

## **TermNet und GermaTermNet in OWL**

Projekt  
**Hypertextualisierung auf  
textgrammatischer Grundlage**  
([www.hytext.info](http://www.hytext.info))

Bianca Selzam

2009

# Inhaltsverzeichnis

1	Klassenmodell.....	3
1.1	Modellierungsidee.....	3
1.2	Klassen.....	3
1.2.1	Klassenhierarchie.....	3
1.2.2	Disjunktheit zwischen TermConcept-Klassen.....	3
1.3	Instanzen.....	4
1.3.1	Instanzen der TermSet-Klassen.....	4
1.3.2	Instanzen der TermConcept-Klassen.....	4
1.3.3	Instanzen der Klasse TermForm.....	4
1.4	Properties.....	5
1.4.1	Einführung von Restrictions.....	5
1.4.2	Subclass-Superclass-Relationen zwischen Termini.....	5
1.4.3	Lexikalische Relationen.....	6
1.4.4	Konzeptuelle Relationen.....	6
1.4.5	Member-Relationen.....	7
1.4.6	Type-Token-Relationen.....	7
1.4.7	Visualisierung und Übersicht der Relationen.....	8
2	Instanzenmodell.....	9
2.1	Modellierungsidee.....	9
2.2	Klassen.....	9
2.3	Instanzen.....	10
2.3.1	Instanzen der Klasse TermSet.....	10
2.3.2	Instanzen der Klasse TermConcept.....	10
2.3.3	Instanzen der Klasse TermForm.....	10
2.3.4	Instanzen der Klasse TermOccurrence.....	10
2.4	Properties.....	11
2.4.1	Relationen analog zum Klassenmodell.....	11
2.4.2	isDisjointWith-Relation.....	11
2.4.3	isBroaderTerm-isNarrowerTerm-Relationen.....	11
2.4.4	Type-Token-Relationen.....	12
2.4.5	Lexikalisierungsrelationen.....	12
2.4.6	Verwendungsrelationen.....	12
2.4.7	Visualisierung und Übersicht der Relationen.....	13
3	OWL Full-Modell.....	14
3.1	Klassen.....	14
3.2	Instanzen.....	14
3.3	Properties.....	15
4	Hybridmodell.....	15
4.1	Vereinigung der beiden Ressourcen.....	15
4.2	Plugin-Relationen.....	15
4.2.1	attachedToNearSynonym.....	16
4.2.2	attachedToGeneralConcept.....	16
4.2.3	attachedToHolonym.....	17
4.2.4	Übersicht der Plugin-Relationen.....	17
5	Klassenmodell.....	17
5.1	attachedToNearSynonym.....	17
5.2	attachedToGeneralConcept.....	18
5.3	attachedToHolonym.....	18
6	Instanzenmodell.....	19
6.1	attachedToNearSynonym.....	19
6.2	attachedToGeneralConcept.....	19
6.3	attachedToHolonym.....	20
7	OWL Full-Modell.....	20

# A) TermNet

## 1 Klassenmodell

In diesem Kapitel wird das Konzept des Klassenmodells erläutert und die darin enthaltenen Klassen, Properties und Instanzen vorgestellt.

### 1.1 Modellierungsidee

Das Klassenmodell des TermNets versteht TermSets und Termini als Konzepte. Hierfür wird für jedes konkrete TermSet und jeden Terminus eine eigene OWL-Klasse definiert. Die graphischen Erscheinungsformen aller Termini werden gesammelt als Instanzen einer weiteren Klasse verstanden; diese werden mittels geeigneter Relationen mit den jeweiligen Termini, die sie lexikalisieren, verbunden. Die Verwendungsinstanzen der Termini werden zusätzlich als Individuen der jeweiligen Term-Klasse definiert.

Das Klassenmodell verwendet wie auch das Instanzenmodell (vgl. Kap. 2) die OWL-Subsprache OWL DL.

### 1.2 Klassen

Die im Klassenmodell verwendeten Klassen vom Typ `OWLClass` werden in den folgenden Abschnitten näher beschrieben.

