ dem dazugehörigen Attributwert entspricht. Für einen Testfall nicht relevante Attribute müssen nicht spezifiziert zu werden.
- Eindeutige Identifizierer: Modellelemente müssen eindeutig identifizierbar sein. Für alle Komponenten (ComponentA, ComponentB, Composite) ist daher das Attribut name unbedingt anzugeben. Auch Lines müssen identifizierbar sein. Als eindeutiger Identifizierer dient hier der Assoziationsname.

2.2.2 *Modellexport und Generierung von EMF-Modellen*

Um aus den im RSM erstellten UML-Modellen EMF-Modelle als die eigentlichen Testdaten der JUnit-Testfälle zu erhalten, sind diese zunächst im Eclipse UML2-Format zu exportieren und anschließend durch einen Konverter nach EMF zu transformieren. Beide Schritte werden im Folgenden beschrieben.

Modellexport

Der Modellexport untergliedert sich in die folgenden Schritte:

Schritt 1: Rechtsklick auf das RSM-Projekt TestModels > Export.

Schritt 2: Auswahl der Kategorie Other > UML 2.1 Model (s. Abb. 2.2.2).

Schritt 3: Auswahl des zu exportierenden Modells und des Zielverzeichnisses, für welches das Verzeichnis `rsm/export` im Bundle `org.sidiff.common.testmodels` zu wählen ist. **Wichtig:** Unbedingt die Option Export applied profiles aktivieren! (s. Abb. 2.2.2).

Generierung von EMF-Modellen

Um EMF-Modelle zu generieren, muss eine OSGI-Konsole mit dem Bundle org.sidiff.common.testmodels gestartet werden. Dieses nimmt ein Kommando generate $< WORKSPACE-URI >$ entgegen, wobei $< WORKSPACE-URI >$ den Betriebssystem-absoluten Pfad zu dem für die SiDiff-Entwicklung genutzten Eclipse-Workspace darstellt. Um die ständige Eingabe dieses Parameters zu vermeiden, kann in der Klasse `TestdataGeneratorCommandProvider` im Bundle und gleichnamigen Paket `org.sidiff.common.testmodels` ein neues Kommando spezifiziert werden, welches die private Methode generate(String workspaceUri) aufruft. Beispiel:

```
1 public void _generateTK(CommandInterpreter commandInterpreter) throws
      Exception {
2   generate("/home/kehrer/projekte/sidiff/workspaces/eclipse-ws-new");
3 }
```

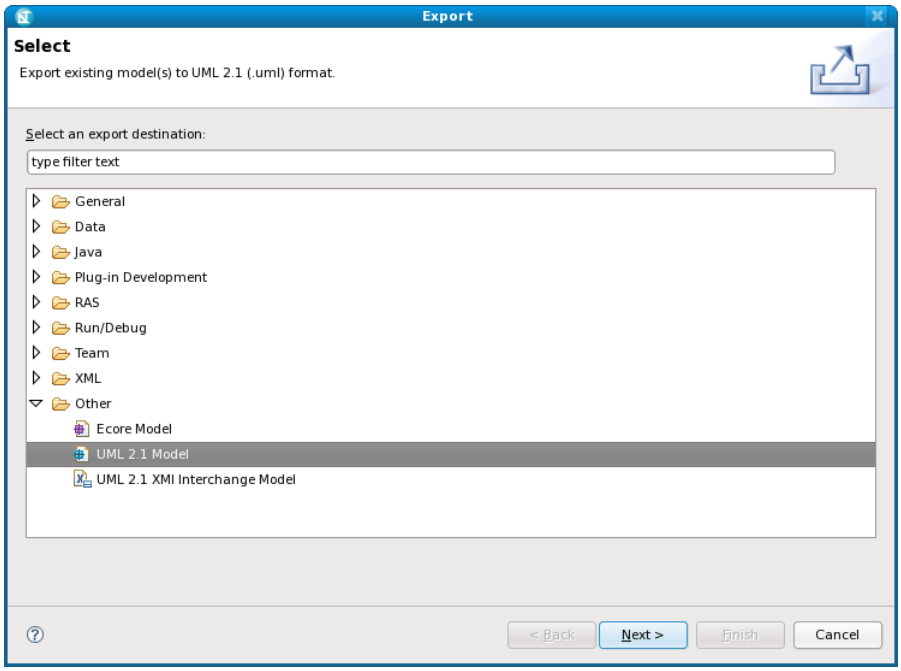

Abb. 2.5: Export Wizard (1)

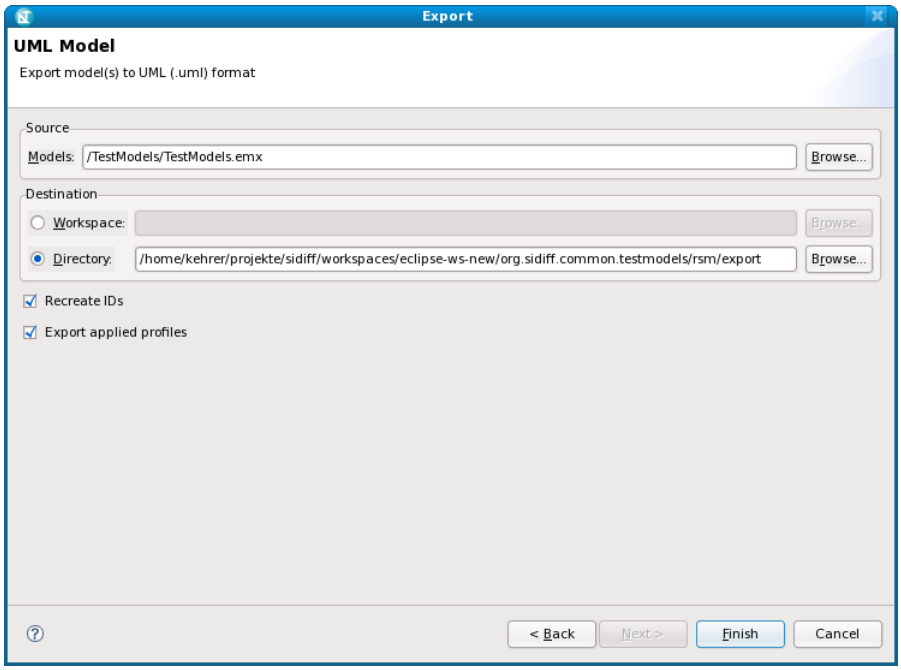

Abb. 2.6: Export Wizard (2)

Die generierten EMF-Modelle werden in den jeweiligen Bundles zum Testen der Compare-Funktionen abgelegt.

3 Einrichten und Ausführen der Unit-Tests

Checkout der relevanten Bundles. Für die Unit-Tests der EMF Funktionen sind dies beispielsweise

- org.sidiff.commons
- org.sidiff.common.emf
- org.sidiff.common.junit
- org.sidiff.common.services
- org.sidiff.common.testmetamodel
- org.sidiff.core.compare.comparefunctions
- org.sidiff.core.compare.comparefunctions.emf
- org.sidiff.core.compare.comparefunctions.emf.test

Starten des Activators in org.sidiff.common.junit als neue OSGi Framework Run-Konfiguration.

Reiter Bundles: Auswahl der benötigten bundles (i.d.R. Workspace + required Bundles)

Reiter Arguments: Wichtig! Das Working Directory auf den lokalen Workspace setzen. Dies ist nötig damit die zu testenden Models später über relative Pfade gefunden werden können!

Einstellungen übernehmen und starten. In der OSGi Konsole sampleTest zum Starten des Beispiels tippen und Return drücken...

4 Dokumentation der Vergleichsfunktionen

An dieser Stelle seien nun die einzelnen vom SiDiff-Kern bereitgestellten Vergleichsfunktionen dokumentiert. Die Vergleichsfunktionen werden dabei anhand ihrer Semantiken gruppiert oder, sofern diese sich keiner allgemeinen Gruppe zuordnen lassen, dem folgenden Abschnitt 4.1 ‚Verschiedenartige Vergleichsfunktionen' zugeschrieben.

4.1 Verschiedenartige Vergleichsfunktionen

In diesem Abschnitt werden verschiedene Vergleichsfunktionen angeführt, die sich keiner größeren allgemeinen Gruppe zuordnen lassen.

4.1.1 `EqualID`

Vorbedingungen

Für die Eingabeknoten müssen IDs existieren.

Semantik

Die Eingabeknoten werden anhand der ID-Attribute miteinander verglichen.

Rückgabewert

Für übereinstimmende IDs wird 1 und für unterschiedliche 0 zurückgegeben.

Parameter

Keine

4.1.2 `ExplicitMatch`

Vorbedingungen

Keine

Semantik

Es wird *kein* expliziter Vergleich durchgeführt. Diese Funktion liefert unabhängig von den übergebenen Eingabeknoten immer 1.

Rückgabewert

Es wird *immer* 1 als Similarity zurückgegeben.

Parameter

Keine

Abgefangene Fehler

Sind für den SiDiff-Kern Debug-Ausgaben aktiviert, so kann im Ausnahmefall eine `SiDiffRuntimeException` ausgelöst werden.

4.1.3 MaximumSimilarity

Vorbedingungen

Keine

Semantik

Es wird *kein* expliziter Vergleich durchgeführt. Diese Funktion liefert unabhängig von den übergebenen Eingabeknoten immer 1.

Rückgabewert

Es wird *immer* 1 als Similarity zurückgegeben.

Parameter

Keine

4.1.4 NoSimilarity

Vorbedingungen

Keine

Semantik

Es wird *kein* expliziter Vergleich durchgeführt. Diese Funktion liefert unabhängig von den übergebenen Eingabeknoten immer 0.
Es handelt sich um eine Vergleichsfunktion, mit der sich Knotentypen vom Vergleich ausschließen lassen.

Rückgabewert

Es wird *immer* 0 als Similarity zurückgegeben.

Parameter

Keine

4.2 `StringAttribute`...

Dieser Abschnitt behandelt Vergleichsfunktionen des Typs `StringAttribute`, mit Hilfe derer sich die Zeichenketten von Attributen der aktuellen Eingabeknoten vergleichen lassen. Allgemein lassen sich bereits folgende Angaben zu dieser Gruppe von Vergleichsfunktionen machen.

Allgemeine Vorbedingungen

Für den Fall, dass explizit ein einzelnes Attribut zum Vergleich angegeben wird, muss dieses existieren. Ist dies nicht der Fall erfolgt die Auslösung einer Ausnahme des Typs `AttributeNotExistsException` oder `AnnotationNotExistsException` im Fall eines zu vergleichenden virtuellen Attributs. Eine `AttributeNotExistsException` wird auch ausgelöst, wenn der im Parameter spezifizierte reguläre Ausdruck keine zu vergleichenden Attributnamen liefert. In jedem der drei oben aufgeführten Fälle terminiert der SiDiff-Kern unverzüglich die Ausführung des Vergleichauftrags.

Allgemeine Semantik

Die Eingabeknoten werden über ein oder mehrere durch den Parameter definierte Attribute miteinander verglichen. Der Vergleich der Attributwerte erfolgt nach der in der jeweiligen Vergleichsfunktion spezifizierten Methode.
Anzumerken ist, dass sich mittels dieser Klasse von Vergleichsfunktionen auch explizit *virtuelle* Attribute miteinander vergleichen lassen. Mehr dazu unter *Allgemeiner Parameter*.

Es bestehen zwei grundsätzliche Optionen:

1. Es wird lediglich ein Attribut zum Vergleich angegeben.
2. Mittels eines regulären Ausdrucks werden mehrere Attribute spezifiziert, die für den Vergleich herangezogen oder von diesem ausgeschlossen werden sollen.

Siehe dazu auch *Allgemeine Parameter*.

Allgemeiner Rückgabewert

1. Es werden *keine* regulären Ausdrücke verwendet – also *ein* Attribut soll verglichen werden:
   - Wenn bei beiden Eingabeknoten das zu vergleichende Attribut nicht vorhanden ist, so wird 1 zurückgegeben.
   - Ist das zu vergleichende Attribut bei lediglich einem Eingabeknoten nicht vorhanden, wird 0 zurückgegeben.
   - Sind beide Attributwerte vorhanden, so wird unter Anwendung der entsprechenden Vergleichsmethode die Ähnlichkeit bestimmt und als Similarity zurückgegeben.
2. Reguläre Ausdrücke werden verwendet (i.d.R. sollen *mehrere* Attribute verglichen werden):
   - Wenn keines der zu vergleichenden Attribute existiert, wird 1 als Ähnlichkeit zurückgegeben.
   - Sofern zu vergleichende Attribute nach Anwendung des regulären Ausdrucks vorliegen, werden die Attribute einzeln miteinander verglichen und die einzelnen Ähnlichkeiten aufaddiert. Zurückgegeben wird der Quotient aus den kumulierten Einzel-Ähnlichkeiten und der Anzahl der durch den regulären Ausdruck gefundenen Anzahl an zu vergleichenden Attributen.
     Bemerkung: Der Vergleich der *einzelnen* Attribute erfolgt nach den gleichen Gesichtspunkten, wie in der Vergleichsoption *ohne* Verwendung regulärer Ausdrücke (s.o.).

Allgemeine Parameter

Als Parameter wird der Namen des zu vergleichenden Attributs übergeben.
Ebenso ist es möglich mittels eines *regulären Ausdrucks* mehrere Attribute explizit zum Vergleich anzugeben oder alle Attribute bis auf eine durch einen regulären Ausdruck definierte

Ausschlussliste zum Vergleich zu markieren. Im ersteren Fall wird dem regulären Ausdruck ein „+“ vorangestellt; im Falle einer Ausschlussliste ein „−“.

Neben den beiden oben genannten Möglichkeiten, besteht auch die Möglichkeit explizit ein *virtuelles* Attribut zum Vergleich zu markieren. Dazu wird dem Attributnamen lediglich ein $ vorangestellt. Für den Vergleich von virtuellen Attributen werden allerdings *keine* regulären Ausdrücke unterstützt, sodass immer nur ein spezielles Attribut vergleichbar ist.

4.2.1 StringAttributeUsingBoyerCI

Diese Vergleichsfunktion führt Vergleiche anhand den unter Abschnitt 4.2 ‚StringAttribute..‘ aufgeführten allgemeinen Bedingungen durch.

Spezielle Semantik und Rückgabewert

Als spezielle Vergleichsmethode wird in diesem Fall der *case-insensitive* Vergleich nach *Boyer* verwendet.

4.2.2 StringAttributeUsingBoyerCS

Diese Vergleichsfunktion führt Vergleiche anhand den unter Abschnitt 4.2 ‚StringAttribute..‘ aufgeführten allgemeinen Bedingungen durch.

Spezielle Semantik und Rückgabewert

Als spezielle Vergleichsmethode wird in diesem Fall der *case-sensitive* Vergleich nach *Boyer* verwendet.

4.2.3 StringAttributeUsingEqualsCI

Diese Vergleichsfunktion führt Vergleiche anhand den unter Abschnitt 4.2 ‚StringAttribute..‘ aufgeführten allgemeinen Bedingungen durch.

Spezielle Semantik und Rückgabewert

Als spezielle Vergleichsmethode wird in diesem Fall der *case-insensitive Equals*-Vergleich verwendet. Dies bedeutet, dass die Zeichenketten auf Gleichheit – allerdings nicht im Sinne der Groß- und Kleinschreibung – verglichen werden.

4.2.4 StringAttributeUsingEqualsCS

Diese Vergleichsfunktion führt Vergleiche anhand den unter Abschnitt 4.2 ‚StringAttribute..‘ aufgeführten allgemeinen Bedingungen durch.

Spezielle Semantik und Rückgabewert

Als spezielle Vergleichsmethode wird in diesem Fall der *case-insensitive Equals*-Vergleich verwendet. Dies bedeutet, dass die Zeichenketten auf exakte Gleichheit – auch im Sinne der Groß- und Kleinschreibung – verglichen werden.

4.2.5 StringAttributeUsingIndexOfCI

Diese Vergleichsfunktion führt Vergleiche anhand den unter Abschnitt 4.2 ‚StringAttribute..' aufgeführten allgemeinen Bedingungen durch.

Spezielle Semantik und Rückgabewert

Als spezielle Vergleichsmethode wird in diesem Fall der *case-insensitive IndexOf*-Vergleich verwendet. Dies bedeutet, dass überprüft wird, ob die Zeichenketten sich gegenseitig beinhalten. Beinhaltet bspw. Zeichenkette A des einen Attributs die Zeichenkette B des Attributs im gegenüberliegenden Knoten, so wird als Ähnlichkeit der Quotient aus der kürzeren Zeichenkette B und der längeren Zeichenkette A zurückgegeben. Die Groß- und Kleinschreibung spielen bei dieser Variante allerdings *keine* Rolle.

4.2.6 StringAttributeUsingIndexOfCS

Diese Vergleichsfunktion führt Vergleiche anhand den unter Abschnitt 4.2 ‚StringAttribute..' aufgeführten allgemeinen Bedingungen durch.

Spezielle Semantik und Rückgabewert

Als spezielle Vergleichsmethode wird in diesem Fall der *case-sensitive IndexOf*-Vergleich verwendet. Dies bedeutet, dass überprüft wird, ob die Zeichenketten sich gegenseitig beinhalten. Beinhaltet bspw. Zeichenkette A des einen Attributs die Zeichenkette B des Attributs im gegenüberliegenden Knoten, so wird als Ähnlichkeit der Quotient aus der kürzeren Zeichenkette B und der längeren Zeichenkette A zurückgegeben. Zusätzlich wird bei dieser Variante die Groß- und Kleinschreibung beachtet.

4.2.7 StringAttributeUsingLcsCI

Diese Vergleichsfunktion führt Vergleiche anhand den unter Abschnitt 4.2 ‚StringAttribute..' aufgeführten allgemeinen Bedingungen durch.

Spezielle Semantik und Rückgabewert

Als spezielle Vergleichsmethode wird in diesem Fall der *case-insensitive LCS*-Vergleich verwendet. Dies bedeutet, dass innerhalb beider Zeichenketten die längste, gemeinsamste Teilsequenz gesucht wird und letztlich der Quotient der Länge dieser und der Länge des längeren Attributwerts als Ähnlichkeit zurückgegeben wird. Bei dieser Variante ist die Groß- und Kleinschreibung allerdings *nicht* von Belang.