#### 1.2.1 Klassenhierarchie

Die Oberklasse aller im Klassenmodell verwendeten Klassen bildet das Konzept `LSP_Domain_Hypermedia_and_Texttechnology`. Diese besitzt die Subklassen `TermSet`, `TermConcept` und `TermForm`, welche respektive als Repräsentanten der TermSets, Termini und graphischen Erscheinungsformen dienen.

Unterhalb der Klasse `TermSet` existiert für jedes der 206 TermSets eine eigene Klasse, deren Namen sich aus dem Präfix `TermSet_` und der Bezeichnung des jeweiligen TermSets zusammensetzen.

Die Klasse `TermConcept` hingegen bildet die Basisklasse aller Termini. Für jeden der 423 Termini besitzt diese eine Subklasse, deren Name aus dem Präfix `TermConcept_` und der entsprechenden Bezeichnung des Terms besteht. Ist zwischen zwei Termini eine Inklusionsbeziehung feststellbar (z. B. zwischen `TermConcept_OWL-Standard` und `TermConcept_Standard`), so wird der speziellere Terminus als Subklasse des anderen modelliert mittels des Konstrukts `<rdfs:subClassOf>`.

Nicht zuletzt repräsentiert die Klasse `TermForm` die Menge aller graphischen Erscheinungsformen der Termini, die als Subklassen von `TermConcept` modelliert sind. `TermForm` besitzt keine weiteren Unterklassen; in dieser Klasse werden stattdessen die orthographischen Varianten der Termini als Instanzen modelliert (vgl. Kap. 1.3).

#### 1.2.2 Disjunktheit zwischen *TermConcept*-Klassen

Sind zwei Subklassen von `TermConcept` disjunkt, d. h. eine konkrete Instanz kann nicht gleichzeitig zu beiden Klassen gehören, so sind diese mittels des OWL-Konstrukts `<owl:disjointwith>` gekennzeichnet, um zu zeigen, dass diese beiden Klassen sich gegenseitig ausschließen.

##### Beispiel:

`TermConcept_1-zu-1-Link` ist disjunkt zu `TermConcept_1-zu-n-Link` und `TermConcept_n-zu-m-Link`. `TermConcept_1-zu-n-Link` wiederum ist disjunkt zu `TermConcept_1-zu-1-Link` und `TermConcept_n-zu-m-Link`, und `TermConcept_n-zu-m-Link` ist ebenfalls zu den beiden übrigen Klassen disjunkt. Dies bedeutet, dass eine konkrete Instanz eines Links entweder vom Typ 1:1, 1:n oder n:m sein kann, niemals aber von zwei oder gar drei Typen gleichzeitig.

**OWL-Code:**

```

<owl:Class rdf:ID="TermConcept_1-zu-1-Link">
  <owl:disjointwith>
    <owl:Class rdf:ID="TermConcept_1-zu-n-Link"/>
    <owl:Class rdf:ID="TermConcept_n-zu-m-Link"/>
  </owl:disjointwith>
</owl:Class>

```

## 1.3 Instanzen

Die Individuen der in Kap. 1.2 vorgestellten Klassen werden in den nachfolgenden drei Unterkapiteln geordnet nach den Klassen, zu denen sie gehören, beschrieben.

### 1.3.1 Instanzen der *TermSet*-Klassen

Das Anlegen von Instanzen in den Subklassen von *TermSet* war in der ursprünglichen Modellierung zunächst nicht vorgesehen. Da sich die Abfrage des *TermNets* mit dem Reasoner-Tool *RacerPro*<sup>1</sup> jedoch ohne das Vorhandensein von *TermSet*-Instanzen nur bedingt durchführen ließ, wurde für jede *TermSet*-Klasse je eine Instanz angelegt, deren Bezeichnung neben dem Präfix *TS\_* den Namen des *TermSets* enthält.

**Beispiel:**

Die *TermSet*-Klasse *TermSet\_1-zu-n-Link* besitzt zur leichteren Verarbeitung des *TermNets* mit einem Reasoner die Instanz *TS\_1-zu-n-Link*.

**OWL-Code:**

```

<TermSet_1-zu-n-Link rdf:ID="TS_1-zu-n-Link">

```

### 1.3.2 Instanzen der *TermConcept*-Klassen

Jede Instanz einer *TermConcept*-Klasse wird als konkrete Verwendung des entsprechenden Terms in einem Text angesehen. Der Name einer solchen Instanz besteht aus dem Präfix *URI\_TermInstance\_* (zur Verdeutlichung, dass es sich um die Referenz auf eine bestimmte Textstelle handelt), dem Namen des *TermConcepts* und einer fortlaufenden Nummer, beginnend bei 1.

**Beispiel:**

Die Klasse *TermConcept\_Hyperlink* besitzt die Instanzen *URI\_TermInstance\_Hyper-Link\_1*, *URI\_TermInstance\_Hyperlink\_1* und *URI\_TermInstance\_hyperlink\_1*; für alle orthographischen Varianten des Konzeptnamens werden also getrennt nummerierte Instanzen angelegt.

**OWL-Code:**

```

<TermConcept_Hyperlink rdf:ID="URI_TermInstance_Hyper-Link_1">
<TermConcept_Hyperlink rdf:ID="URI_TermInstance_Hyperlink_1">
<TermConcept_Hyperlink rdf:ID="URI_TermInstance_hyperlink_1">

```

### 1.3.3 Instanzen der Klasse *TermForm*

Die Klasse *TermForm* repräsentiert alle graphischen Erscheinungsformen der *TermConcepts*, d. h. alle orthographischen Varianten der Termini. Deshalb wird in dieser Klasse für alle 471 Schreibweisen der 423 Termini je eine Instanz angelegt, deren Name lediglich aus der Bezeichnung des Terms besteht.

**Beispiel:**

Die beiden Schreibweisen *Link* und *link* des Konzepts *Link* werden als zwei getrennte *TermForm*-Individuen *Link* und *link* modelliert.

**OWL-Code:**

```

<TermForm rdf:ID="Link">
<TermForm rdf:ID="link">

```

<sup>1</sup> <http://www.racer-systems.com/de/index.phtml?lang>

## 1.4 Properties

Die Verbindungen zwischen den Instanzen der in Kap. 1.2 beschriebenen Klassen werden durch verschiedene Typen von ObjectProperties (Konstrukt `<owl:ObjectProperty>`) hergestellt, welche in den Kapiteln 1.4.3 bis 1.4.6 näher spezifiziert werden. Zuvor werden jedoch einige grundlegende Charakteristika der Modellierung vorgestellt, die sich zum Teil aus der verwendeten Beschreibungssprache OWL DL ergeben.

### 1.4.1 Einführung von Restrictions

In TermNet I waren sowohl die lexikalischen als auch die konzeptuellen Relationen als Relationen zwischen Konzepten definiert worden. Analog dazu stellen auch die Properties in TermNet II (`hasMember`, `isMemberOf`, `isHyponymOf`, `isHypernymOf`, `isHolonymOf`, `isMeronymOf`, `isAbbreviationOf` und `isExpansionOf`) ausnahmslos Relationen zwischen Klassen dar. Da OWL DL jedoch nur die Definition von Relationen zwischen Individuen erlaubt, kann dieser Modellierungsansatz nicht ohne Modifikationen übernommen werden. Die Definition der obigen Properties explizit als Relationen zwischen den entsprechenden Klassen hätte die Behandlung der Klassen als Individuen zur Folge, was wiederum einen Umsprung nach OWL Full verursachen würde. Um dieses Szenario zu vermeiden, sind die Relationen wie folgt modelliert:

Soll eine Klasse A mit einer Klasse B mittels der Property R verknüpft werden, so wird für die Klasse A eine AllValues-Restriction (Konstrukt `<owl:allValuesFrom>`) definiert, die für den Wertebereich der Property R alle Individuen der Klasse B festlegt, sofern der Definitionsbereich aus Klasse A besteht.

#### Beispiel:

Der Terminus *einfacher Link* gehört zum TermSet *einfacher Link*. Daher wird in der Klasse `TermConcept_einfacher_Link` die Restriction `isMemberOf only TermSet_einfacher_Link` definiert.