4.2.8 StringAttributeUsingLcsCS

Diese Vergleichsfunktion führt Vergleiche anhand den unter Abschnitt 4.2 ‚StringAttribute..' aufgeführten allgemeinen Bedingungen durch.

Spezielle Semantik und Rückgabewert

Als spezielle Vergleichsmethode wird in diesem Fall der *case-insensitive LCS*-Vergleich verwendet. Dies bedeutet, dass innerhalb beider Zeichenketten die längste, gemeinsamste Teilsequenz gesucht wird und letztlich der Quotient der Länge dieser und der Länge des längeren Attributwerts als Ähnlichkeit zurückgegeben wird. Die Groß- und Kleinschreibung ist bei dieser Variante von Belang!

4.3 `NumericAttribute`...

Dieser Abschnitt behandelt Vergleichsfunktionen des Typs `NumericAttribute`, mit Hilfe derer sich numerische Attributwerte der aktuellen Eingabeknoten vergleichen lassen. Allgemein lassen sich bereits folgende Angaben zu dieser Gruppe von Vergleichsfunktionen machen.

Allgemeine Vorbedingungen

Für den Fall, dass explizit ein einzelnes Attribut zum Vergleich angegeben wird, muss dieses existieren. Ist dies nicht der Fall erfolgt die Auslösung einer Ausnahme des Typs `AttributeNotExistsException` oder `AnnotationNotExistsException` im Fall eines zu vergleichenden virtuellen Attributs. Eine `AttributeNotExistsException` wird auch ausgelöst, wenn der im Parameter spezifizierte reguläre Ausdruck keine zu vergleichenden Attributnamen liefert. In *jedem* Fall muss das auszuwertende Attribut einen numerischen Wert beinhalten. Ist diese Voraussetzung nicht gegeben, löst die Vergleichsfunktion eine Ausnahme vom Typ `InvalidAttributeValue` aus und der SiDiff-Kern terminiert – wie in den anderen o.g. Szenarien – unverzüglich die Ausführung des Vergleichauftrags.

Allgemeine Semantik

Diese Gruppe der Vergleichsfunktionen führt den Vergleich von Eingabeknoten aufgrund eines *numerischen* Attributs durch. Dabei wird im Parameter der Name des zu vergleichenden, numerischen Attributs angegeben. Die spezielle Vergleichsmethode ist in den implementierenden Vergleichsfunktionen, die weiter unten in diesem Abschnitt gefunden werden können, spezifiziert und genauer erläutert.

Allgemeiner Rückgabewert

- Sofern für beide Eingabeknoten das im Parameter spezifizierte Attribut existiert, wird anhand der entsprechenden Vergleichsmethode die Ähnlichkeit der beiden numerischen Attributwerte bestimmt und zurückgegeben.
- Existiert für einen oder gar beide Eingabeknoten das angegebene Attribut nicht, beträgt die Ähnlichkeit 0.
- Die Ähnlichkeit beträgt ebenfalls 0, sofern einer der beiden Attributwerte nicht numerisch ist und somit kein Vergleich durchgeführt werden kann.

Allgemeine Parameter

Die Parameter der implementierenden Vergleichsfunktionen unterscheiden sich teilweise. In jedem Fall muss jedoch der Name des zu vergleichenden Attributs übergeben werden.
Mehr zum genaueren Parameterformat im Abschnitt *Spezielle Parameter* der jeweiligen Vergleichsfunktion.

4.3.1 NumericAttributeSimilarUsingGauss

Diese Vergleichsfunktion führt Vergleiche anhand den unter Abschnitt 4.3 ‚NumericAttribute..' aufgeführten allgemeinen Bedingungen durch.

Spezielle Parameter

Diese Vergleichsfunktion benötigt zwei Parameter, die durch „;" voneinander abgetrennt werden. Folgend beide Unter-Parameter in der anzugebenden Reihenfolge:

1. Der Name des zu vergleichenden, numerischen Attributs.
2. Ein Skalierungsfaktor, der in die Ähnlichkeitsberechnung eingeht.
   Mehr dazu unter *Spezielle Semantik und Rückgabewert*.

Ein gültiger Parameter wäre also: `myattribute;0.75`

Sofern ein ungültiges Parameterformat übergeben wird oder der Skalierungsfaktor nicht parsbar ist, erfolgt die Auslösung einer Ausnahme des Typs `InvalidParameterSyntaxException`.

Spezielle Semantik und Rückgabewert

Diese Vergleichsfunktion vergleicht Ganz- bzw. 64-Bit Fließkomma-Zahlen vom Typ `Double` anhand der *Gaußschen Normalverteilung* auf Ähnlichkeit. Dazu wird im Parameter dieser Vergleichsfunktion ebenfalls ein Skalierungsfaktor angegeben, um die Form der Glockenkurve individuell beeinflussen zu können.

Im Detail:

- Zwischen beiden Attributwerten wird die Differenz gebildet und diese quadriert. Danach wird der Quotient aus der quadrierten Differenz und dem Skalierungsfaktor gebildet und das negierte Zwischenergebnis letztlich auf die Eulersche Exponentialfunktion angewandt.
- In mathematischer Form wie folgt: $\exp\left(-\frac{(value1-value2)^2}{scale}\right)$

4.3.2 NumericAttributeEqualUsingInteger

Diese Vergleichsfunktion führt Vergleiche anhand den unter Abschnitt 4.3 ‚NumericAttribute..' aufgeführten allgemeinen Bedingungen durch.

Spezielle Parameter

Diese Vergleichsfunktion benötigt als Parameter lediglich den Namen des zu vergleichenden numerischen Attributs.

Spezielle Semantik und Rückgabewert

Diese Vergleichsfunktion vergleicht **`Integer`**-Werte (also 32-Bit große Ganzzahl-Werte) auf Identität.

- Sind beide Attributwerte gleich, so beträgt die Ähnlichkeit 1.
- Bei verschiedenen Attributwerten wird 0 zurückgegeben.

4.3.3 NumericAttributeEqualUsingFloat

Diese Vergleichsfunktion führt Vergleiche anhand den unter Abschnitt 4.3 ‚NumericAttribute..' aufgeführten allgemeinen Bedingungen durch.

Spezielle Parameter

Diese Vergleichsfunktion benötigt als Parameter lediglich den Namen des zu vergleichenden numerischen Attributs.

Spezielle Semantik und Rückgabewert

Diese Vergleichsfunktion vergleicht **Float**-Werte (also 32-Bit große Fließkomma-Zahlen) auf Identität.

- Sind beide Attributwerte gleich, so beträgt die Ähnlichkeit 1.
- Bei verschiedenen Attributwerten wird 0 zurückgegeben.

4.3.4 NumericAttributeEqualUsingDouble

Diese Vergleichsfunktion führt Vergleiche anhand den unter Abschnitt 4.3 ‚NumericAttribute..' aufgeführten allgemeinen Bedingungen durch.

Spezielle Parameter

Diese Vergleichsfunktion benötigt als Parameter lediglich den Namen des zu vergleichenden numerischen Attributs.

Spezielle Semantik und Rückgabewert

Diese Vergleichsfunktion vergleicht **Double**-Werte (also 64-Bit große Fließkomma-Zahlen) auf Identität.

- Sind beide Attributwerte gleich, so beträgt die Ähnlichkeit 1.
- Bei verschiedenen Attributwerten wird 0 zurückgegeben.

4.4 `Parents`...

Dieser Abschnitt behandelt Vergleichsfunktionen des Typs `Parents`, mit denen sich die Eingabeknoten anhand ihrer Elternknoten vergleichen lassen.

Allgemeine Vorbedingungen

Mindestens einer der beiden Eingabeknoten sollte einen Elternknoten besitzen.
Ist dieses nicht der Fall – z.B. wenn die Eingabeknoten Wurzelelemente darstellen –, wird eine Ausnahme des Typs `NothingToCompareException` ausgelöst.

Allgemeine Semantik

Die Eingabeknoten werden anhand ihrer Elternknoten miteinander verglichen. Die genaue Art der Ähnlichkeitsbestimmung ist dabei in den implementierenden Vergleichsfunktionen, die weiter unten im Abschnitt eingesehen werden können, festgelegt.

Allgemeiner Rückgabewert

- Besitzt genau *einer der beiden* Eingabeknoten keinen Elternknoten, so wird 0 als Similarity zurückgegeben.
- Besitzen *beide* Eingabeknoten keinen Elternknoten, so wird eine Ausnahme des Typs `NothingToCompareException` ausgelöst.
- Für den übrigen Fall – also wenn für beide Eingabeknoten Elternknoten existieren – erfolgt die Ähnlichkeitsbestimmung anhand der Semantik der speziellen Vergleichsfunktion (s.u.).

Allgemeine Parameter

Keine der Vergleichsfunktionen dieses Typs benötigen einen Parameter.

4.4.1 `ParentsMatchedOrSimilar`

Diese Vergleichsfunktion führt Vergleiche anhand den unter Abschnitt 4.4 ‚`Parents`..' aufgeführten allgemeinen Bedingungen durch.

Spezielle Semantik und Rückgabewert

Die Eingabeknoten werden anhand der direkten Übereinstimmung bzw. der Ähnlichkeit der Elternknoten miteinander verglichen. Zunächst wird also geprüft, ob die Elternknoten bereits übereinstimmen – ist dies nicht der Fall, so erfolgt der Vergleich aufgrund der Ähnlichkeiten der Elternknoten untereinander.

Als Ähnlichkeit der Eingabeknoten wird also letztlich bei Übereinstimmung der Elternknoten 1, andernfalls die bekannte Ähnlichkeit der beiden Elternknoten zurückgegeben.

Bemerkung: Diese Vergleichsfunktion ist nicht reflexiv.

4.4.2 `ParentsMatched`

Diese Vergleichsfunktion führt Vergleiche anhand den unter Abschnitt 4.4 ‚`Parents`..' aufgeführten allgemeinen Bedingungen durch.

Spezielle Semantik und Rückgabewert

Die Eingabeknoten werden anhand ihrer Elternknoten miteinander verglichen. Dabei werden allein *Übereinstimmungen* berücksichtigt. Es erfolgt also *kein* Vergleich aufgrund von Ähnlichkeiten.

Bei Übereinstimmung der Elternknoten wird 1, andernfalls 0 als Ähnlichkeit zurückgegeben.

4.4.3 ParentsSimilar

Diese Vergleichsfunktion führt Vergleiche anhand den unter Abschnitt 4.4 ‚Parents..' aufgeführten allgemeinen Bedingungen durch.

Spezielle Semantik und Rückgabewert

Die Eingabeknoten werden anhand ihrer Elternknoten miteinander verglichen. Dabei werden allein die bekannten *Ähnlichkeitswerte* der Eltern berücksichtigt. Es erfolgt also *kein* Vergleich aufgrund von Übereinstimmungen.

Als Ähnlichkeit der beiden Eingabeknoten wird somit die Ähnlichkeit der beiden Elternknoten zurückgegeben.

Bemerkung: Diese Vergleichsfunktion ist nicht reflexiv.

4.4.4 ParentsEqualHash

Diese Vergleichsfunktion führt Vergleiche anhand den unter Abschnitt 4.4 ‚Parents..' aufgeführten allgemeinen Bedingungen durch.

Spezielle Vorbedingungen

Den (Eltern-)Knoten müssen bereits Hashwerte in Form von Annotationen zugeordnet sein. Ist dies nicht der Fall oder Hashwerte werden in dem entsprechenden Projekt generell nicht unterstützt, wird eine Ausnahme des Typs `AnnotationNotExistsException` ausgelöst und der SiDiff-Kern terminiert den Vergleichsauftrag instantan.

Spezielle Semantik und Rückgabewert

Die Eingabeknoten werden anhand ihrer Elternknoten miteinander verglichen. Dabei wird der Vergleich auf die Identität der ermittelten *Hashwerte* der Eltern zurückgeführt.

Sind die Hashwerte der Eltern identisch, so beträgt die Ähnlichkeit der Eingabeknoten 1, ansonsten 0.

4.4.5 ParentsEqualType

Diese Vergleichsfunktion führt Vergleiche anhand den unter Abschnitt 4.4 ‚Parents..' aufgeführten allgemeinen Bedingungen durch.

Spezielle Semantik und Rückgabewert

Die Eingabeknoten werden anhand ihrer Elternknoten miteinander verglichen. Dabei wird der Vergleich auf Identität der *Typen* der Eltern zurückgeführt.

Sind die Typen der Eltern gleich, so beträgt die Ähnlichkeit der Eingabeknoten 1, ansonsten 0.

4.5 `ChildrenIO`...

Dieser Abschnitt behandelt Vergleichsfunktionen des Typs `ChildrenIO`, mit denen sich die Eingabeknoten anhand ihrer Kindknoten unter *Nichtbeachtung* der Reihenfolge vergleichen lassen.

Allgemeine Vorbedingungen

Die zu vergleichenden Eingabeknoten sollten Kindknoten besitzen, damit nicht-triviale Ähnlichkeitswerte ermittelbar sind. Mehr zu trivialen Ähnlichkeiten unter *Allgemeiner Rückgabewert*.

Allgemeine Semantik

Diese Gruppe von Vergleichsfunktionen vergleicht die Eingabeknoten anhand spezifischer Merkmale ihrer Kindknoten miteinander. Bereits anmerkbar ist, dass bei sämtlichen Vergleichen dieser Gruppe die Reihenfolge der Kindknoten nicht von Belang ist. Mehr zur genauen Semantik und auch Rückgabewert ist in den Dokumentationen der spezifischen Vergleichsfunktionen weiter unten in diesem Abschnitt einsehbar.

Allgemeiner Rückgabewert

Die Bestimmung der Rückgabewerte der spezifischen implementierenden Vergleichsfunktionen unterscheidet sich unterschiedlich stark. Daher sei an dieser Stelle auf die Dokumentationen der spezifischen Vergleichsfunktionen weiter unten in diesem Abschnitt verwiesen.

Allgemein lassen sich jedoch bereits folgende triviale Ähnlichkeitsbeziehungen und deren Rückgabewerte bestimmen:

- Existieren für lediglich einen der beiden Eingabeknoten keine Kindknoten, so beträgt die resultierende Ähnlichkeit 0.
- In dem Fall, dass *beide* Eingabeknoten keine Kindknoten besitzen, lösen Vergleichsfunktionen dieses Typs eine Ausnahme des Typs `NothingToCompareException` aus.

Allgemeine Parameter

Keine der Vergleichsfunktionen dieses Typs benötigen einen Parameter.

4.5.1 ChildrenMatchedOrSimilarIO

Diese Vergleichsfunktion führt Vergleiche anhand den unter Abschnitt 4.5 ‚ChildrenIO..' aufgeführten allgemeinen Bedingungen durch.

Spezielle Semantik und Rückgabewert

Die Kindknotenmengen werden bei dieser Vergleichsfunktion anhand von sowohl Übereinstimmungen als auch Ähnlichkeiten miteinander verglichen.

- Wird bei einem Kindknoten eine Übereinstimmung in der gegenüberliegenden Menge gefunden, geht diese anteilig mit dem Wert 1 in den Vergleich ein.
- Wenn keine Übereinstimmung für einen Kindknoten gefunden wurde, so geht der Knoten mit der höchsten Ähnlichkeit in den Vergleich ein.

Als Gesamt-Ähnlichkeit wird letztlich der Quotient aus den kumulierten Einzel-Ähnlichkeiten und der Anzahl der Kindknoten des Eingabeknotens mit den meisten Kindknoten zurückgegeben.

Bemerkungen:

- Diese Vergleichsfunktion ist nicht reflexiv!
- Die Knoten ermittelter Korrelationen werden vom weiteren Vergleich ausgeschlossen, sodass diese nicht ein weiteres mal anderen Knoten zugeordnet werden können.
- Da es sich lediglich um Vergleichs-*Heuristiken* handelt, besteht bei Anwendung von Ähnlichkeits-Korrelationen ebenfalls die Möglichkeit von suboptimalen Zuordnungen von Knotenpaaren – also dass bspw. ein Knoten A mit einem Knoten B als korrelierend betrachtet wird, obwohl er im weiteren Vergleich einem Knoten C wesentlich besser entsprechen würde.

4.5.2 ChildrenMatchedIO

Diese Vergleichsfunktion führt Vergleiche anhand den unter Abschnitt 4.5 ‚ChildrenIO..' aufgeführten allgemeinen Bedingungen durch.

Spezielle Semantik und Rückgabewert

Die Kindknotenmengen werden bei dieser Vergleichsfunktion allein anhand von *Übereinstimmungen* miteinander verglichen.

- Wird bei einem Kindknoten eine Übereinstimmung in der gegenüberliegenden Menge gefunden, geht diese anteilig mit dem Wert 1 in den Vergleich ein.
- Ist für einen Kindknoten allerdings eine Übereinstimmung *außerhalb* der anderen Kindknotenmenge existent, so geht dies anteilig mit -1 in die Gesamt-Ähnlichkeit ein. Eine „fehlerhafte" Übereinstimmung resultiert also in einer *Minderung* der Ähnlichkeit.
- Wenn keine Übereinstimmung für einen Kindknoten gefunden wurde, so geschieht nichts.

Als Gesamt-Ähnlichkeit wird letztlich der Quotient aus den kumulierten Einzel-Ähnlichkeiten und der Anzahl der Kindknoten des Eingabeknotens mit den meisten Kindknoten zurückgegeben. Sofern der Betrag der kumulierten Einzel-Ähnlichkeiten aufgrund vieler „fehlerhafter" Übereinstimmungen allerdings negativ ist, beträgt die Gesamtähnlichkeit 0.

Bemerkung: Diese Vergleichsfunktion ist nicht reflexiv!

4.5.3 ChildrenSimilarIO

Diese Vergleichsfunktion führt Vergleiche anhand den unter Abschnitt 4.5 ‚ChildrenIO..' aufgeführten allgemeinen Bedingungen durch.

Spezielle Semantik und Rückgabewert

Die Kindknotenmengen werden bei dieser Vergleichsfunktion allein anhand von bereits bekannten *Ähnlichkeiten* miteinander verglichen. Dabei wird für jeden Kindknoten der Menge A die beste Übereinstimmung in der gegenüberliegen Kindknoten-Menge B gesucht und die Ähnlichkeiten von der Knoten von Menge A zu Menge B kumuliert.

Als Gesamt-Ähnlichkeit wird letztlich der Quotient aus den kumulierten Einzel-Ähnlichkeiten und der Anzahl der Kindknoten des Eingabeknotens mit den meisten Kindknoten zurückgegeben.