#### OWL-Code:

```
<owl:Class rdf:ID="TermConcept_einfacher_Link">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom>
        <owl:Class rdf:ID="TermSet_einfacher_Link"/>
      </owl:allValuesFrom>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#isMemberOf"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

### 1.4.2 Subclass-Superclass-Relationen zwischen Termini

Ist zwischen zwei Termini desselben Systems eine Ober- und Unterklassen-Beziehung feststellbar, so ist der untergeordnete Terminus als Subklasse (Konstrukt `<rdfs:subClassOf>`) des anderen formuliert.

#### Beispiel:

`TermConcept_einfacher_Link` ist eine Subklasse von `TermConcept_Link`, da das Konzept des *einfachen Links* eine Spezialisierung des Konzepts *Link* darstellt.

#### OWL-Code:

```
<owl:Class rdf:ID="TermConcept_einfacher_Link">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#TermConcept_Link"/>
  </rdfs:subClassOf>
</owl:Class>
```

### 1.4.3 Lexikalische Relationen

Die beiden in TermNet modellierten lexikalischen Relationen sind die Abkürzungsrelation `isAbbreviationOf` und `isExpansionOf`. Diese zueinander inversen ObjectProperties werden zwischen zwei Subklassen von `TermConcept` angelegt, wenn der eine Terminus eine Abkürzung des zweiten Terms darstellt.

#### Beispiel:

Der Begriff *Web* ist eine Abkürzung des Ausdrucks *World Wide Web* – somit sind die Klassen `TermConcept_web` und `TermConcept_world_wide_web` mit der Property `isAbbreviationOf` verknüpft.

#### OWL-Code:

```
<owl:Class rdf:ID="TermConcept_web">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom>
        <owl:Class rdf:ID="TermConcept_world_wide_web"/>
      </owl:allValuesFrom>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#isAbbreviationOf"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

### 1.4.4 Konzeptuelle Relationen

Im TermNet sind die konzeptuellen Relationen Hyperonymie, Hyponymie, Meronymie und Holonymie modelliert. Hyperonymie und Hyponymie werden hierbei als zueinander invers angesehen, während dies bei Meronymie und Holonymie nicht der Fall ist. Konzeptuelle Relationen können nur zwischen je zwei TermSets bestehen.

Da die Hyperonymie- und Hyponymie-Beziehungen zwischen TermSets nicht immer identisch mit einer Subclass-Superclass-Beziehung sind, wird auf letztere Variante verzichtet. Stattdessen werden zwei ObjectProperties eingeführt, die solche Ober- und Unterbegriff-Relationen formulieren. Ist ein TermSet ein Oberbegriff eines anderen TermSets, so sind die beiden Klassen mit den zueinander inversen ObjectProperties `isHyponymOf` und `isHypernymOf` verbunden. Diese beiden Properties sind zudem als transitiv deklariert.

#### Beispiel:

Ein Inhaltslink ist eine Spezialisierung des Begriffs *Link* – somit sind die Klassen `TermSet_Inhaltslink` und `TermSet_Link` mittels der Property `isHyponymOf` verbunden.

#### OWL-Code:

```
<owl:Class rdf:ID="TermSet_Inhaltslink">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom>
        <owl:Class rdf:about="#TermSet_Link"/>
      </owl:allValuesFrom>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#isHyponymOf"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

Ist ein TermSet ein Teil eines anderen TermSets, so sind die beiden entsprechenden Klassen mit den zueinander inversen ObjectProperties `isHolonymOf` und `isMeronymOf` verbunden. Im Princeton WordNet<sup>2</sup> sind diese Properties invers zueinander deklariert, in GermaNet<sup>3</sup> jedoch nicht. Letzteres erscheint

<sup>2</sup> <http://wordnet.princeton.edu/>

<sup>3</sup> <http://www.sfs.uni-tuebingen.de/1sd/>

sinnvoll, wenn man sich folgendes Beispiel vor Augen führt: Eine Erbse ist sicherlich ein Teil einer Erbsensuppe, jedoch stellt eine Erbsensuppe kaum das Ganze zu einer einzelnen Erbse dar. Daher sind auch die Holonym-Meronym-Relationen im TermNet nicht zueinander invers modelliert.

**Beispiel:**

Eine Hypertextbasis ist ein Teil eines Hypertextsystems – somit sind die Klassen TermSet\_Hypertextbasis und TermSet\_Hypertextsystem mit der Property isMeronymOf verknüpft.

**OWL-Code:**

```
<owl:Class rdf:ID="TermSet_Hypertextbasis">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom>
        <owl:Class rdf:about="#TermSet_Hypertextsystem"/>
      </owl:allValuesFrom>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#isMeronymOf"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  ..
</owl:Class>
```

### 1.4.5 Member-Relationen

Die Member-Relation zwischen einem Term und einem TermSet ist durch die zueinander inversen ObjectProperties hasMember und isMemberOf realisiert. Wird ein TermSet durch einen bestimmten Terminus lexikalisiert, so ist die Klasse des TermSet durch die Property hasMember mit der Klasse des entsprechenden Terminus verbunden. Umgekehrt wird außerdem der Klasse des Terminus die Klasse des TermSets mittels der Property isMemberOf zugewiesen.

**Beispiel:**

Das TermSet *Link* wird u. a. durch den Terminus *Relation* lexikalisiert; daher sind die Klassen TermSet\_Link und TermConcept\_Relation mittels der Property hasMember verknüpft.

**OWL-Code:**

```
<owl:Class rdf:ID="TermConcept_Relation">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="isMemberOf"/>
      </owl:onProperty>
      <owl:allValuesFrom>
        <owl:Class rdf:ID="TermSet_Link"/>
      </owl:allValuesFrom>
    </owl:Restriction>
  </rdfs:subClassOf>
  ..
</owl:Class>
```

### 1.4.6 Type-Token-Relationen

Die Zuordnung der Verwendungsinstanzen der TermConcepts zu ihren graphischen Erscheinungsformen findet über die zueinander inversen ObjectProperties isTypeOf und isTokenOf statt. Jede Verwendung eines TermConcepts im Text wird mit der im konkreten Fall verwendeten Schreibweise (Instanz von TermForm) mit der Relation isTokenOf verbunden.

**Beispiel:**

Die Textstelle URI\_TermInstance\_Link\_1 verwendet die Schreibweise *Link* des TermConcepts *Link*. Die Property isTypeOf verbindet also die Instanzen *Link* der Klasse TermForm und URI\_TermInstance\_Link\_1 der Klasse TermConcept.

**OWL-Code:**