Bemerkungen:

- Diese Vergleichsfunktion ist nicht reflexiv!
- Die Knoten ermittelter Korrelationen werden vom weiteren Vergleich ausgeschlossen, sodass diese nicht ein weiteres mal anderen Knoten zugeordnet werden können.
- Da es sich lediglich um Vergleichs-*Heuristiken* handelt, besteht ebenfalls die Möglichkeit von suboptimalen Zuordnungen von Knotenpaaren – also dass bspw. ein Knoten A mit einem Knoten B als korrelierend betrachtet wird, obwohl er im weiteren Vergleich einem Knoten C wesentlich besser entsprechen würde.

4.5.4 ChildrenEqualViewingHashesIO

Diese Vergleichsfunktion führt Vergleiche anhand den unter Abschnitt 4.5 ‚ChildrenIO..' aufgeführten allgemeinen Bedingungen durch.

Spezielle Vorbedingungen

Den (Kind-)Knoten müssen bereits Hashwerte in Form von Annotationen zugeordnet sein. Ist dies nicht der Fall oder Hashwerte werden in dem entsprechenden Projekt generell nicht unterstützt, wird eine Ausnahme des Typs `AnnotationNotExistsException` ausgelöst und der SiDiff-Kern terminiert den Vergleichsauftrag instantan.

Spezielle Semantik und Rückgabewert

Die Kindknotenmengen werden bei dieser Vergleichsfunktion anhand der *Hashwerte* der einzelnen Knoten auf vollständige Gleichheit überprüft. Besitzen beide Knotenmengen die gleiche Anzahl an Elementen und *jeder* Knoten hat in der gegenüberliegen Knotenmenge ein Äquivalent mit gleichem Hashwert, so beträgt die Ähnlichkeit 1, ansonsten 0.

4.5.5 ChildrenEqualViewingMatchesIO

Diese Vergleichsfunktion führt Vergleiche anhand den unter Abschnitt 4.5 ‚ChildrenIO..' aufgeführten allgemeinen Bedingungen durch.

Spezielle Semantik und Rückgabewert

Die Kindknotenmengen werden bei dieser Vergleichsfunktion anhand von bereits gefundenen *Übereinstimmungen* der einzelnen Knoten auf vollständige Übereinstimmung überprüft. Besitzen beide Knotenmengen die gleiche Anzahl an Elementen und *jeder* Knoten hat in der gegenüberliegen Knotenmenge eine Übereinstimmung, so beträgt die Ähnlichkeit 1, ansonsten 0.

4.5.6 ChildrenEqualViewingTypesIO

Diese Vergleichsfunktion führt Vergleiche anhand den unter Abschnitt 4.5 ‚ChildrenIO..' aufgeführten allgemeinen Bedingungen durch.

Spezielle Semantik und Rückgabewert

Die Kindknotenmengen werden bei dieser Vergleichsfunktion anhand der *Typen* der einzelnen Knoten auf vollständige Gleichheit überprüft. Besitzen beide Knotenmengen die gleiche Anzahl an Elementen und *jeder* Knoten hat in der gegenüberliegen Knotenmenge ein Äquivalent mit gleichem Knotentyp, so beträgt die Ähnlichkeit 1, ansonsten 0.

4.6 ChildrenCO...

Dieser Abschnitt behandelt Vergleichsfunktionen des Typs `ChildrenCO`, mit denen sich die Eingabeknoten anhand ihrer Kindknoten unter *Beachtung* der Reihenfolge vergleichen lassen.

Allgemeine Vorbedingungen

Die zu vergleichenden Eingabeknoten sollten Kindknoten besitzen, damit nicht-triviale Ähnlichkeitswerte ermittelbar sind. Mehr zu trivialen Ähnlichkeiten unter *Allgemeiner Rückgabewert*.

Allgemeine Semantik

Diese Gruppe von Vergleichsfunktionen vergleicht die Eingabeknoten anhand spezifischer Merkmale ihrer Kindknoten miteinander. Bereits anmerkbar ist, dass sämtliche Vergleiche die Reihenfolge der Kindknoten berücksichtigen. Mehr zur genauen Semantik und auch Rückgabewert ist in den Dokumentationen der spezifischen Vergleichsfunktionen weiter unten in diesem Abschnitt einsehbar.

Allgemeiner Rückgabewert

Die Bestimmung der Rückgabewerte der spezifischen implementierenden Vergleichsfunktionen unterscheidet sich unterschiedlich stark. Daher sei an dieser Stelle auf die Dokumentationen der spezifischen Vergleichsfunktionen weiter unten in diesem Abschnitt verwiesen.

Allgemein lassen sich jedoch bereits folgende triviale Ähnlichkeitsbeziehungen und deren Rückgabewerte bestimmen:

- Existiert für lediglich einen der beiden Eingabeknoten keine Kindknoten, so beträgt die resultierende Ähnlichkeit 0.
- In dem Fall, dass *beide* Eingabeknoten keine Kindknoten besitzen, lösen Vergleichsfunktionen dieses Typs eine Ausnahme des Typs `NothingToCompareException` aus.

Allgemeine Parameter

Zu den Parametern der einzelnen Vergleichsfunktionen kann an dieser Stelle keine genaue Aussage gemacht werden. Die Mehrheit der Vergleichsfunktionen dieses Typs benötigt keinen Parameter, während Vergleichsfunktion, die sich u.a. auf Ähnlichkeiten stützen, einen sog. Schwellwert zu Erkennung hinreichend genauer Korrelationen benötigen.

Beachten Sie bitte die jeweiligen spezifischen Angaben im Abschnitt *Spezifische Parameter* einer jeden Vergleichsfunktion.

4.6.1 ChildrenMatchedOrSimilarCO

Diese Vergleichsfunktion führt Vergleiche anhand den unter Abschnitt 4.6 ‚ChildrenCO..' aufgeführten allgemeinen Bedingungen durch.

Spezielle Semantik und Rückgabewert

Die Kindknotenmengen werden bei dieser Vergleichsfunktion anhand von sowohl Übereinstimmungen als auch Ähnlichkeiten miteinander verglichen. Die Reihenfolge der Knoten ist dabei von Belang, sodass der LCS-Algorithmus (Longest Common Subsequence) Anwendung findet. Dazu wird innerhalb der Knotenmengen die längste gemeinsame Teilsequenz gesucht. Genaueres in der folgenden Auflistung.

- Eine gültige Korrelation zwischen zwei Kindknoten wird erkannt, wenn die beiden Knoten bereits als Übereinstimmung gekennzeichnet sind.
- Liegt keine Übereinstimmung vor, werden die Ähnlichkeiten vom Knoten aus Menge A mit dem gegenüberliegenden Knoten in Menge B betrachtet. Liegt die Ähnlichkeit über dem unter *Spezielle Parameter* beschriebenen Schwellwert oder entspricht genau diesem, wird auf diese Weise eine Korrelation festgestellt.
- Trifft keine der beiden Bedingungen zu, endet eine längste Teilsequenz an diesem Knotenpaar.

Als Gesamt-Ähnlichkeit wird letztlich der Quotient aus der Länge der längsten gefundenen Teilsequenz und der Anzahl der Kindknoten des Eingabeknotens mit den meisten Kindknoten gebildet und zurückgegeben.

Bemerkung: Diese Vergleichsfunktion ist nicht reflexiv!

Spezielle Parameter

Diese Vergleichsfunktion benötigt als Parameter einen Schwellwert im Bereich von 0 bis 1, der angibt, ab welcher Ähnlichkeit der LCS-Algorithmus gültige Korrelationen beim Ähnlichkeitsvergleich erkennt. Ein gültiger Schwellwert wäre z.B. 0.55.

4.6.2 ChildrenMatchedCO

Diese Vergleichsfunktion führt Vergleiche anhand den unter Abschnitt 4.6 ‚ChildrenCO..' aufgeführten allgemeinen Bedingungen durch.

Spezielle Semantik und Rückgabewert

Die Kindknotenmengen werden bei dieser Vergleichsfunktion allein anhand von *Übereinstimmungen* miteinander verglichen. Die Reihenfolge der Knoten ist dabei von Belang, sodass der LCS-Algorithmus (Longest Common Subsequence) Anwendung findet. Dazu wird innerhalb der Knotenmengen die längste gemeinsame Teilsequenz gesucht. Genaueres in der folgenden Auflistung.

- Eine gültige Korrelation zwischen zwei Kindknoten wird erkannt, wenn die beiden Knoten bereits als Übereinstimmung gekennzeichnet sind.
- Liegt keine Übereinstimmung vor, endet eine längste Teilsequenz an diesem Knotenpaar.

Als Gesamt-Ähnlichkeit wird letztlich der Quotient aus der Länge der längsten gefundenen Teilsequenz und der Anzahl der Kindknoten des Eingabeknotens mit den meisten Kindknoten gebildet und zurückgegeben.

Spezielle Parameter

Diese Vergleichsfunktion benötigt keine Parameter.

4.6.3 ChildrenSimilarCO

Diese Vergleichsfunktion führt Vergleiche anhand den unter Abschnitt 4.6 ‚ChildrenCO..' aufgeführten allgemeinen Bedingungen durch.

Spezielle Semantik und Rückgabewert

Die Kindknotenmengen werden bei dieser Vergleichsfunktion allein anhand von *Ähnlichkeiten* miteinander verglichen. Die Reihenfolge der Knoten ist dabei von Belang, sodass der LCS-Algorithmus (Longest Common Subsequence) Anwendung findet. Dazu wird innerhalb der Knotenmengen die längste gemeinsame Teilsequenz gesucht. Genaueres in der folgenden Auflistung.

- Eine gültige Korrelation zwischen zwei Kindknoten wird erkannt, wenn die Ähnlichkeit des Knotens aus Menge A mit dem gegenüberliegenden Knoten in Menge B über dem unter *Spezielle Parameter* beschriebenen Schwellwert liegt oder genau diesem entspricht.
- Liegt keine solche Korrelation vor, endet eine mögliche längste Teilsequenz an diesem Knotenpaar.

Als Gesamt-Ähnlichkeit wird letztlich der Quotient aus der Länge der längsten gefundenen Teilsequenz und der Anzahl der Kindknoten des Eingabeknotens mit den meisten Kindknoten gebildet und zurückgegeben.

Bemerkung: Diese Vergleichsfunktion ist nicht reflexiv!

Spezielle Parameter

Diese Vergleichsfunktion benötigt als Parameter einen Schwellwert im Bereich von 0 bis 1, der angibt, ab welcher Ähnlichkeit der LCS-Algorithmus gültige Korrelationen beim Ähnlichkeitsvergleich erkennt. Ein gültiger Schwellwert wäre z.B. 0.55.

4.6.4 ChildrenEqualViewingHashesCO

Diese Vergleichsfunktion führt Vergleiche anhand den unter Abschnitt 4.6 ‚ChildrenCO..' aufgeführten allgemeinen Bedingungen durch.

Spezielle Vorbedingungen

Den (Kind-)Knoten müssen bereits Hashwerte in Form von Annotationen zugeordnet sein. Ist dies nicht der Fall oder Hashwerte werden in dem entsprechenden Projekt generell nicht unterstützt, wird eine Ausnahme des Typs `AnnotationNotExistsException` ausgelöst und der SiDiff-Kern terminiert den Vergleichsauftrag instantan.

Spezielle Semantik und Rückgabewert

Die Kindknotenmengen werden bei dieser Vergleichsfunktion anhand der *Hashwerte* der einzelnen Knoten auf vollständige Gleichheit überprüft. Die Reihenfolge der Kindknoten ist dabei von Belang. Besitzen beide Knotenmengen die gleiche Anzahl an Elementen und *jeder* Kindknoten der Menge A hat in der gegenüberliegenden Menge B an gleicher Position ein Äquivalent mit gleichem Hashwert, so beträgt die Ähnlichkeit 1, ansonsten 0.

4.6.5 ChildrenEqualViewingMatchesCO

Diese Vergleichsfunktion führt Vergleiche anhand den unter Abschnitt 4.6 ‚ChildrenCO..' aufgeführten allgemeinen Bedingungen durch.

Spezielle Semantik und Rückgabewert

Die Kindknotenmengen werden bei dieser Vergleichsfunktion anhand von *Übereinstimmungen* der einzelnen Knoten auf vollständige Gleichheit überprüft. Die Reihenfolge der Kindknoten ist dabei von Belang. Besitzen beide Knotenmengen die gleiche Anzahl an Elementen und *je-*

der Kindknoten der Menge A hat in der gegenüberliegenden Menge B an gleicher Position ein Äquivalent mit vorliegender Übereinstimmung, so beträgt die Ähnlichkeit 1, ansonsten 0.

4.6.6 ChildrenEqualViewingTypesCO

Diese Vergleichsfunktion führt Vergleiche anhand den unter Abschnitt 4.6 ‚ChildrenCO..' aufgeführten allgemeinen Bedingungen durch.

Spezielle Semantik und Rückgabewert

Die Kindknotenmengen werden bei dieser Vergleichsfunktion anhand der *Typen* der einzelnen Knoten auf vollständige Gleichheit überprüft. Die Reihenfolge der Kindknoten ist dabei von Belang. Besitzen beide Knotenmengen die gleiche Anzahl an Elementen und *jeder* Kindknoten der Menge A hat in der gegenüberliegenden Menge B an gleicher Position ein Äquivalent gleichen Knotentyps, so beträgt die Ähnlichkeit 1, ansonsten 0.

4.7 RemoteNodesIO...

Dieser Abschnitt behandelt Vergleichsfunktionen des Typs `RemoteNodesIO`, mit denen sich die Eingabeknoten anhand von Knotenmengen, die sich durch Abarbeitung von Typpfaden ergeben, unter *Nichtbeachtung* der Reihenfolge vergleichen lassen.

Allgemeine Vorbedingungen

Die im Pfadausdruck spezifizierten Kantentypen müssen existieren, da ansonsten eine Ausnahme des Typs `UnknownTypeException` ausgelöst wird. Ferner sollten die sich ergebenden Knotenmengen Elemente beeinhalten, damit nicht-triviale Ähnlichkeitswerte ermittelbar sind. Mehr zu trivialen Ähnlichkeiten unter *Allgemeiner Rückgabewert*.

Allgemeine Semantik

Die Eingabeknoten werden über eine durch den Parameter indirekt bestimmte Knotenmenge verglichen. Dabei wird im Parameter über einen Pfadausdruck ein Kantenzug festgelegt, aus dessen Abarbeitung sich am Zielpunkt letztlich die zu vergleichenden Knotenmengen ergeben. Der Pfadausdruck erlaubt auch die Angabe regulärer Ausdrücke; siehe dazu auch *Allgemeine Parameter*. Mehr zur genauen Semantik und auch Rückgabewert ist in den Dokumentationen der spezifischen Vergleichsfunktionen weiter unten in diesem Abschnitt einsehbar. Allgemein ist jedoch an dieser Stelle noch festzuhalten, dass Vergleichsfunktionen dieser Klasse im Gegensatz zu 4.8 ‚`RemoteNodesCO`..' die Reihenfolge der gefundenen Knoten *nicht* beachten.

Allgemeiner Rückgabewert

Die Bestimmung der Rückgabewerte der spezifischen implementierenden Vergleichsfunktionen unterscheidet sich unterschiedlich stark. Daher sei an dieser Stelle auf die Dokumentationen der spezifischen Vergleichsfunktionen weiter unten in diesem Abschnitt verwiesen.

Allgemein lassen sich jedoch bereits folgende triviale Ähnlichkeitsbeziehungen und deren Rückgabewerte bestimmen:

- Existieren für lediglich eine der sich ergebenden Knotemengen keine Elemente, so beträgt die resultierende Ähnlichkeit 0.
- In dem Fall, dass *beide* resultierenden Knotenmengen keine Elemente beinhalten, lösen Vergleichsfunktionen dieses Typs eine Ausnahme des Typs `NothingToCompareException` aus.

Allgemeine Parameter

Als Parameter wird ein XPath-ähnlicher Pfadausdruck übergeben, der vom Eingabeknoten aus beginnend die Knotenmenge spezifiziert, die verglichen werden soll. Der Pfadausdruck setzt sich dabei aus mehreren Navigationsschritten zusammen, die über `/` konkateniert werden. Jeder einzelne Navigationsschritt wiederum besteht aus drei Teilangaben.

1. Zunächst wird mittels eines einer speziellen Referenzangabe angegeben, welche Menge an Knoten über den Navigationsschritt im Graph bzw. Baum angesteuert werden soll – bspw. steht `C` dafür, dass nur Kindknoten betrachtet werden sollen.

   Eine Tabelle mit einer Auflistung und Erläuterung der möglichen Referenzangaben kann folgend eingesehen werden. In der letzten Spalte ist zusätzlich vermerkt, ob die mit dieser Dokumentation einhergehende Version des SiDiff-Kerns bereits die beschriebene Funktionalität unterstützt – das Feature also implementiert ist. Da bereits teilweise Bezug auf die in einem Navigationsschritt zu definierende Navigationsachse Bezug genommen wird, lesen Sie bitte auch den unter Punkt 2. zu findenden (s.u.) zugehörigen Abschnitt.