```
<TermForm rdf:ID="Link">  
  <isTypeOf>  
    <TermConcept_Link rdf:ID="URI_TermInstance_Link_1">  
      <isTokenOf rdf:resource="#Link"/>  
    </TermConcept_Link>  
  </isTypeOf>  
</TermForm>
```

**1.4.7 Visualisierung und Übersicht der Relationen**

Eine graphische Übersicht der Klassenhierarchie, der eingeführten Individuen und den dazwischen definierten Relationen bietet Abbildung 1. Die konzeptuellen und lexikalischen Relationen sind in dieser Version der Grafik noch nicht berücksichtigt.

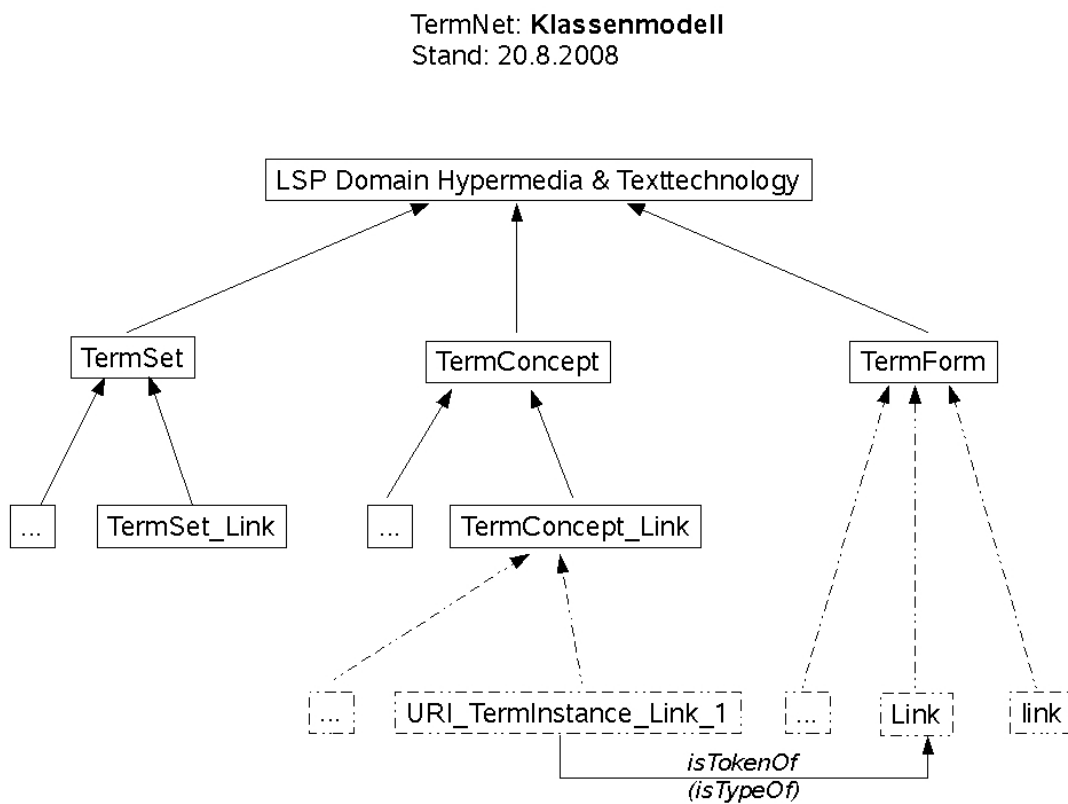


Abbildung 1: Visualisierung des Klassenmodells

Die im Klassenmodell des TermNet verwendeten Relationen werden zusammen mit ihren jeweiligen Definitionsbereichen (*domains*) und Wertebereichen (*ranges*) sowie eventuellen charakteristischen Eigenschaften in der folgenden Tabelle aufgelistet.

Property	Domain	Range	Charakteristika	inverse Property	Anzahl
<i>Lexikalische Relationen</i>					
isAbbreviationOf	TermConcept	TermConcept		isExpansionOf	27
isExpansionOf	TermConcept	TermConcept		isAbbreviationOf	26
<i>Konzeptuelle Relationen</i>					
isHypernymOf	TermSet	TermSet	transitiv	isHyponymOf	44
isHyponymOf	TermSet	TermSet	transitiv	isHypernymOf	138
isMeronymOf	TermSet	TermSet			33
isHolonymOf	TermSet	TermSet			19
<i>Member-Relationen</i>					
hasMember	TermSet	TermConcept		isMemberOf	206
isMemberOf	TermConcept	TermSet		hasMember	423
<i>Type-Token-Relationen</i>					
isTypeOf	TermForm	URI_TermInstance		isTokenOf	471
isTokenOf	URI_TermInstance	TermForm		isTypeOf	471

## 2 Instanzenmodell

Die Modellierung des Instanzenmodells orientiert sich, soweit möglich, am Aufbau des Klassenmodells. Sofern Teile der Modellierung übernommen werden können, wird in den folgenden Kapiteln auf den entsprechenden Abschnitt in Kap. 1 verwiesen.

### 2.1 Modellierungsidee

Während im Klassenmodell die TermSets und Termini als Klassen verstanden wurden, werden diese im Instanzenmodell als Individuen zweier Klassen TermSet und TermConcept modelliert. Die Termverwendungsinstanzen, die im Klassenmodell als Instanzen der Klasse TermConcept realisiert wurden (vgl. Kap. 1.3.2), werden nun der neu eingeführten Klasse TermOccurrence zugeordnet, da eine Instanziierung von Instanzen in OWL DL nicht möglich ist.

Weiterhin kann die Disjunktivität zwischen zwei Termini nicht mehr über `<owl:disjointwith>` modelliert werden, da dies nur zwischen Klassen möglich ist. Stattdessen werden nun disjunkte Termini über die ObjectProperty `isDisjointwith` miteinander verbunden.

Die Klassenhierarchie, die im Klassenmodell zwischen Termini mittels `<rdfs:subClassOf>` erstellt wurde, wird im Instanzenmodell durch die Einführung zweier ObjectProperties `isBroaderTermOf` und `isNarrowerTermOf` simuliert (vgl. Kap. 2.4.3).

Analog zum Klassenmodell wird auch das Instanzenmodell in der OWL-Sprache OWL DL modelliert.

### 2.2 Klassen

Wie schon im Klassenmodell bildet auch im Instanzenmodell die Klasse `LSP_Domain_Hypermedia_and_Texttechnology` die Oberklasse aller Konzepte. Sie besitzt die Subklassen `TermSet`, `TermConcept`, `TermForm` und `TermOccurrence`, wobei die ersten drei genannten Klassen dieselben Konzepte repräsentieren wie die entsprechenden Klassen im Klassenmodell.

Die Klasse `TermOccurrence` beinhaltet alle Verwendungsinstanzen der in der Klasse `TermConcept` modellierten Termini. Die zum Klassenmodell analoge Modellierung (Verwendungsinstanzen sind Individuen von `TermConcept`) konnte hier nicht beibehalten werden (vgl. Kap. 2.1).

## 2.3 Instanzen

Die folgenden vier Unterkapitel beschreiben die verschiedenen Typen der Individuen, die bei der Modellierung des Instanzenmodells eingeführt wurden.

### 2.3.1 Instanzen der Klasse TermSet

Jedes konkrete TermSet ist als Instanz der Klasse TermSet modelliert. Hieraus ergibt sich eine Gesamtzahl von 206 Individuen dieser Klasse.

**Beispiel:**

TermSet\_Link und TermSet\_einfacher\_Link sind konkrete TermSets und somit Instanzen der Klasse TermSet.

**OWL-Code:**