#	Bedeutung der Navigationsangabe	Impl.
O	*(Outgoing)* Alle von den Kontextknoten *ausgehenden* Kanten bzw. deren Zielknoten betrachten, die über den im Navigationsschritt angegebenen Kantentyp erreicht werden.	Ja
I	*(Incoming)* Alle zu den Kontextknoten *eingehenden* Kanten bzw. deren Quellknoten betrachten, die über den im Navigationsschritt angegebenen Kantentyp erreicht werden.	Ja
U	*(Undirected)* Alle vom aktuellen Knoten *ausgehenden* und zum aktuellen Knoten *eingehenden* Kanten verfolgen, die über den im Navigationsschritt angegebenen Kantentyp erreicht werden.	Ja
P	*(Parent)* Im aktuellen Navigationsschritt nur zu jenen *Elternknoten* der vorliegenden Kontextknotenmenge übergehen, die über den im Navigationsschritt angegebenen Kantentyp mit einem der Kontextknoten verbunden sind.	Nein
C	*(Children)* Im aktuellen Navigationsschritt zu allen *Kindknoten* der vorliegenden Kontextknotenmenge übergehen, die über den angegebenen Kantentyp erreicht werden können.	Nein
S	*(Siblings)* Alle *Geschwisterknoten* selektieren, wobei nur diejenigen Kontextknoten berücksichtigt werden, deren Kantentyp zum Elternknoten dem in der Navigationsachse Angegebenen entspricht. Geschwisterknoten sind derart definiert, dass der Kantentyp zum entsprechenden Elternknoten mit dem Kantentyp des Kontextknoten zum Elternelement identisch sein muss. Das für die Navigationsachse erlaubte Wildcard-Zeichen * bedeutet bei dieser Option, dass im aktuellen Navigationsschritt für *alle* vorliegenden Kontextknoten die Geschwisterknoten selektiert werden.	Nein
D	*(Depth)* Alle Knoten, die sich in der *Baum*struktur auf gleicher Hierarchieebene wie die Kontextknoten befinden, selektieren.	Nein
A	*(All)* Alle Knoten im gesamten Dokument selektieren.	Nein

2. Im Anschluss daran wird – separiert durch `:` – die *Navigationsachse* definiert. Die Navigationsachse wird durch einen Kantentyp beschrieben, über den die aus dem vorherigen Navigationsschritt vorliegenden Kontextknoten mit der unter 1. (s.o.) definierten Zielmenge verbunden sein müssen, um in die neue Kontextknotenmenge aufgenommen zu werden. An dieser Stelle wird also eine Selektion nach Kantentyp getroffen.

   Ferner ist es möglich anstatt eines fixen Kantentyps einen regulären Ausdruck zu definieren und somit eine Menge von Kantentypen zu spezifizieren. Dabei können zum einen *positive* reguläre Ausdrücke definiert werden, die explizit eine zu betrachtenden Menge an Kantentypen festlegen, als auch *negative* reguläre Ausdrücke, die Ausschlusslisten definieren – also festlegen, welche Kantentypen *nicht* betrachtet und verfolgt werden sollen.

   Damit für die Auswertungslogik erkennbar ist, ob der Benutzer einen solchen (positiven bzw. negativen) regulären Ausdruck definiert oder einen fixen Kantentyp angegeben hat, wird im Falle eines regulären Ausdrucks vor selbigen ein weiteres Steuerzeichen gesetzt. Der Vorrat an solch gültigen Präfixen und deren Semantik ist dabei folgender Tabelle zu entnehmen.

Bitte umblättern.

Präfix	Bedeutung des Steuerzeichens
(kein)	Angabe eines fixen Kantentyps.
+	Angabe eines regulären Ausdrucks: Alle auf den Ausdruck zutreffenden Kantentypen werden berücksichtigt (*positiver* regulärer Ausdruck).
-	Angabe eines regulären Ausdrucks: Alle Kantentypen, die *nicht* dem regulären Ausdruck entsprechen, werden verfolgt (*negativer* regulärer Ausdruck).

Bemerkung: Beachten Sie bitte, dass die Verwendung regulärer Ausdrücke den Vergleich bei größeren zu Grunde liegenden Eingabe-Modellen *erheblich* verlangsamt!

3. An letzter Position wird dem Benutzer eine weitere Möglichkeit eingeräumt *optional* Selektionsbedingungen in Form eines aus XPath bekannten *Prädikats* zu definieren. Das Prädikat muss dabei von eckigen Klammern umschlossen werden und unmittelbar mit der vorherigen Kantentypdefinition kontateniert sein, sofern eines definiert und angegeben werden soll. Der Prädikatsausdruck sähe syntaktisch also folgendermaßen aus: `[`*Prädikat*`]`

   Zum derzeitigen Zeitpunkt werden vom SiDiff-Kern allerdings noch keine auswertbaren Prädikate angeboten. Daher sei diese Definition an dieser Stelle nur aufgezeigt, um kenntlich zu machen, welche Mächtigkeit für diese Klasse an Vergleichsfunktionen unterstellt werden kann. Der mögliche Funktionsumfang an Prädikaten ist dabei annähernd unbegrenzt. Beispielsweise könnte in späteren SiDiff-Versionen dem Benutzer die Option eingeräumt werden mittels Prädikaten zusätzlich auch Selektionen nach *Knotentypen* vorzunehmen.

   Sofern in nachfolgenden Versionen des SiDiff-Kerns Prädikatsausdrücke implementiert sind, werden solche in der entsprechend zugehörigen Dokumentation an dieser Stelle dokumentiert sein.

Folgend einige Beispiele für gültige Pfadausdrücke. Unterstellt seien dabei die existenten Kantentypen `typA` und `typB`. Aufgrund der Tatsache, dass Prädikatsausdrücke bisher nicht im SiDiff-Kern implementiert sind, werden solche Ausdrücke nicht in den folgenden Beispielen verwendet.

Beispiel	Bemerkung
`I:typA/O:typB/L:typA`	Typpfad ohne reguläre Ausdrücke
`A:typB/O:+[typA|typB]/L:typA`	*Positiver* regulärer Ausdruck an 2. Position
`D:-[typA]/I:typB`	*Negativer* regulärer Ausdruck an 1. Position
`C:-[typA]/I:typB/L:+[typA|typB]`	Anwendung von zwei regulären Ausdrücken

Bemerkung: Explizit definierte Kantentypen müssen existieren, da ansonsten eine Ausnahme ausgelöst wird.

Siehe auch

Beachten Sie auch die Vergleichsfunktionen des Typs 4.9 ‚`OutgoingReferencesIO`..‘ und 4.12 ‚`IncomingReferencesIO`..‘, die sich bei aus- oder eingehenden Kanten *eines einzelnen* Typs performanter erweisen, als Funktionen der Klasse 4.7 ‚`RemoteNodesIO`..‘, die auf zu parsende *Pfad*ausdrücke zurückgreifen müssen.

4.7.1 RemotesNodesMatchedOrSimilarIO

Diese Vergleichsfunktion führt Vergleiche anhand den unter Abschnitt 4.7 ‚RemoteNodesIO..' aufgeführten allgemeinen Bedingungen durch.

Spezielle Semantik und Rückgabewert

Die aus dem Pfadausdruck resultierenden Knotenmengen werden bei dieser Vergleichsfunktion anhand von sowohl Übereinstimmungen als auch Ähnlichkeiten miteinander verglichen.

- Wird bei einem Knoten eine Übereinstimmung in der gegenüberliegenden Menge gefunden, geht diese anteilig mit dem Wert 1 in den Vergleich ein.
- Wenn keine Übereinstimmung für einen Knoten gefunden wurde, so geht der Knoten mit der höchsten Ähnlichkeit in den Vergleich ein.

Als Gesamt-Ähnlichkeit wird letztlich der Quotient aus den kumulierten Einzel-Ähnlichkeiten und der Größe der Knotenmenge mit den meisten Elementen zurückgegeben.

Bemerkungen:

- Diese Vergleichsfunktion ist nicht reflexiv!
- Die Knoten ermittelter Korrelationen werden vom weiteren Vergleich ausgeschlossen, sodass diese nicht ein weiteres mal anderen Knoten zugeordnet werden können.
- Da es sich lediglich um Vergleichs-*Heuristiken* handelt, besteht bei Anwendung von Ähnlichkeits-Korrelationen ebenfalls die Möglichkeit von suboptimalen Zuordnungen von Knotenpaaren – also dass bspw. ein Knoten A mit einem Knoten B als korrelierend betrachtet wird, obwohl er im weiteren Vergleich einem Knoten C wesentlich besser entsprechen würde.

4.7.2 RemotesNodesMatchedIO

Diese Vergleichsfunktion führt Vergleiche anhand den unter Abschnitt 4.7 ‚RemoteNodesIO..' aufgeführten allgemeinen Bedingungen durch.

Spezielle Semantik und Rückgabewert

Die aus dem Pfadausdruck resultierenden Knotenmengen werden bei dieser Vergleichsfunktion allein anhand von *Übereinstimmungen* miteinander verglichen.

- Wird bei einem Knoten eine Übereinstimmung in der gegenüberliegenden Menge gefunden, geht diese anteilig mit dem Wert 1 in den Vergleich ein.
- Ist für einen Knoten allerdings eine Übereinstimmung *außerhalb* der anderen Knotenmenge existent, so geht diese anteilig mit -1 in die Gesamt-Ähnlichkeit ein. Eine „fehlerhafte" Übereinstimmung resultiert also in einer *Minderung* der Ähnlichkeit.
- Wenn keine Übereinstimmung für einen Knoten gefunden wurde, so geschieht nichts.

Als Gesamt-Ähnlichkeit wird letztlich der Quotient aus den kumulierten Einzel-Ähnlichkeiten und der Größe der resultierenden Knotenmenge mit den meisten Elementen zurückgegeben. Sofern der Betrag der kumulierten Einzel-Ähnlichkeiten aufgrund vieler „fehlerhafter" Übereinstimmungen allerdings negativ ist, beträgt die Gesamtähnlichkeit 0.

Bemerkung: Diese Vergleichsfunktion ist nicht reflexiv!

4.7.3 RemotesNodesSimilarIO

Diese Vergleichsfunktion führt Vergleiche anhand den unter Abschnitt 4.7 ‚RemoteNodesIO..' aufgeführten allgemeinen Bedingungen durch.

Spezielle Semantik und Rückgabewert

Die resultierenden Knotenmengen werden bei dieser Vergleichsfunktion allein anhand von bereits bekannten *Ähnlichkeiten* miteinander verglichen. Dabei wird für jeden Knoten der Menge A die beste Korrelation in der gegenüberliegen Knotenmenge B gesucht und die Ähnlichkeiten von der Knoten von Menge A zu Menge B kumuliert.

Als Gesamt-Ähnlichkeit wird letztlich der Quotient aus den kumulierten Einzel-Ähnlichkeiten und der Größe der resultierenden Knotenmenge mit den meisten Elementen zurückgegeben.

Bemerkungen:

- Diese Vergleichsfunktion ist nicht reflexiv!
- Die Knoten ermittelter Korrelationen werden vom weiteren Vergleich ausgeschlossen, sodass diese nicht ein weiteres mal anderen Knoten zugeordnet werden können.
- Da es sich lediglich um Vergleichs-*Heuristiken* handelt, besteht ebenfalls die Möglichkeit von suboptimalen Zuordnungen von Knotenpaaren – also dass bspw. ein Knoten A mit einem Knoten B als korrelierend betrachtet wird, obwohl er im weiteren Vergleich einem Knoten C wesentlich besser entsprechen würde.

4.7.4 RemotesNodesEqualViewingHashesIO

Diese Vergleichsfunktion führt Vergleiche anhand den unter Abschnitt 4.7 ‚RemoteNodesIO..' aufgeführten allgemeinen Bedingungen durch.

Spezielle Vorbedingungen

Den Knoten müssen bereits Hashwerte in Form von Annotationen zugeordnet sein. Ist dies nicht der Fall oder Hashwerte werden in dem entsprechenden Projekt generell nicht unterstützt, wird eine Ausnahme des Typs `AnnotationNotExistsException` ausgelöst und der SiDiff-Kern terminiert den Vergleichsauftrag instantan.

Spezielle Semantik und Rückgabewert

Die resultierenden Knotenmengen werden bei dieser Vergleichsfunktion anhand der *Hashwerte* der einzelnen Knoten auf vollständige Gleichheit überprüft. Besitzen beide Knotenmengen die gleiche Anzahl an Elementen und *jeder* Knoten hat in der gegenüberliegen Knotenmenge ein Äquivalent mit gleichem Hashwert, so beträgt die Ähnlichkeit 1, ansonsten 0.

4.7.5 RemotesNodesEqualViewingMatchesIO

Diese Vergleichsfunktion führt Vergleiche anhand den unter Abschnitt 4.7 ‚RemoteNodesIO..' aufgeführten allgemeinen Bedingungen durch.

Spezielle Semantik und Rückgabewert

Die resultierenden Knotenmengen werden bei dieser Vergleichsfunktion anhand von bereits gefundenen *Übereinstimmungen* der einzelnen Knoten auf vollständige Übereinstimmung überprüft. Besitzen beide Knotenmengen die gleiche Anzahl an Elementen und *jeder* Knoten hat in der gegenüberliegen Knotenmenge eine Übereinstimmung, so beträgt die Ähnlichkeit 1, ansonsten 0.

4.7.6 RemotesNodesEqualViewingTypesIO

Diese Vergleichsfunktion führt Vergleiche anhand den unter Abschnitt 4.7 ‚RemoteNodesIO..' aufgeführten allgemeinen Bedingungen durch.

Spezielle Semantik und Rückgabewert

Die resultierenden Knotenmengen werden bei dieser Vergleichsfunktion anhand der *Typen* der einzelnen Knoten auf vollständige Gleichheit überprüft. Besitzen beide Knotenmengen die gleiche Anzahl an Elementen und *jeder* Knoten hat in der gegenüberliegen Knotenmenge ein Äquivalent mit gleichem Knotentyp, so beträgt die Ähnlichkeit 1, ansonsten 0.

4.8 RemoteNodesCO...

Dieser Abschnitt behandelt Vergleichsfunktionen des Typs `RemoteNodesCO`, mit denen sich die Eingabeknoten anhand von Knotenmengen, die sich durch Abarbeitung von Typpfaden ergeben, unter *Nichtbeachtung* der Reihenfolge vergleichen lassen.

Allgemeine Vorbedingungen

Die im Pfadausdruck spezifizierten Kantentypen müssen existieren, da ansonsten eine Ausnahme des Typs `UnknownTypeException` ausgelöst wird. Ferner sollten die sich ergebenden Knotenmengen Elemente beeinhalten, damit nicht-triviale Ähnlichkeitswerte ermittelbar sind. Mehr zu trivialen Ähnlichkeiten unter *Allgemeiner Rückgabewert*.

Allgemeine Semantik

Die Eingabeknoten werden über eine durch den Parameter indirekt bestimmte Knotenmenge verglichen. Dabei wird im Parameter über einen Pfadausdruck ein Kantenzug festgelegt, aus dessen Abarbeitung sich am Zielpunkt letztlich die zu vergleichenden Knotenmengen ergeben (siehe dazu auch *Allgemeine Parameter*). Mehr zur genauen Semantik und auch Rückgabewert ist in den Dokumentationen der spezifischen Vergleichsfunktionen weiter unten in diesem Abschnitt einsehbar. Allgemein ist jedoch an dieser Stelle noch festzuhalten, dass Vergleichsfunktionen dieser Klasse im Gegensatz zu 4.7 ‚`RemoteNodesIO`..' die Reihenfolge der gefundenen Knoten beachten.

Allgemeiner Rückgabewert

Die Bestimmung der Rückgabewerte der spezifischen implementierenden Vergleichsfunktionen unterscheidet sich unterschiedlich stark. Daher sei an dieser Stelle auf die Dokumentationen der spezifischen Vergleichsfunktionen weiter unten in diesem Abschnitt verwiesen.

Allgemein lassen sich jedoch bereits folgende triviale Ähnlichkeitsbeziehungen und deren Rückgabewerte bestimmen:

- Existieren für lediglich eine der sich ergebenden Knotemengen keine Elemente, so beträgt die resultierende Ähnlichkeit 0.
- In dem Fall, dass *beide* resultierenden Knotenmengen keine Elemente beinhalten, lösen Vergleichsfunktionen dieses Typs eine Ausnahme des Typs `NothingToCompareException` aus.

Allgemeine Parameter

Als Parameter wird ein XPath-ähnlicher Pfadausdruck übergeben, der vom Eingabeknoten aus beginnend die Knotenmenge spezifiziert, die verglichen werden soll. Der Pfadausdruck setzt sich dabei aus mehreren Navigationsschritten zusammen, die über `/` konkateniert werden. Jeder einzelne Navigationsschritt wiederum besteht aus drei Teilangaben.

1. Zunächst wird mittels eines einer speziellen Referenzangabe angegeben, welche Menge an Knoten über den Navigationsschritt im Graph bzw. Baum angesteuert werden soll – bspw. steht `C` dafür, dass nur Kindknoten betrachtet werden sollen.

   Eine Tabelle mit einer Auflistung und Erläuterung der möglichen Referenzangaben kann folgend eingesehen werden. In der letzten Spalte ist zusätzlich vermerkt, ob die mit dieser Dokumentation einhergehende Version des SiDiff-Kerns bereits die beschriebene Funktionalität unterstützt – das Feature also implementiert ist. Da bereits teilweise Bezug auf die in einem Navigationsschritt zu definierende Navigationsachse Bezug genommen wird, lesen Sie bitte auch den unter Punkt 2. zu findenden (s.u.) zugehörigen Abschnitt.

#	Bedeutung der Navigationsangabe	Impl.
O	*(Outgoing)* Alle von den Kontextknoten *ausgehenden* Kanten bzw. deren Zielknoten betrachten, die über den im Navigationsschritt angegebenen Kantentyp erreicht werden.	Ja
I	*(Incoming)* Alle zu den Kontextknoten *eingehenden* Kanten bzw. deren Quellknoten betrachten, die über den im Navigationsschritt angegebenen Kantentyp erreicht werden.	Ja
U	*(Undirected)* Alle vom aktuellen Knoten *ausgehenden* und zum aktuellen Knoten *eingehenden* Kanten verfolgen, die über den im Navigationsschritt angegebenen Kantentyp erreicht werden.	Ja
P	*(Parent)* Im aktuellen Navigationsschritt nur zu jenen *Elternknoten* der vorliegenden Kontextknotenmenge übergehen, die über den im Navigationsschritt angegebenen Kantentyp mit einem der Kontextknoten verbunden sind.	Nein
C	*(Children)* Im aktuellen Navigationsschritt zu allen *Kindknoten* der vorliegenden Kontextknotenmenge übergehen, die über den angegebenen Kantentyp erreicht werden können.	Nein
S	*(Siblings)* Alle *Geschwisterknoten* selektieren, wobei nur diejenigen Kontextknoten berücksichtigt werden, deren Kantentyp zum Elternknoten dem in der Navigationsachse Angegebenen entspricht. Geschwisterknoten sind derart definiert, dass der Kantentyp zum entsprechenden Elternknoten mit dem Kantentyp des Kontextknoten zum Elternelement identisch sein muss. Das für die Navigationsachse erlaubte Wildcard-Zeichen * bedeutet bei dieser Option, dass im aktuellen Navigationsschritt für *alle* vorliegenden Kontextknoten die Geschwisterknoten selektiert werden.	Nein
D	*(Depth)* Alle Knoten, die sich in der *Baum*struktur auf gleicher Hierarchieebene wie die Kontextknoten befinden, selektieren.	Nein
A	*(All)* Alle Knoten im gesamten Dokument selektieren.	Nein

2. Im Anschluss daran wird – separiert durch `:` – die *Navigationsachse* definiert. Die Navigationsachse wird durch einen Kantentyp beschrieben, über den die aus dem vorherigen Navigationsschritt vorliegenden Kontextknoten mit der unter 1. (s.o.) definierten Zielmenge verbunden sein müssen, um in die neue Kontextknotenmenge aufgenommen zu werden. An dieser Stelle wird also eine Selektion nach Kantentyp getroffen.