```
<TermSet rdf:ID="TermSet_Link">
<TermSet rdf:ID="TermSet_einfacher_Link">
```

### 2.3.2 Instanzen der Klasse TermConcept

Die einzelnen Termini des TermNets werden durch Instanzen der Klasse TermConcept dargestellt. Die 423 Subklassen von TermConcept im Klassenmodell werden hier also durch genauso viele Instanzen abgebildet.

**Beispiel:**

TermConcept\_Link und TermConcept\_Hyperlink sind zwei der 423 Termini und somit zwei Instanzen der Klasse TermConcept.

**OWL-Code:**

```
<TermConcept rdf:ID="TermConcept_Link">
<TermConcept rdf:ID="TermConcept_Hyperlink">
```

### 2.3.3 Instanzen der Klasse TermForm

Die Klasse TermForm beinhaltet alle 471 orthographischen Varianten der 423 Termini. Für jede Schreibweise wird ein eigenes Individuum von TermForm angelegt, dessen Bezeichnung nur aus dem repräsentierten Begriff selbst besteht. Diese Modellierung ist identisch mit der des Klassenmodells (vgl. Kap. 1.3.3).

**Beispiel:**

*Link* und *link* sind zwei Schreibweisen des Konzepts *Link*; somit sind Link und link zwei der Instanzen der Klasse TermForm.

**OWL-Code:**

```
<TermForm rdf:ID="Link">
<TermForm rdf:ID="link">
```

### 2.3.4 Instanzen der Klasse TermOccurrence

In der Klasse TermOccurrence werden alle Verwendungen der in TermConcept modellierten Termini gespeichert. Ein Individuum dieser Klasse repräsentiert hierbei eine URI, die auf die jeweilige Verwendung des Terminus im Text hinweist (vgl. Kap. 1.3.2).

**Beispiel:**

Eine Verwendungsinstanz des Terms *Link* wird als Individuum URI\_TermInstance\_Link\_1 der Klasse TermOccurrence abgebildet.

**OWL-Code:**