   Ferner ist es möglich anstatt eines fixen Kantentyps einen regulären Ausdruck zu definieren und somit eine Menge von Kantentypen zu spezifizieren. Dabei können zum einen *positive* reguläre Ausdrücke definiert werden, die explizit eine zu betrachtenden Menge an Kantentypen festlegen, als auch *negative* reguläre Ausdrücke, die Ausschlusslisten definieren – also festlegen, welche Kantentypen *nicht* betrachtet und verfolgt werden sollen.

   Damit für die Auswertungslogik erkennbar ist, ob der Benutzer einen solchen (positiven bzw. negativen) regulären Ausdruck definiert oder einen fixen Kantentyp angegeben hat, wird im Falle eines regulären Ausdrucks vor selbigen ein weiteres Steuerzeichen gesetzt. Der Vorrat an solch gültigen Präfixen und deren Semantik ist dabei folgender Tabelle zu entnehmen.

Bitte umblättern.

Präfix	Bedeutung des Steuerzeichens
(kein)	Angabe eines fixen Kantentyps.
+	Angabe eines regulären Ausdrucks: Alle auf den Ausdruck zutreffenden Kantentypen werden berücksichtigt (*positiver* regulärer Ausdruck).
-	Angabe eines regulären Ausdrucks: Alle Kantentypen, die *nicht* dem regulären Ausdruck entsprechen, werden verfolgt (*negativer* regulärer Ausdruck).

Bemerkung: Beachten Sie bitte, dass die Verwendung regulärer Ausdrücke den Vergleich bei größeren zu Grunde liegenden Eingabe-Modellen *erheblich* verlangsamt!

3. An letzter Position wird dem Benutzer eine weitere Möglichkeit eingeräumt *optional* Selektionsbedingungen in Form eines aus XPath bekannten *Prädikats* zu definieren. Das Prädikat muss dabei von eckigen Klammern umschlossen werden und unmittelbar mit der vorherigen Kantentypdefinition kontateniert sein, sofern eines definiert und angegeben werden soll. Der Prädikatsausdruck sähe syntaktisch also folgendermaßen aus: `[`*Prädikat*`]`

   Zum derzeitigen Zeitpunkt werden vom SiDiff-Kern allerdings noch keine auswertbaren Prädikate angeboten. Daher sei diese Definition an dieser Stelle nur aufgezeigt, um kenntlich zu machen, welche Mächtigkeit für diese Klasse an Vergleichsfunktionen unterstellt werden kann. Der mögliche Funktionsumfang an Prädikaten ist dabei annähernd unbegrenzt. Beispielsweise könnte in späteren SiDiff-Versionen dem Benutzer die Option eingeräumt werden mittels Prädikaten zusätzlich auch Selektionen nach *Knotentypen* vorzunehmen.

   Sofern in nachfolgenden Versionen des SiDiff-Kerns Prädikatsausdrücke implementiert sind, werden solche in der entsprechend zugehörigen Dokumentation an dieser Stelle dokumentiert sein.

Folgend einige Beispiele für gültige Pfadausdrücke. Unterstellt seien dabei die existenten Kantentypen `typA` und `typB`. Aufgrund der Tatsache, dass Prädikatsausdrücke bisher nicht im SiDiff-Kern implementiert sind, werden solche Ausdrücke nicht in den folgenden Beispielen verwendet.

Beispiel	Bemerkung
`I:typA/O:typB/L:typA`	Typpfad ohne reguläre Ausdrücke
`A:typB/O:+[typA|typB]/L:typA`	*Positiver* regulärer Ausdruck an 2. Position
`D:-[typA]/I:typB`	*Negativer* regulärer Ausdruck an 1. Position
`C:-[typA]/I:typB/L:+[typA|typB]`	Anwendung von zwei regulären Ausdrücken

Bemerkung: Explizit definierte Kantentypen müssen existieren, da ansonsten eine Ausnahme ausgelöst wird.

Siehe auch

Beachten Sie auch die Vergleichsfunktionen des Typs 4.10 ‚`OutgoingReferencesCO`..‘ und 4.13 ‚`IncomingReferencesCO`..‘, die sich bei aus- oder eingehenden Kanten *eines einzelnen* Typs performanter erweisen, als Funktionen der Klasse 4.8 ‚`RemoteNodesCO`..‘, die auf zu parsende *Pfad*ausdrücke zurückgreifen müssen.

4.8.1 RemotesNodesMatchedOrSimilarCO

Diese Vergleichsfunktion führt Vergleiche anhand den unter Abschnitt 4.8 ‚RemoteNodesCO..' aufgeführten allgemeinen Bedingungen durch.

Spezielle Semantik und Rückgabewert

Die resultierenden Knotenmengen werden bei dieser Vergleichsfunktion anhand von sowohl Übereinstimmungen als auch Ähnlichkeiten miteinander verglichen. Die Reihenfolge der Knoten ist dabei von Belang, sodass der LCS-Algorithmus (Longest Common Subsequence) Anwendung findet. Dazu wird innerhalb der Knotenmengen die längste gemeinsame Teilsequenz gesucht. Genaueres in der folgenden Auflistung.

- Eine gültige Korrelation zwischen zwei Knoten wird erkannt, wenn die beiden Knoten bereits als übereinstimmend gekennzeichnet sind.
- Liegt keine Übereinstimmung vor, werden die Ähnlichkeiten vom Knoten aus Menge A mit dem gegenüberliegenden Knoten in Menge B betrachtet. Liegt die Ähnlichkeit über dem unter *Spezielle Parameter* beschriebenen Schwellwert oder entspricht genau diesem, wird auf diese Weise eine Korrelation festgestellt.
- Trifft keine der beiden Bedingungen zu, endet eine längste Teilsequenz an diesem Knotenpaar.

Als Gesamt-Ähnlichkeit wird letztlich der Quotient aus der Länge der längsten gefundenen Teilsequenz und der Größe der resultierenden Knotenmenge mit den meisten Elementen gebildet und zurückgegeben.

Bemerkung: Diese Vergleichsfunktion ist nicht reflexiv!

Spezielle Parameter

Diese Vergleichsfunktion benötigt neben dem abzuarbeitenden Typpfad als Parameter auch einen Schwellwert im Bereich von 0 bis 1, der angibt, ab welcher Ähnlichkeit der LCS-Algorithmus gültige Korrelationen beim Ähnlichkeitsvergleich erkennt. Ein gültiger Schwellwert wäre z.B. 0.55. Mehr zur genauen Syntax des Parameters ist unter *Allgemeine Parameter* einsehbar.

4.8.2 RemotesNodesMatchedCO

Diese Vergleichsfunktion führt Vergleiche anhand den unter Abschnitt 4.8 ‚RemoteNodesCO..' aufgeführten allgemeinen Bedingungen durch.

Spezielle Semantik und Rückgabewert

Die resultierenden Knotenmengen werden bei dieser Vergleichsfunktion allein anhand von *Übereinstimmungen* miteinander verglichen. Die Reihenfolge der Knoten ist dabei von Belang, sodass der LCS-Algorithmus (Longest Common Subsequence) Anwendung findet. Dazu wird innerhalb der Knotenmengen die längste gemeinsame Teilsequenz gesucht. Genaueres in der folgenden Auflistung.

- Eine gültige Korrelation zwischen zwei Knoten wird erkannt, wenn die beiden Knoten bereits als übereinstimmend gekennzeichnet sind.
- Liegt keine Übereinstimmung vor, endet eine längste Teilsequenz an diesem Knotenpaar.

Als Gesamt-Ähnlichkeit wird der Quotient aus der Länge der längsten gefundenen Teilsequenz und der Größe der resultierenden Knotenmenge mit den meisten Elementen zurückgegeben.

Spezielle Parameter

Diese Vergleichsfunktion benötigt als Parameter lediglich den abzuarbeitenden Typpfad. Siehe dazu auch *Allgemeine Parameter*.

4.8.3 RemotesNodesSimilarCO

Diese Vergleichsfunktion führt Vergleiche anhand den unter Abschnitt 4.8 ‚RemoteNodesCO..' aufgeführten allgemeinen Bedingungen durch.

Spezielle Semantik und Rückgabewert

Die resultierenden Knotenmengen werden bei dieser Vergleichsfunktion allein anhand von *Ähnlichkeiten* miteinander verglichen. Die Reihenfolge der Knoten ist dabei von Belang, sodass der LCS-Algorithmus (Longest Common Subsequence) Anwendung findet. Dazu wird innerhalb der Knotenmengen die längste gemeinsame Teilsequenz gesucht. Genaueres in der folgenden Auflistung.

- Eine gültige Korrelation zwischen zwei Knoten wird erkannt, wenn die Ähnlichkeit des Knotens aus Menge A mit dem gegenüberliegenden Knoten in Menge B über dem unter *Spezielle Parameter* beschriebenen Schwellwert liegt oder genau diesem entspricht.
- Liegt keine solche Korrelation vor, endet eine mögliche längste Teilsequenz an diesem Knotenpaar.

Als Gesamt-Ähnlichkeit wird letztlich der Quotient aus der Länge der längsten gefundenen Teilsequenz und der Größe der resultierenden Knotenmenge mit den meisten Elementen gebildet und zurückgegeben.

Bemerkung: Diese Vergleichsfunktion ist nicht reflexiv!

Spezielle Parameter

Diese Vergleichsfunktion benötigt neben dem abzuarbeitenden Typpfad als Parameter auch einen Schwellwert im Bereich von 0 bis 1, der angibt, ab welcher Ähnlichkeit der LCS-Algorithmus gültige Korrelationen beim Ähnlichkeitsvergleich erkennt. Ein gültiger Schwellwert wäre z.B. 0.55. Mehr zur genauen Syntax des Parameters ist unter *Allgemeine Parameter* zu Beginn dieses Abschnitts einsehbar.

4.8.4 RemotesNodesEqualViewingHashesCO

Diese Vergleichsfunktion führt Vergleiche anhand den unter Abschnitt 4.8 ‚RemoteNodesCO..' aufgeführten allgemeinen Bedingungen durch.

Spezielle Vorbedingungen

Den Knoten müssen bereits Hashwerte in Form von Annotationen zugeordnet sein. Ist dies nicht der Fall oder Hashwerte werden in dem entsprechenden Projekt generell nicht unterstützt, wird eine Ausnahme des Typs `AnnotationNotExistsException` ausgelöst und der SiDiff-Kern terminiert den Vergleichsauftrag instantan.

Spezielle Semantik und Rückgabewert

Die resultierenden Knotenmengen werden bei dieser Vergleichsfunktion anhand der *Hashwerte* der einzelnen Knoten auf vollständige Gleichheit überprüft. Die Reihenfolge der Knoten ist dabei von Belang. Besitzen beide Knotenmengen die gleiche Anzahl an Elementen und *jeder* Knoten der Menge A hat in der gegenüberliegenden Menge B an gleicher Position ein Äquivalent mit gleichem Hashwert, so beträgt die Ähnlichkeit 1, ansonsten 0.

Spezielle Parameter

Diese Vergleichsfunktion benötigt als Parameter lediglich den abzuarbeitenden Typpfad. Siehe dazu auch *Allgemeine Parameter*.

4.8.5 RemotesNodesEqualViewingMatchesCO

Diese Vergleichsfunktion führt Vergleiche anhand den unter Abschnitt 4.8 ‚RemoteNodesCO..' aufgeführten allgemeinen Bedingungen durch.

Spezielle Semantik und Rückgabewert

Die resultierenden Knotenmengen werden bei dieser Vergleichsfunktion anhand von *Übereinstimmungen* der einzelnen Knoten auf vollständige Gleichheit überprüft. Die Reihenfolge der Knoten ist dabei von Belang. Besitzen beide Knotenmengen die gleiche Anzahl an Elementen und *jeder* Knoten der Menge A hat in der gegenüberliegenden Menge B an gleicher Position ein Äquivalent mit vorliegender Übereinstimmung, so beträgt die Ähnlichkeit 1, ansonsten 0.

Spezielle Parameter

Diese Vergleichsfunktion benötigt als Parameter lediglich den abzuarbeitenden Typpfad. Siehe dazu auch *Allgemeine Parameter*.

4.8.6 RemotesNodesEqualViewingTypesCO

Diese Vergleichsfunktion führt Vergleiche anhand den unter Abschnitt 4.8 ‚RemoteNodesCO..' aufgeführten allgemeinen Bedingungen durch.

Spezielle Semantik und Rückgabewert

Die resultierenden Knotenmengen werden bei dieser Vergleichsfunktion anhand der *Knotentypen* der einzelnen Knoten auf vollständige Gleichheit überprüft. Die Reihenfolge der Knoten ist dabei von Belang. Besitzen beide Knotenmengen die gleiche Anzahl an Elementen und *jeder* Knoten der Menge A hat in der gegenüberliegenden Menge B an gleicher Position ein Äquivalent gleichen Knotentyps, so beträgt die Ähnlichkeit 1, ansonsten 0.

Spezielle Parameter

Diese Vergleichsfunktion benötigt als Parameter lediglich den abzuarbeitenden Typpfad. Siehe dazu auch *Allgemeine Parameter*.

4.9 `OutgoingReferencesIO...`

Dieser Abschnitt behandelt Vergleichsfunktionen des Typs `OutgoingReferencesIO`, mit denen sich die Eingabeknoten anhand ausgehender Kanten eines bestimmten Typs oder alternativ anhand allen ausgehenden Kanten unter *Nichtbeachtung* der Reihenfolge vergleichen lassen.

Allgemeine Vorbedingungen

Der im Parameter spezifizierte Kantentyp muss existieren, da ansonsten eine Ausnahme des Typs `UnknownTypeException` ausgelöst wird. Ferner sollten die sich ergebenden Knotenmengen Elemente beeinhalten, damit nicht-triviale Ähnlichkeitswerte ermittelbar sind. Mehr zu trivialen Ähnlichkeiten unter *Allgemeiner Rückgabewert*.

Allgemeine Semantik

Die Eingabeknoten werden über eine durch den Parameter indirekt bestimmte Knotenmenge verglichen. Dabei wird die Knotenmenge zum Vergleich herangezogen, die sich aus Abarbeitung der vom Eingabeknoten ausgehenden Kanten des im Parameter spezifizierten Typs ergeben. Es besteht auch die Möglichkeit *alle* ausgehenden Kanten auszuwählen. Mehr zum genauen Semantik und auch Rückgabewert ist in den Dokumentationen der spezifischen Vergleichsfunktionen weiter unten in diesem Abschnitt einsehbar. Allgemein ist jedoch an dieser Stelle noch festzuhalten, dass Vergleichsfunktionen dieser Klasse im Gegensatz zu 4.10 ‚`OutgoingReferencesCO`..' die Reihenfolge der gefundenen Knoten *nicht* beachten.

Allgemeiner Rückgabewert

Die Bestimmung der Rückgabewerte der spezifischen implementierenden Vergleichsfunktionen unterscheidet sich unterschiedlich stark. Daher sei an dieser Stelle auf die Dokumentationen der spezifischen Vergleichsfunktionen weiter unten in diesem Abschnitt verwiesen.

Allgemein lassen sich jedoch bereits folgende triviale Ähnlichkeitsbeziehungen und deren Rückgabewerte bestimmen:

- Existiert für lediglich eine der sich ergebenden Knotemengen keine Elemente, so beträgt die resultierende Ähnlichkeit 0.
- In dem Fall, dass *beide* resultierenden Knotenmengen keine Elemente beinhalten, lösen Vergleichsfunktionen dieses Typs eine Ausnahme des Typs `NothingToCompareException` aus.

Allgemeine Parameter

Als Parameter wird der Name des zu betrachtenden, ausgehenden Kantentyps angegeben. Ferner ist es durch Verwendung eines `*` möglich, *alle* ausgehenden Kanten und somit letztlich alle Zielknoten zum Vergleich heranzuziehen.

Folgend zwei Beispiele für gültige Parameterwerte:

- Ausgehende Kanten des Typs `typA` betrachten: `typA`
- Alle ausgehenden Kanten bzw. Zielknoten heranziehen: `*`

Bemerkung: Sofern ein spezieller Kantentyp angegeben wird, muss dieser existent sein, da ansonsten eine Ausnahme ausgelöst wird.

4.9.1 OutgoingReferencesMatchedOrSimilarIO

Diese Vergleichsfunktion führt Vergleiche anhand den unter
Abschnitt 4.9 ‚OutgoingReferencesIO..' aufgeführten allgemeinen Bedingungen durch.

Spezielle Semantik und Rückgabewert

Die aus Abarbeitung des im Parameter angegebenen ausgehenden Kantentyps resultierenden Knotenmengen werden bei dieser Vergleichsfunktion anhand von sowohl Übereinstimmungen als auch Ähnlichkeiten miteinander verglichen.

- Wird bei einem Knoten eine Übereinstimmung in der gegenüberliegenden Menge gefunden, geht dies anteilig mit dem Wert 1 in den Vergleich ein.
- Wenn keine Übereinstimmung für einen Kindknoten gefunden wurde, so geht der Knoten mit der höchsten Ähnlichkeit in den Vergleich ein.

Als Gesamt-Ähnlichkeit wird letztlich der Quotient aus den kumulierten Einzel-Ähnlichkeiten und der Größe der Knotenmenge mit den meisten Elementen zurückgegeben.

Bemerkungen:

- Diese Vergleichsfunktion ist nicht reflexiv!
- Die Knoten ermittelter Korrelationen werden vom weiteren Vergleich ausgeschlossen, sodass diese nicht ein weiteres mal anderen Knoten zugeordnet werden können.
- Da es sich lediglich um Vergleichs-*Heuristiken* handelt, besteht bei Anwendung von Ähnlichkeits-Korrelationen ebenfalls die Möglichkeit von suboptimalen Zuordnungen von Knotenpaaren – also dass bspw. ein Knoten A mit einem Knoten B als korrelierend betrachtet wird, obwohl er im weiteren Vergleich einem Knoten C wesentlich besser entsprechen würde.

4.9.2 OutgoingReferencesMatchedIO

Diese Vergleichsfunktion führt Vergleiche anhand den unter
Abschnitt 4.9 ‚OutgoingReferencesIO..' aufgeführten allgemeinen Bedingungen durch.

Spezielle Semantik und Rückgabewert

Die aus dem Pfadausdruck resultierenden Knotenmengen werden bei dieser Vergleichsfunktion allein anhand von *Übereinstimmungen* miteinander verglichen.

- Wird bei einem Knoten eine Übereinstimmung in der gegenüberliegenden Menge gefunden, geht dies anteilig mit dem Wert 1 in den Vergleich ein.
- Ist für einen Knoten allerdings eine Übereinstimmung *außerhalb* der anderen Knotenmenge existent, so geht dies anteilig mit -1 in die Gesamt-Ähnlichkeit ein. Eine „fehlerhafte" Übereinstimmung resultiert also in einer *Minderung* der Ähnlichkeit.
- Wenn keine Übereinstimmung für einen Knoten gefunden wurde, so geschieht nichts.

Als Gesamt-Ähnlichkeit wird letztlich der Quotient aus den kumulierten Einzel-Ähnlichkeiten und der Größe der resultierenden Knotenmenge mit den meisten Elementen zurückgegeben. Sofern der Betrag der kumulierten Einzel-Ähnlichkeiten aufgrund vieler „fehlerhafter" Übereinstimmungen allerdings negativ ist, beträgt die Gesamtähnlichkeit 0.

Bemerkung: Diese Vergleichsfunktion ist nicht reflexiv!

4.9.3 OutgoingReferencesSimilarIO

Diese Vergleichsfunktion führt Vergleiche anhand den unter
Abschnitt 4.9 ‚OutgoingReferencesIO..' aufgeführten allgemeinen Bedingungen durch.

Spezielle Semantik und Rückgabewert

Die resultierenden Knotenmengen werden bei dieser Vergleichsfunktion allein anhand von bereits bekannten *Ähnlichkeiten* miteinander verglichen. Dabei wird für jeden Knoten der Menge A die beste Übereinstimmung in der gegenüberliegen Knotenmenge B gesucht und die Ähnlichkeiten von der Knoten von Menge A zu Menge B kumuliert.

Als Gesamt-Ähnlichkeit wird letztlich der Quotient aus den kumulierten Einzel-Ähnlichkeiten und der Größe der resultierenden Knotenmenge mit den meisten Elementen zurückgegeben.

Bemerkungen:

- Diese Vergleichsfunktion ist nicht reflexiv!
- Die Knoten ermittelter Korrelationen werden vom weiteren Vergleich ausgeschlossen, sodass diese nicht ein weiteres mal anderen Knoten zugeordnet werden können.
- Da es sich lediglich um Vergleichs-*Heuristiken* handelt, besteht ebenfalls die Möglichkeit von suboptimalen Zuordnungen von Knotenpaaren – also dass bspw. ein Knoten A mit einem Knoten B als korrelierend betrachtet wird, obwohl er im weiteren Vergleich einem Knoten C wesentlich besser entsprechen würde.

4.9.4 OutgoingReferencesEqualViewingHashesIO

Diese Vergleichsfunktion führt Vergleiche anhand den unter
Abschnitt 4.9 ‚OutgoingReferencesIO..' aufgeführten allgemeinen Bedingungen durch.

Spezielle Vorbedingungen

Den Knoten müssen bereits Hashwerte in Form von Annotationen zugeordnet sein. Ist dies nicht der Fall oder Hashwerte werden in dem entsprechenden Projekt generell nicht unterstützt, wird eine Ausnahme des Typs `AnnotationNotExistsException` ausgelöst und der SiDiff-Kern terminiert den Vergleichsauftrag instantan.

Spezielle Semantik und Rückgabewert

Die resultierenden Knotenmengen werden bei dieser Vergleichsfunktion anhand der *Hashwerte* der einzelnen Knoten auf vollständige Gleichheit überprüft. Besitzen beide Knotenmengen die gleiche Anzahl an Elementen und *jeder* Knoten hat in der gegenüberliegen Knotenmenge ein Äquivalent mit gleichem Hashwert, so beträgt die Ähnlichkeit 1, ansonsten 0.

4.9.5 OutgoingReferencesEqualViewingMatchesIO

Diese Vergleichsfunktion führt Vergleiche anhand den unter
Abschnitt 4.9 ‚OutgoingReferencesIO..' aufgeführten allgemeinen Bedingungen durch.

Spezielle Semantik und Rückgabewert

Die resultierenden Knotenmengen werden bei dieser Vergleichsfunktion anhand von bereits gefundenen *Übereinstimmungen* der einzelnen Knoten auf vollständige Übereinstimmung überprüft. Besitzen beide Knotenmengen die gleiche Anzahl an Elementen und *jeder* Knoten hat in der gegenüberliegen Knotenmenge eine Übereinstimmung, so beträgt die Ähnlichkeit 1, ansonsten 0.

4.9.6 OutgoingReferencesEqualViewingTypesIO

Diese Vergleichsfunktion führt Vergleiche anhand den unter
Abschnitt 4.9 ‚OutgoingReferencesIO..' aufgeführten allgemeinen Bedingungen durch.

Spezielle Semantik und Rückgabewert

Die resultierenden Knotenmengen werden bei dieser Vergleichsfunktion anhand der *Typen* der einzelnen Knoten auf vollständige Gleichheit überprüft. Besitzen beide Knotenmengen die gleiche Anzahl an Elementen und *jeder* Knoten hat in der gegenüberliegen Knotenmenge ein Äquivalent mit gleichem Knotentyp, so beträgt die Ähnlichkeit 1, ansonsten 0.

4.10 `OutgoingReferencesCO`...

Dieser Abschnitt behandelt Vergleichsfunktionen des Typs `OutgoingReferencesCO`, mit denen sich die Eingabeknoten anhand ausgehender Kanten eines bestimmtem Typs oder alternativ anhand allen ausgehenden Kanten unter *Beachtung* der Reihenfolge vergleichen lassen.

Allgemeine Vorbedingungen

Der im Parameter spezifizierte Kantentyp muss existieren, da ansonsten eine Ausnahme des Typs `UnknownTypeException` ausgelöst wird. Ferner sollten die sich ergebenden Knotenmengen Elemente beeinhalten, damit nicht-triviale Ähnlichkeitswerte ermittelbar sind. Mehr zu trivialen Ähnlichkeiten unter *Allgemeiner Rückgabewert*.

Allgemeine Semantik

Die Eingabeknoten werden über eine durch den Parameter indirekt bestimmte Knotenmenge verglichen. Dabei wird die Knotenmenge zum Vergleich herangezogen, die sich aus Abarbeitung der vom Eingabeknoten ausgehenden Kanten des im Parameter spezifizierten Typs ergeben. Es besteht auch die Möglichkeit *alle* ausgehenden Kanten auszuwählen. Mehr zum genauen Semantik und auch Rückgabewert ist in den Dokumentationen der spezifischen Vergleichsfunktionen weiter unten in diesem Abschnitt einsehbar. Allgemein ist jedoch an dieser Stelle noch festzuhalten, dass Vergleichsfunktionen dieser Klasse im Gegensatz zu 4.9 ‚`OutgoingReferencesIO`..' die Reihenfolge der gefundenen Knoten mit in den Vergleich einbeziehen.

Allgemeiner Rückgabewert

Die Bestimmung der Rückgabewerte der spezifischen implementierenden Vergleichsfunktionen unterscheidet sich unterschiedlich stark. Daher sei an dieser Stelle auf die Dokumentationen der spezifischen Vergleichsfunktionen weiter unten in diesem Abschnitt verwiesen.

Allgemein lassen sich jedoch bereits folgende triviale Ähnlichkeitsbeziehungen und deren Rückgabewerte bestimmen:

- Existiert für lediglich eine der sich ergebenden Knotemengen keine Elemente, so beträgt die resultierende Ähnlichkeit 0.
- In dem Fall, dass *beide* resultierenden Knotenmengen keine Elemente beinhalten, lösen Vergleichsfunktionen dieses Typs eine Ausnahme des Typs `NothingToCompareException` aus.

Allgemeine Parameter

Als Parameter wird der Name des zu betrachtenden, ausgehenden Kantentyps angegeben. Ferner ist es durch Verwendung eines `*` möglich, *alle* eingehenden Kanten und somit letztlich alle Quellknoten zum Vergleich heranzuziehen.

Manche implementierenden Vergleichsfunktionen dieser Kategorie benötigen jedoch ebenfalls einen Schwellwert als zweiten Parameter. Dieser schließt sich an den angegebenen Kantentyp bzw. `*` an und wird durch ein `;` von eben jenem abgetrennt. Sofern eine Vergleichsfunktion auf die Angabe dieses Schwellwerts angewiesen ist, wird dieses in der Dokumentation der einzelnen Vergleichsfunktion explizit vermerkt. Weitere Informationen zur Semantik des Schwellwerts sind ebenfalls an dieser Stelle zu finden.

Bitte umblättern.

Folgend zwei Beispiele für gültige Parameterwerte mit und ohne Schwellwert:

ohne Sw.	mit Sw.	Bedeutung
`typA`	`typA;0.3`	Ausgehende Kanten des Typs `typA` betrachten
`*`	`*;0.25`	Alle ausgehenden Kanten bzw. Quellknoten heranziehen

Bemerkung: Sofern ein spezieller Kantentyp angegeben wird, muss dieser existent sein, da ansonsten eine Ausnahme ausgelöst wird.

4.10.1 OutgoingReferencesMatchedOrSimilarCO

Diese Vergleichsfunktion führt Vergleiche anhand den unter
Abschnitt 4.10 ‚`OutgoingReferencesCO`..' aufgeführten allgemeinen Bedingungen durch.

Spezielle Semantik und Rückgabewert

Die resultierenden Knotenmengen werden bei dieser Vergleichsfunktion anhand von sowohl Übereinstimmungen als auch Ähnlichkeiten miteinander verglichen. Die Reihenfolge der Knoten ist dabei von Belang, sodass der LCS-Algorithmus (Longest Common Subsequence) Anwendung findet. Dazu wird innerhalb der Knotenmengen die längste gemeinsame Teilsequenz gesucht. Genaueres in der folgenden Auflistung.

- Eine gültige Korrelation zwischen zwei Knoten wird erkannt, wenn die beiden Knoten bereits als übereinstimmend gekennzeichnet sind.
- Liegt keine Übereinstimmung vor, werden die Ähnlichkeiten vom Knoten aus Menge A mit dem gegenüberliegenden Knoten in Menge B betrachtet. Liegt die Ähnlichkeit über dem unter *Spezielle Parameter* beschriebenen Schwellwert oder entspricht genau diesem, wird auf diese Weise eine Korrelation festgestellt.
- Trifft keine der beiden Bedingungen zu, endet eine längste Teilsequenz an diesem Knotenpaar.

Als Gesamt-Ähnlichkeit wird letztlich der Quotient aus der Länge der längsten gefundenen Teilsequenz und der Größe der resultierenden Knotenmenge mit den meisten Elementen gebildet und zurückgegeben.

Bemerkung: Diese Vergleichsfunktion ist nicht reflexiv!

Spezielle Parameter

Diese Vergleichsfunktion benötigt neben dem Name des Kantentyps als Parameter auch einen Schwellwert im Bereich von 0 bis 1, der angibt, ab welcher Ähnlichkeit der LCS-Algorithmus gültige Korrelationen beim Ähnlichkeitsvergleich erkennt. Ein gültiger Schwellwert wäre z.B. 0.55. Mehr zur genauen Syntax des Parameters ist unter *Allgemeine Parameter* zu Beginn dieses Abschnitts einsehbar.

4.10.2 OutgoingReferencesMatchedCO

Diese Vergleichsfunktion führt Vergleiche anhand den unter
Abschnitt 4.10 ‚OutgoingReferencesCO..' aufgeführten allgemeinen Bedingungen durch.

Spezielle Semantik und Rückgabewert

Die resultierenden Knotenmengen werden bei dieser Vergleichsfunktion allein anhand von *Übereinstimmungen* miteinander verglichen. Die Reihenfolge der Knoten ist dabei von Belang, sodass der LCS-Algorithmus (Longest Common Subsequence) Anwendung findet. Dazu wird innerhalb der Knotenmengen die längste gemeinsame Teilsequenz gesucht. Genaueres in der folgenden Auflistung.

- Eine gültige Korrelation zwischen zwei Knoten wird erkannt, wenn die beiden Knoten bereits als Übereinstimmung gekennzeichnet sind.
- Liegt keine Übereinstimmung vor, endet eine längste Teilsequenz an diesem Knotenpaar.

Als Gesamt-Ähnlichkeit wird letztlich der Quotient aus der Länge der längsten gefundenen Teilsequenz und der Größe der resultierenden Knotenmenge mit den meisten Elementen gebildet und zurückgegeben.

Spezielle Parameter

Diese Vergleichsfunktion benötigt als Parameter lediglich den Namen des zu verfolgenden Kantentyps oder * als Wildcard. Siehe dazu auch *Allgemeine Parameter*.

4.10.3 OutgoingReferencesSimilarCO

Diese Vergleichsfunktion führt Vergleiche anhand den unter
Abschnitt 4.10 ‚OutgoingReferencesCO..' aufgeführten allgemeinen Bedingungen durch.

Spezielle Semantik und Rückgabewert

Die resultierenden Knotenmengen werden bei dieser Vergleichsfunktion allein anhand von Ähnlichkeiten miteinander verglichen. Die Reihenfolge der Knoten ist dabei von Belang, sodass der LCS-Algorithmus (Longest Common Subsequence) Anwendung findet. Dazu wird innerhalb der Knotenmengen die längste gemeinsame Teilsequenz gesucht. Genaueres in der folgenden Auflistung.

- Eine gültige Korrelation zwischen zwei Knoten wird erkannt, wenn die Ähnlichkeit des Knotens aus Menge A mit dem gegenüberliegenden Knoten in Menge B über dem unter *Spezielle Parameter* beschriebenen Schwellwert liegt oder genau diesem entspricht.
- Liegt keine solche Korrelation vor, endet eine mögliche längste Teilsequenz an diesem Knotenpaar.

Als Gesamt-Ähnlichkeit wird letztlich der Quotient aus der Länge der längsten gefundenen Teilsequenz und der Größe der resultierenden Knotenmenge mit den meisten Elementen gebildet und zurückgegeben.

Bemerkung: Diese Vergleichsfunktion ist nicht reflexiv!

Spezielle Parameter

Diese Vergleichsfunktion benötigt neben dem Name des Kantentyps als Parameter auch einen Schwellwert im Bereich von 0 bis 1, der angibt, ab welcher Ähnlichkeit der LCS-Algorithmus gültige Korrelationen beim Ähnlichkeitsvergleich erkennt. Ein gültiger Schwellwert wäre z.B. 0.55. Mehr zur genauen Syntax des Parameters ist unter *Allgemeine Parameter* zu Beginn dieses Abschnitts einsehbar.

4.10.4 OutgoingReferencesEqualViewingHashesCO

Diese Vergleichsfunktion führt Vergleiche anhand den unter
Abschnitt 4.10 ‚OutgoingReferencesCO..' aufgeführten allgemeinen Bedingungen durch.

Spezielle Vorbedingungen

Den Knoten müssen bereits Hashwerte in Form von Annotationen zugeordnet sein. Ist dies nicht der Fall oder Hashwerte werden in dem entsprechenden Projekt generell nicht unterstützt, wird eine Ausnahme des Typs `AnnotationNotExistsException` ausgelöst und der SiDiff-Kern terminiert den Vergleichsauftrag instantan.

Spezielle Semantik und Rückgabewert

Die resultierenden Knotenmengen werden bei dieser Vergleichsfunktion anhand der *Hashwerte* der einzelnen Knoten auf vollständige Gleichheit überprüft. Die Reihenfolge der Knoten ist dabei von Belang. Besitzen beide Knotenmengen die gleiche Anzahl an Elementen und *jeder* Knoten der Menge A hat in der gegenüberliegenden Menge B an gleicher Position ein Äquivalent mit gleichem Hashwert, so beträgt die Ähnlichkeit 1, ansonsten 0.

Spezielle Parameter

Diese Vergleichsfunktion benötigt als Parameter lediglich den Namen des zu verfolgenden Kantentyps oder * als Wildcard. Siehe dazu auch *Allgemeine Parameter*.

4.10.5 OutgoingReferencesEqualViewingMatchesCO

Diese Vergleichsfunktion führt Vergleiche anhand den unter
Abschnitt 4.10 ‚OutgoingReferencesCO..' aufgeführten allgemeinen Bedingungen durch.

Spezielle Semantik und Rückgabewert

Die resultierenden Knotenmengen werden bei dieser Vergleichsfunktion anhand von *Übereinstimmungen* der einzelnen Knoten auf vollständige Gleichheit überprüft. Die Reihenfolge der Knoten ist dabei von Belang. Besitzen beide Knotenmengen die gleiche Anzahl an Elementen und *jeder* Knoten der Menge A hat in der gegenüberliegenden Menge B an gleicher Position ein Äquivalent mit vorliegender Übereinstimmung, so beträgt die Ähnlichkeit 1, ansonsten 0.

Spezielle Parameter

Diese Vergleichsfunktion benötigt als Parameter lediglich den Namen des zu verfolgenden Kantentyps oder * als Wildcard. Siehe dazu auch *Allgemeine Parameter*.

4.10.6 OutgoingReferencesEqualViewingTypesCO

Diese Vergleichsfunktion führt Vergleiche anhand den unter
Abschnitt 4.10 ‚OutgoingReferencesCO..' aufgeführten allgemeinen Bedingungen durch.

Spezielle Semantik und Rückgabewert

Die resultierenden Knotenmengen werden bei dieser Vergleichsfunktion anhand der *Typen* der einzelnen Knoten auf vollständige Gleichheit überprüft. Die Reihenfolge der Knoten ist dabei von Belang. Besitzen beide Knotenmengen die gleiche Anzahl an Elementen und *jeder* Knoten der Menge A hat in der gegenüberliegenden Menge B an gleicher Position ein Äquivalent gleichen Knotentyps, so beträgt die Ähnlichkeit 1, ansonsten 0.

Spezielle Parameter

Diese Vergleichsfunktion benötigt als Parameter lediglich den Namen des zu verfolgenden Kantentyps oder * als Wildcard. Siehe dazu auch *Allgemeine Parameter*.

4.11 `OutgoingReference`...

Dieser Abschnitt behandelt Vergleichsfunktionen des Typs `OutgoingReferenceIO`, mit denen sich die Eingabeknoten anhand *einer* ausgehenden Kante eines bestimmtem Typs vergleichen lassen.

Allgemeine Vorbedingungen

Der im Parameter spezifizierte Kantentyp muss existieren, da ansonsten eine Ausnahme des Typs `UnknownTypeException` ausgelöst wird. Ferner dürfen die betreffenden Eingabeknoten nicht mehr als eine ausgehende Kante des entsprechenden Typs besitzen, da ansonsten eine Ausnahme vom Typ `NonUniqueException` ausgelöst wird. Damit ebenfalls nicht-triviale Ähnlichkeitswerte ermittelbar sind, sollten die Eingabeknoten eine ausgehende Kante des angegebenen Typs besitzen. Mehr zu trivialen Ähnlichkeiten unter *Allgemeiner Rückgabewert*.

Allgemeine Semantik

Die Eingabeknoten werden über den Zielknoten *genau einer* ausgehenden Referenz miteinander verglichen. Der Typ der ausgehenden Kante wird dabei im Parameter festgelegt. Mehr zur genauen Semantik und auch dem Rückgabewert ist in den Dokumentationen der spezifischen Vergleichsfunktionen weiter unten in diesem Abschnitt einsehbar.

Allgemeiner Rückgabewert

Die Bestimmung der Rückgabewerte der spezifischen implementierenden Vergleichsfunktionen unterscheidet sich unterschiedlich stark. Daher sei an dieser Stelle auf die Dokumentationen der spezifischen Vergleichsfunktionen weiter unten in diesem Abschnitt verwiesen.

Allgemein lassen sich jedoch bereits folgende triviale Ähnlichkeitsbeziehungen und deren Rückgabewerte bestimmen:

- Existiert für lediglich einen der Eingabeknoten keine ausgehende Kante des spezifizierten Typs, so beträgt die resultierende Ähnlichkeit 0.
- In dem Fall, dass *beide* Eingabeknoten keine ausgehende Kante des entsprechenden Typs beinhalten, liefern Vergleichsfunktionen dieses Typs 1 als Ähnlichkeit.

Allgemeine Parameter

Als Parameter wird bei dieser Gruppe von Vergleichsfunktionen allein der Name des ausgehenden Kantentyps, dessen Zielknoten zum Vergleich herangezogen werden sollen, übergeben.

Bemerkungen: Der Kantentyp muss existieren, da ansonsten eine Ausnahme ausgelöst wird. Ebenfalls dürfen die Eingabeknoten nicht mehr als eine ausgehende Kante des entsprechenden Typs vorweisen. Siehe dazu auch *Allgemeine Vorbedingungen*.

4.11.1 OutgoingReferenceMatchedOrSimilar

Diese Vergleichsfunktion führt Vergleiche anhand den unter
Abschnitt 4.11 ‚`OutgoingReference`..' aufgeführten allgemeinen Bedingungen durch.

Spezielle Semantik und Rückgabewert

Die Eingabeknoten werden anhand der direkten Übereinstimmung bzw. der Ähnlichkeit der sich ergebenden Zielknoten miteinander verglichen. Zunächst wird also geprüft, ob die Zielknoten bereits übereinstimmen – ist dies nicht der Fall, so erfolgt der Vergleich anhand der Ähnlichkeiten der Zielknoten untereinander.

Als Ähnlichkeit wird also letztlich bei Übereinstimmung der Zielknoten 1, andernfalls die bekannte Ähnlichkeit der beiden Zielknoten zurückgegeben.

Bemerkung: Diese Vergleichsfunktion ist nicht reflexiv.

4.11.2 OutgoingReferenceMatched

Diese Vergleichsfunktion führt Vergleiche anhand den unter
Abschnitt 4.11 ‚`OutgoingReference`..' aufgeführten allgemeinen Bedingungen durch.

Spezielle Semantik und Rückgabewert

Die Eingabeknoten werden anhand der sich ergebenden Zielknoten miteinander verglichen. Dabei werden allein *Übereinstimmungen* berücksichtigt. Es erfolgt also *kein* Vergleich aufgrund von Ähnlichkeiten.

Bei Übereinstimmung der Zielknoten wird 1, andernfalls 0 als Ähnlichkeit zurückgegeben.

4.11.3 OutgoingReferenceSimilar

Diese Vergleichsfunktion führt Vergleiche anhand den unter
Abschnitt 4.11 ‚`OutgoingReference`..' aufgeführten allgemeinen Bedingungen durch.

Spezielle Semantik und Rückgabewert

Die Eingabeknoten werden anhand der sich ergebenden Zielknoten miteinander verglichen. Dabei werden allein die bekannten *Ähnlichkeitswerte* der Zielknoten berücksichtigt. Es erfolgt also *kein* Vergleich aufgrund von Übereinstimmungen.

Als Ähnlichkeit der beiden Eingabeknoten wird somit die Ähnlichkeit der beiden Zielknoten zurückgegeben.

Bemerkung: Diese Vergleichsfunktion ist nicht reflexiv.

4.11.4 OutgoingReferenceEqualHash

Diese Vergleichsfunktion führt Vergleiche anhand den unter
Abschnitt 4.11 ‚OutgoingReference..' aufgeführten allgemeinen Bedingungen durch.

Spezielle Vorbedingungen

Den (Ziel-)Knoten müssen bereits Hashwerte in Form von Annotationen zugeordnet sein. Ist dies nicht der Fall oder Hashwerte werden in dem entsprechenden Projekt generell nicht unterstützt, wird eine Ausnahme des Typs `AnnotationNotExistsException` ausgelöst und der SiDiff-Kern terminiert den Vergleichsauftrag instantan.

Spezielle Semantik und Rückgabewert

Die Eingabeknoten werden anhand der sich ergebenden Zielknoten miteinander verglichen. Dabei wird der Vergleich auf die Identität der ermittelten *Hashwerte* der Zielknoten zurückgeführt.

Sind die Hashwerte der Zielknoten identisch, so beträgt die Ähnlichkeit der Eingabeknoten 1, ansonsten 0.

4.11.5 OutgoingReferenceEqualType

Diese Vergleichsfunktion führt Vergleiche anhand den unter
Abschnitt 4.11 ‚OutgoingReference..' aufgeführten allgemeinen Bedingungen durch.

Spezielle Semantik und Rückgabewert

Die Eingabeknoten werden anhand der sich ergebenden Zielknoten miteinander verglichen. Dabei wird der Vergleich auf Identität der *Typen* der Zielknoten zurückgeführt.

Sind die Typen der Zielknoten gleich, so beträgt die Ähnlichkeit der Eingabeknoten 1, ansonsten 0.

4.12 IncomingReferencesIO...

Dieser Abschnitt behandelt Vergleichsfunktionen des Typs `IncomingReferencesIO`, mit denen sich die Eingabeknoten anhand eingehender Kanten eines bestimmten Typs oder alternativ anhand allen eingehenden Kanten unter *Nichtbeachtung* der Reihenfolge vergleichen lassen.

Allgemeine Vorbedingungen

Der im Parameter spezifizierte Kantentyp muss existieren, da ansonsten eine Ausnahme des Typs `UnknownTypeException` ausgelöst wird. Ferner sollten die sich ergebenden Knotenmengen Elemente beeinhalten, damit nicht-triviale Ähnlichkeitswerte ermittelbar sind. Mehr zu trivialen Ähnlichkeiten unter *Allgemeiner Rückgabewert*.

Allgemeine Semantik

Die Eingabeknoten werden über eine durch den Parameter indirekt bestimmte Knotenmenge verglichen. Dabei wird die Knotenmenge zum Vergleich herangezogen, die sich aus Abarbeitung der vom Eingabeknoten eingehenden Kanten des im Parameter spezifizierten Typs ergeben. Es besteht auch die Möglichkeit *alle* eingehenden Kanten auszuwählen. Mehr zum genauen Semantik und auch Rückgabewert ist in den Dokumentationen der spezifischen Vergleichsfunktionen weiter unten in diesem Abschnitt einsehbar. Allgemein ist jedoch an dieser Stelle noch festzuhalten, dass Vergleichsfunktionen dieser Klasse im Gegensatz zu 4.13 ‚`IncomingReferencesCO`..' die Reihenfolge der gefundenen Knoten *nicht* beachten.

Allgemeiner Rückgabewert

Die Bestimmung der Rückgabewerte der spezifischen implementierenden Vergleichsfunktionen unterscheidet sich unterschiedlich stark. Daher sei an dieser Stelle auf die Dokumentationen der spezifischen Vergleichsfunktionen weiter unten in diesem Abschnitt verwiesen.

Allgemein lassen sich jedoch bereits folgende triviale Ähnlichkeitsbeziehungen und deren Rückgabewerte bestimmen:

- Existiert für lediglich eine der sich ergebenden Knotemengen keine Elemente, so beträgt die resultierende Ähnlichkeit 0.
- In dem Fall, dass *beide* resultierenden Knotenmengen keine Elemente beinhalten, lösen Vergleichsfunktionen dieses Typs eine Ausnahme des Typs `NothingToCompareException` aus.

Allgemeine Parameter

Als Parameter wird der Name des zu betrachtenden, eingehenden Kantentyps angegeben. Ferner ist es durch Verwendung eines `*` möglich, *alle* eingehenden Kanten und somit letztlich alle Quellknoten zum Vergleich heranzuziehen.

Folgend zwei Beispiele für gültige Parameterwerte:

- Eingehende Kanten des Typs `typA` betrachten: `typA`
- Alle eingehenden Kanten bzw. Quellknoten heranziehen: `*`

Bemerkung: Sofern ein spezieller Kantentyp angegeben wird, muss dieser existent sein, da ansonsten eine Ausnahme ausgelöst wird.

4.12.1 IncomingReferencesMatchedOrSimilarIO

Diese Vergleichsfunktion führt Vergleiche anhand den unter
Abschnitt 4.12 ‚IncomingReferencesIO..' aufgeführten allgemeinen Bedingungen durch.

Spezielle Semantik und Rückgabewert

Die aus Abarbeitung des im Parameter angegebenen eingehenden Kantentyps resultierenden Knotenmengen werden bei dieser Vergleichsfunktion anhand von sowohl Übereinstimmungen als auch Ähnlichkeiten miteinander verglichen.

- Wird bei einem Knoten eine Übereinstimmung in der gegenüberliegenden Menge gefunden, geht dies anteilig mit dem Wert 1 in den Vergleich ein.
- Wenn keine Übereinstimmung für einen Kindknoten gefunden wurde, so geht der Knoten mit der höchsten Ähnlichkeit in den Vergleich ein.

Als Gesamt-Ähnlichkeit wird letztlich der Quotient aus den kumulierten Einzel-Ähnlichkeiten und der Größe der Knotenmenge mit den meisten Elementen zurückgegeben.

Bemerkungen:

- Diese Vergleichsfunktion ist nicht reflexiv!
- Die Knoten ermittelter Korrelationen werden vom weiteren Vergleich ausgeschlossen, sodass diese nicht ein weiteres mal anderen Knoten zugeordnet werden können.
- Da es sich lediglich um Vergleichs-*Heuristiken* handelt, besteht bei Anwendung von Ähnlichkeits-Korrelationen ebenfalls die Möglichkeit von suboptimalen Zuordnungen von Knotenpaaren – also dass bspw. ein Knoten A mit einem Knoten B als korrelierend betrachtet wird, obwohl er im weiteren Vergleich einem Knoten C wesentlich besser entsprechen würde.

4.12.2 IncomingReferencesMatchedIO

Diese Vergleichsfunktion führt Vergleiche anhand den unter
Abschnitt 4.12 ‚IncomingReferencesIO..' aufgeführten allgemeinen Bedingungen durch.

Spezielle Semantik und Rückgabewert

Die aus dem Pfadausdruck resultierenden Knotenmengen werden bei dieser Vergleichsfunktion allein anhand von *Übereinstimmungen* miteinander verglichen.

- Wird bei einem Knoten eine Übereinstimmung in der gegenüberliegenden Menge gefunden, geht dies anteilig mit dem Wert 1 in den Vergleich ein.
- Ist für einen Knoten allerdings eine Übereinstimmung *außerhalb* der anderen Knotenmenge existent, so geht dies anteilig mit -1 in die Gesamt-Ähnlichkeit ein. Eine „fehlerhafte" Übereinstimmung resultiert also in einer *Minderung* der Ähnlichkeit.
- Wenn keine Übereinstimmung für einen Knoten gefunden wurde, so geschieht nichts.

Als Gesamt-Ähnlichkeit wird letztlich der Quotient aus den kumulierten Einzel-Ähnlichkeiten und der Größe der resultierenden Knotenmenge mit den meisten Elementen zurückgegeben. Sofern der Betrag der kumulierten Einzel-Ähnlichkeiten aufgrund vieler „fehlerhafter" Übereinstimmungen allerdings negativ ist, beträgt die Gesamtähnlichkeit 0.

Bemerkung: Diese Vergleichsfunktion ist nicht reflexiv!

4.12.3 IncomingReferencesSimilarIO

Diese Vergleichsfunktion führt Vergleiche anhand den unter
Abschnitt 4.12 ‚IncomingReferencesIO..' aufgeführten allgemeinen Bedingungen durch.

Spezielle Semantik und Rückgabewert

Die resultierenden Knotenmengen werden bei dieser Vergleichsfunktion allein anhand von bereits bekannten *Ähnlichkeiten* miteinander verglichen. Dabei wird für jeden Knoten der Menge A die beste Übereinstimmung in der gegenüberliegen Knotenmenge B gesucht und die Ähnlichkeiten von der Knoten von Menge A zu Menge B kumuliert.

Als Gesamt-Ähnlichkeit wird letztlich der Quotient aus den kumulierten Einzel-Ähnlichkeiten und der Größe der resultierenden Knotenmenge mit den meisten Elementen zurückgegeben.

Bemerkungen:

- Diese Vergleichsfunktion ist nicht reflexiv!
- Die Knoten ermittelter Korrelationen werden vom weiteren Vergleich ausgeschlossen, sodass diese nicht ein weiteres mal anderen Knoten zugeordnet werden können.
- Da es sich lediglich um Vergleichs-*Heuristiken* handelt, besteht ebenfalls die Möglichkeit von suboptimalen Zuordnungen von Knotenpaaren – also dass bspw. ein Knoten A mit einem Knoten B als korrelierend betrachtet wird, obwohl er im weiteren Vergleich einem Knoten C wesentlich besser entsprechen würde.

4.12.4 IncomingReferencesEqualViewingHashesIO

Diese Vergleichsfunktion führt Vergleiche anhand den unter
Abschnitt 4.12 ‚IncomingReferencesIO..' aufgeführten allgemeinen Bedingungen durch.

Spezielle Vorbedingungen

Den Knoten müssen bereits Hashwerte in Form von Annotationen zugeordnet sein. Ist dies nicht der Fall oder Hashwerte werden in dem entsprechenden Projekt generell nicht unterstützt, wird eine Ausnahme des Typs `AnnotationNotExistsException` ausgelöst und der SiDiff-Kern terminiert den Vergleichsauftrag instantan.

Spezielle Semantik und Rückgabewert

Die resultierenden Knotenmengen werden bei dieser Vergleichsfunktion anhand der *Hashwerte* der einzelnen Knoten auf vollständige Gleichheit überprüft. Besitzen beide Knotenmengen die gleiche Anzahl an Elementen und *jeder* Knoten hat in der gegenüberliegen Knotenmenge ein Äquivalent mit gleichem Hashwert, so beträgt die Ähnlichkeit 1, ansonsten 0.

4.12.5 IncomingReferencesEqualViewingMatchesIO

Diese Vergleichsfunktion führt Vergleiche anhand den unter
Abschnitt 4.12 ‚IncomingReferencesIO..' aufgeführten allgemeinen Bedingungen durch.

Spezielle Semantik und Rückgabewert

Die resultierenden Knotenmengen werden bei dieser Vergleichsfunktion anhand von bereits gefundenen *Übereinstimmungen* der einzelnen Knoten auf vollständige Übereinstimmung überprüft. Besitzen beide Knotenmengen die gleiche Anzahl an Elementen und *jeder* Knoten hat in der gegenüberliegen Knotenmenge eine Übereinstimmung, so beträgt die Ähnlichkeit 1, ansonsten 0.

4.12.6 IncomingReferencesEqualViewingTypesIO

Diese Vergleichsfunktion führt Vergleiche anhand den unter
Abschnitt 4.12 ‚IncomingReferencesIO..' aufgeführten allgemeinen Bedingungen durch.

Spezielle Semantik und Rückgabewert

Die resultierenden Knotenmengen werden bei dieser Vergleichsfunktion anhand der *Typen* der einzelnen Knoten auf vollständige Gleichheit überprüft. Besitzen beide Knotenmengen die gleiche Anzahl an Elementen und *jeder* Knoten hat in der gegenüberliegen Knotenmenge ein Äquivalent mit gleichem Knotentyp, so beträgt die Ähnlichkeit 1, ansonsten 0.

4.13 IncomingReferencesCO...

Dieser Abschnitt behandelt Vergleichsfunktionen des Typs `IncomingReferencesCO`, mit denen sich die Eingabeknoten anhand eingehender Kanten eines bestimmtem Typs oder alternativ anhand allen eingehenden Kanten unter *Beachtung* der Reihenfolge vergleichen lassen.

Allgemeine Vorbedingungen

Der im Parameter spezifizierte Kantentyp muss existieren, da ansonsten eine Ausnahme des Typs `UnknownTypeException` ausgelöst wird. Ferner sollten die sich ergebenden Knotenmengen Elemente beeinhalten, damit nicht-triviale Ähnlichkeitswerte ermittelbar sind. Mehr zu trivialen Ähnlichkeiten unter *Allgemeiner Rückgabewert*.

Allgemeine Semantik

Die Eingabeknoten werden über eine durch den Parameter indirekt bestimmte Knotenmenge verglichen. Dabei wird die Knotenmenge zum Vergleich herangezogen, die sich aus Abarbeitung der vom Eingabeknoten eingehenden Kanten des im Parameter spezifizierten Typs ergeben. Es besteht auch die Möglichkeit *alle* eingehenden Kanten auszuwählen. Mehr zum genauen Semantik und auch Rückgabewert ist in den Dokumentationen der spezifischen Vergleichsfunktionen weiter unten in diesem Abschnitt einsehbar. Allgemein ist jedoch an dieser Stelle noch festzuhalten, dass Vergleichsfunktionen dieser Klasse im Gegensatz zu 4.12 ‚`IncomingReferencesIO`..' die Reihenfolge der gefundenen Knoten mit in den Vergleich einbeziehen.

Allgemeiner Rückgabewert

Die Bestimmung der Rückgabewerte der spezifischen implementierenden Vergleichsfunktionen unterscheidet sich unterschiedlich stark. Daher sei an dieser Stelle auf die Dokumentationen der spezifischen Vergleichsfunktionen weiter unten in diesem Abschnitt verwiesen.

Allgemein lassen sich jedoch bereits folgende triviale Ähnlichkeitsbeziehungen und deren Rückgabewerte bestimmen:

- Existiert für lediglich eine der sich ergebenden Knotemengen keine Elemente, so beträgt die resultierende Ähnlichkeit 0.
- In dem Fall, dass *beide* resultierenden Knotenmengen keine Elemente beinhalten, lösen Vergleichsfunktionen dieses Typs eine Ausnahme des Typs `NothingToCompareException` aus.

Allgemeine Parameter

Als Parameter wird der Name des zu betrachtenden, eingehenden Kantentyps angegeben. Ferner ist es durch Verwendung eines `*` möglich, *alle* eingehenden Kanten und somit letztlich alle Quellknoten zum Vergleich heranzuziehen.

Manche implementierenden Vergleichsfunktionen dieser Kategorie benötigen jedoch ebenfalls einen Schwellwert als zweiten Parameter. Dieser schließt sich an den angegebenen Kantentyp bzw. `*` an und wird durch ein `;` von eben jenem abgetrennt. Sofern eine Vergleichsfunktion auf die Angabe dieses Schwellwerts angewiesen ist, wird dieses in der Dokumentation der einzelnen Vergleichsfunktion explizit vermerkt. Weitere Informationen zur Semantik des Schwellwerts sind ebenfalls an dieser Stelle zu finden.

Bitte umblättern.

Folgend zwei Beispiele für gültige Parameterwerte mit und ohne Schwellwert:

ohne Sw.	mit Sw.	Bedeutung
`typA`	`typA;0.3`	Eingehende Kanten des Typs `typA` betrachten
`*`	`*;0.25`	Alle eingehenden Kanten bzw. Quellknoten heranziehen

Bemerkung: Sofern ein spezieller Kantentyp angegeben wird, muss dieser existent sein, da ansonsten eine Ausnahme ausgelöst wird.

4.13.1 IncomingReferencesMatchedOrSimilarCO

Diese Vergleichsfunktion führt Vergleiche anhand den unter
Abschnitt 4.13 ‚IncomingReferencesCO..' aufgeführten allgemeinen Bedingungen durch.

Spezielle Semantik und Rückgabewert

Die resultierenden Knotenmengen werden bei dieser Vergleichsfunktion anhand von sowohl Übereinstimmungen als auch Ähnlichkeiten miteinander verglichen. Die Reihenfolge der Knoten ist dabei von Belang, sodass der LCS-Algorithmus (Longest Common Subsequence) Anwendung findet. Dazu wird innerhalb der Knotenmengen die längste gemeinsame Teilsequenz gesucht. Genaueres in der folgenden Auflistung.

- Eine gültige Korrelation zwischen zwei Knoten wird erkannt, wenn die beiden Knoten bereits als übereinstimmend gekennzeichnet sind.
- Liegt keine Übereinstimmung vor, werden die Ähnlichkeiten vom Knoten aus Menge A mit dem gegenüberliegenden Knoten in Menge B betrachtet. Liegt die Ähnlichkeit über dem unter *Spezielle Parameter* beschriebenen Schwellwert oder entspricht genau diesem, wird auf diese Weise eine Korrelation festgestellt.
- Trifft keine der beiden Bedingungen zu, endet eine längste Teilsequenz an diesem Knotenpaar.

Als Gesamt-Ähnlichkeit wird letztlich der Quotient aus der Länge der längsten gefundenen Teilsequenz und der Größe der resultierenden Knotenmenge mit den meisten Elementen gebildet und zurückgegeben.

Bemerkung: Diese Vergleichsfunktion ist nicht reflexiv!

Spezielle Parameter

Diese Vergleichsfunktion benötigt neben dem Name des Kantentyps als Parameter auch einen Schwellwert im Bereich von 0 bis 1, der angibt, ab welcher Ähnlichkeit der LCS-Algorithmus gültige Korrelationen beim Ähnlichkeitsvergleich erkennt. Ein gültiger Schwellwert wäre z.B. 0.55. Mehr zur genauen Syntax des Parameters ist unter *Allgemeine Parameter* zu Beginn dieses Abschnitts einsehbar.

4.13.2 IncomingReferencesMatchedCO

Diese Vergleichsfunktion führt Vergleiche anhand den unter
Abschnitt 4.13 ‚IncomingReferencesCO..' aufgeführten allgemeinen Bedingungen durch.

Spezielle Semantik und Rückgabewert

Die resultierenden Knotenmengen werden bei dieser Vergleichsfunktion allein anhand von *Übereinstimmungen* miteinander verglichen. Die Reihenfolge der Knoten ist dabei von Belang, sodass der LCS-Algorithmus (Longest Common Subsequence) Anwendung findet. Dazu wird innerhalb der Knotenmengen die längste gemeinsame Teilsequenz gesucht. Genaueres in der folgenden Auflistung.

- Eine gültige Korrelation zwischen zwei Knoten wird erkannt, wenn die beiden Knoten bereits als Übereinstimmung gekennzeichnet sind.
- Liegt keine Übereinstimmung vor, endet eine längste Teilsequenz an diesem Knotenpaar.

Als Gesamt-Ähnlichkeit wird letztlich der Quotient aus der Länge der längsten gefundenen Teilsequenz und der Größe der resultierenden Knotenmenge mit den meisten Elementen gebildet und zurückgegeben.

Spezielle Parameter

Diese Vergleichsfunktion benötigt als Parameter lediglich den Namen des zu verfolgenden Kantentyps oder * als Wildcard. Siehe dazu auch *Allgemeine Parameter*.

4.13.3 IncomingReferencesSimilarCO

Diese Vergleichsfunktion führt Vergleiche anhand den unter
Abschnitt 4.13 ‚IncomingReferencesCO..' aufgeführten allgemeinen Bedingungen durch.

Spezielle Semantik und Rückgabewert

Die resultierenden Knotenmengen werden bei dieser Vergleichsfunktion allein anhand von *Ähnlichkeiten* miteinander verglichen. Die Reihenfolge der Knoten ist dabei von Belang, sodass der LCS-Algorithmus (Longest Common Subsequence) Anwendung findet. Dazu wird innerhalb der Knotenmengen die längste gemeinsame Teilsequenz gesucht. Genaueres in der folgenden Auflistung.

- Eine gültige Korrelation zwischen zwei Knoten wird erkannt, wenn die Ähnlichkeit des Knotens aus Menge A mit dem gegenüberliegenden Knoten in Menge B über dem unter *Spezielle Parameter* beschriebenen Schwellwert liegt oder genau diesem entspricht.
- Liegt keine solche Korrelation vor, endet eine mögliche längste Teilsequenz an diesem Knotenpaar.

Als Gesamt-Ähnlichkeit wird letztlich der Quotient aus der Länge der längsten gefundenen Teilsequenz und der Größe der resultierenden Knotenmenge mit den meisten Elementen gebildet und zurückgegeben.

Bemerkung: Diese Vergleichsfunktion ist nicht reflexiv!

Spezielle Parameter

Diese Vergleichsfunktion benötigt neben dem Name des Kantentyps als Parameter auch einen Schwellwert im Bereich von 0 bis 1, der angibt, ab welcher Ähnlichkeit der LCS-Algorithmus gültige Korrelationen beim Ähnlichkeitsvergleich erkennt. Ein gültiger Schwellwert wäre z.B. 0.55. Mehr zur genauen Syntax des Parameters ist unter *Allgemeine Parameter* zu Beginn dieses Abschnitts einsehbar.

4.13.4 IncomingReferencesEqualViewingHashesC0

Diese Vergleichsfunktion führt Vergleiche anhand den unter
Abschnitt 4.13 ‚IncomingReferencesC0..' aufgeführten allgemeinen Bedingungen durch.

Spezielle Vorbedingungen

Den Knoten müssen bereits Hashwerte in Form von Annotationen zugeordnet sein. Ist dies nicht der Fall oder Hashwerte werden in dem entsprechenden Projekt generell nicht unterstützt, wird eine Ausnahme des Typs `AnnotationNotExistsException` ausgelöst und der SiDiff-Kern terminiert den Vergleichsauftrag instantan.

Spezielle Semantik und Rückgabewert

Die resultierenden Knotenmengen werden bei dieser Vergleichsfunktion anhand der *Hashwerte* der einzelnen Knoten auf vollständige Gleichheit überprüft. Die Reihenfolge der Knoten ist dabei von Belang. Besitzen beide Knotenmengen die gleiche Anzahl an Elementen und *jeder* Knoten der Menge A hat in der gegenüberliegenden Menge B an gleicher Position ein Äquivalent mit gleichem Hashwert, so beträgt die Ähnlichkeit 1, ansonsten 0.

Spezielle Parameter

Diese Vergleichsfunktion benötigt als Parameter lediglich den Namen des zu verfolgenden Kantentyps oder * als Wildcard. Siehe dazu auch *Allgemeine Parameter*.

4.13.5 IncomingReferencesEqualViewingMatchesC0

Diese Vergleichsfunktion führt Vergleiche anhand den unter
Abschnitt 4.13 ‚IncomingReferencesC0..' aufgeführten allgemeinen Bedingungen durch.

Spezielle Semantik und Rückgabewert

Die resultierenden Knotenmengen werden bei dieser Vergleichsfunktion anhand von *Übereinstimmungen* der einzelnen Knoten auf vollständige Gleichheit überprüft. Die Reihenfolge der Knoten ist dabei von Belang. Besitzen beide Knotenmengen die gleiche Anzahl an Elementen und *jeder* Knoten der Menge A hat in der gegenüberliegenden Menge B an gleicher Position ein Äquivalent mit vorliegender Übereinstimmung, so beträgt die Ähnlichkeit 1, ansonsten 0.

Spezielle Parameter

Diese Vergleichsfunktion benötigt als Parameter lediglich den Namen des zu verfolgenden Kantentyps oder * als Wildcard. Siehe dazu auch *Allgemeine Parameter*.

4.13.6 IncomingReferencesEqualViewingTypesC0

Diese Vergleichsfunktion führt Vergleiche anhand den unter
Abschnitt 4.13 ‚IncomingReferencesC0..' aufgeführten allgemeinen Bedingungen durch.

Spezielle Semantik und Rückgabewert

Die resultierenden Knotenmengen werden bei dieser Vergleichsfunktion anhand der *Typen* der einzelnen Knoten auf vollständige Gleichheit überprüft. Die Reihenfolge der Knoten ist dabei von Belang. Besitzen beide Knotenmengen die gleiche Anzahl an Elementen und *jeder* Knoten der Menge A hat in der gegenüberliegenden Menge B an gleicher Position ein Äquivalent gleichen Knotentyps, so beträgt die Ähnlichkeit 1, ansonsten 0.

Spezielle Parameter

Diese Vergleichsfunktion benötigt als Parameter lediglich den Namen des zu verfolgenden Kantentyps oder * als Wildcard. Siehe dazu auch *Allgemeine Parameter*.

4.14 IncomingReference...

Dieser Abschnitt behandelt Vergleichsfunktionen des Typs `IncomingReferenceIO`, mit denen sich die Eingabeknoten anhand *einer* eingehenden Kante eines bestimmtem Typs vergleichen lassen.

Allgemeine Vorbedingungen

Der im Parameter spezifizierte Kantentyp muss existieren, da ansonsten eine Ausnahme des Typs `UnknownTypeException` ausgelöst wird. Ferner dürfen die betreffenden Eingabeknoten nicht mehr als eine eingehende Kante des entsprechenden Typs besitzen, da ansonsten eine Ausnahme vom Typ `NonUniqueException` ausgelöst wird. Damit ebenfalls nicht-triviale Ähnlichkeitswerte ermittelbar sind, sollten die Eingabeknoten eine eingehende Kante des angegebenen Typs besitzen. Mehr zu trivialen Ähnlichkeiten unter *Allgemeiner Rückgabewert*.

Allgemeine Semantik

Die Eingabeknoten werden über den Quellknoten *genau einer* eingehenden Referenz miteinander verglichen. Der Typ der eingehenden Kante wird dabei im Parameter festgelegt. Mehr zur genauen Semantik und auch dem Rückgabewert ist in den Dokumentationen der spezifischen Vergleichsfunktionen weiter unten in diesem Abschnitt einsehbar.

Allgemeiner Rückgabewert

Die Bestimmung der Rückgabewerte der spezifischen implementierenden Vergleichsfunktionen unterscheidet sich unterschiedlich stark. Daher sei an dieser Stelle auf die Dokumentationen der spezifischen Vergleichsfunktionen weiter unten in diesem Abschnitt verwiesen.

Allgemein lassen sich jedoch bereits folgende triviale Ähnlichkeitsbeziehungen und deren Rückgabewerte bestimmen:

- Existiert für lediglich einen der Eingabeknoten keine eingehende Kante des spezifizierten Typs, so beträgt die resultierende Ähnlichkeit 0.
- In dem Fall, dass *beide* Eingabeknoten keine eingehende Kante des entsprechenden Typs beinhalten, liefern Vergleichsfunktionen dieses Typs 1 als Ähnlichkeit.

Allgemeine Parameter

Als Parameter wird bei dieser Gruppe von Vergleichsfunktionen allein der Name des eingehenden Kantentyps, dessen Quellknoten zum Vergleich herangezogen werden sollen, übergeben.

Bemerkungen: Der Kantentyp muss existieren, da ansonsten eine Ausnahme ausgelöst wird. Ebenfalls dürfen die Eingabeknoten nicht mehr als eine eingehende Kante des entsprechenden Typs vorweisen. Siehe dazu auch *Allgemeine Vorbedingungen*.

4.14.1 IncomingReferenceMatchedOrSimilar

Diese Vergleichsfunktion führt Vergleiche anhand den unter
Abschnitt 4.14 ‚IncomingReference..' aufgeführten allgemeinen Bedingungen durch.

Spezielle Semantik und Rückgabewert

Die Eingabeknoten werden anhand der direkten Übereinstimmung bzw. der Ähnlichkeit der sich ergebenden Quellknoten miteinander verglichen. Zunächst wird also geprüft, ob die Quellknoten bereits übereinstimmen – ist dies nicht der Fall, so erfolgt der Vergleich anhand der Ähnlichkeiten der Quellknoten untereinander.

Als Ähnlichkeit wird also letztlich bei Übereinstimmung der Quellknoten 1, andernfalls die bekannte Ähnlichkeit der beiden Quellknoten zurückgegeben.

Bemerkung: Diese Vergleichsfunktion ist nicht reflexiv.

4.14.2 IncomingReferenceMatched

Diese Vergleichsfunktion führt Vergleiche anhand den unter
Abschnitt 4.14 ‚IncomingReference..' aufgeführten allgemeinen Bedingungen durch.

Spezielle Semantik und Rückgabewert

Die Eingabeknoten werden anhand der sich ergebenden Quellknoten miteinander verglichen. Dabei werden allein *Übereinstimmungen* berücksichtigt. Es erfolgt also *kein* Vergleich aufgrund von Ähnlichkeiten.

Bei Übereinstimmung der Quellknoten wird 1, andernfalls 0 als Ähnlichkeit zurückgegeben.

4.14.3 IncomingReferenceSimilar

Diese Vergleichsfunktion führt Vergleiche anhand den unter
Abschnitt 4.14 ‚IncomingReference..' aufgeführten allgemeinen Bedingungen durch.

Spezielle Semantik und Rückgabewert

Die Eingabeknoten werden anhand der sich ergebenden Quellknoten miteinander verglichen. Dabei werden allein die bekannten *Ähnlichkeitswerte* der Quellknoten berücksichtigt. Es erfolgt also *kein* Vergleich aufgrund von Übereinstimmungen.

Als Ähnlichkeit der beiden Eingabeknoten wird somit die Ähnlichkeit der beiden Quellknoten zurückgegeben.

Bemerkung: Diese Vergleichsfunktion ist nicht reflexiv.

4.14.4 IncomingReferenceEqualHash

Diese Vergleichsfunktion führt Vergleiche anhand den unter
Abschnitt 4.14 ‚IncomingReference..' aufgeführten allgemeinen Bedingungen durch.

Spezielle Vorbedingungen

Den (Ziel-)Knoten müssen bereits Hashwerte in Form von Annotationen zugeordnet sein. Ist dies nicht der Fall oder Hashwerte werden in dem entsprechenden Projekt generell nicht unterstützt, wird eine Ausnahme des Typs `AnnotationNotExistsException` ausgelöst und der SiDiff-Kern terminiert den Vergleichsauftrag instantan.

Spezielle Semantik und Rückgabewert

Die Eingabeknoten werden anhand der sich ergebenden Quellknoten miteinander verglichen. Dabei wird der Vergleich auf die Identität der ermittelten *Hashwerte* der Quellknoten zurückgeführt.

Sind die Hashwerte der Quellknoten identisch, so beträgt die Ähnlichkeit der Eingabeknoten 1, ansonsten 0.

4.14.5 IncomingReferenceEqualType

Diese Vergleichsfunktion führt Vergleiche anhand den unter
Abschnitt 4.14 ‚IncomingReference..' aufgeführten allgemeinen Bedingungen durch.

Spezielle Semantik und Rückgabewert

Die Eingabeknoten werden anhand der sich ergebenden Quellknoten miteinander verglichen. Dabei wird der Vergleich auf Identität der *Typen* der Quellknoten zurückgeführt.

Sind die Typen der Quellknoten gleich, so beträgt die Ähnlichkeit der Eingabeknoten 1, ansonsten 0.