```
<TermOccurrence rdf:ID="URI_TermInstance_Link_1">
```

## 2.4 Properties

Die im Instanzenmodell verwendeten Relationen bestehen aus sämtlichen Relationen des Klassenmodells (vgl. Kap. 1.4). Da die Modellierung der lexikalischen, konzeptuellen und Member-Relationen nahezu unverändert hieraus übernommen wurden, werden diese in Kap. 2.4.1 nur kurz umrissen; zur näheren Erläuterung sei in die korrespondierenden Kapitel der Beschreibung des Klassenmodells verwiesen.

Die Relationen des Instanzenmodells, die zusätzlich zur Modellierung nötig waren und im Klassenmodell nicht enthalten sind bzw. verändert werden mussten, werden anschließend in den Kapiteln 2.4.4 bis 2.4.6 beschrieben.

### 2.4.1 Relationen analog zum Klassenmodell

Die konzeptuellen Relationen `isHyponymOf`, `isHypernymOf`, `isHolonymOf` und `isMeronymOf` wurden ohne Veränderung aus dem Klassenmodell übernommen (vgl. Kap. 1.4.4). Da die Zuweisung der Properties im Instanzenmodell nicht zwischen Klassen, sondern zwischen Instanzen vorgenommen wird, konnte hierbei auf die Modellierung durch Restrictions (vgl. Kap. Fehler: Referenz nicht gefunden) verzichtet werden.

Ebenso wurden die lexikalischen Relationen `isAbbreviationOf` und `isExpansionOf` (vgl. Kap. 1.4.3) sowie die Member-Relationen `hasMember` und `isMemberOf` (vgl. Kap. 1.4.5) unverändert aus dem Klassenmodell des TermNet übernommen.

### 2.4.2 isDisjointWith-Relation

Die Notwendigkeit dieser Relation ergibt sich aus der Struktur des Instanzenmodells (vgl. Kap. 2.1). Zwei Termini, die sich inhaltlich gegenseitig ausschließen, werden über diese ObjectProperty zueinander in Beziehung gesetzt.

#### Beispiel:

Ein Link kann entweder vom Typ 1:1-Link, 1:n-Link oder n:m-Link sein; eine gleichzeitige Zugehörigkeit zu zwei oder gar drei dieser Typen ist nicht möglich. `TermConcept_1-zu-1-Link`, `TermConcept_1-zu-n-Link` und `TermConcept_n-zu-m-Link` sind somit wechselseitig disjunkt.

#### OWL-Code:

```
<TermConcept rdf:ID="TermConcept_n-zu-m-Link">
  ..
  <isDisjointwith>
    <TermConcept rdf:ID="TermConcept_1-zu-n-Link">
      ...
      <isDisjointwith rdf:resource="#TermConcept_n-zu-m-Link"/>
      <isDisjointwith>
        <TermConcept rdf:ID="TermConcept_1-zu-1-Link">
          ..
          <isDisjointwith rdf:resource="#TermConcept_n-zu-m-Link"/>
          <isDisjointwith rdf:resource="#TermConcept_1-zu-n-Link"/>
        </TermConcept>
      </isDisjointwith>
    </TermConcept>
  </isDisjointwith>
  <isDisjointwith rdf:resource="#TermConcept_1-zu-1-Link"/>
</TermConcept>
```

### 2.4.3 isBroaderTerm-isNarrowerTerm-Relationen

Die Klassenhierarchie, die im Klassenmodell durch die implizite Subklassen-Beziehung zwischen Termini erstellt wurde, muss im Instanzenmodell durch ObjectProperties simuliert werden, da eine analoge Modellierung zwischen Instanzen in OWL DL nicht durchführbar ist (vgl. Kap. 2.1). Ist ein Terminus eine Spezialisierung eines anderen Terms, so wird dieser als „narrowerTerm“ des anderen verstanden. Die zu `isNarrowerTermOf` inverse Property `isBroaderTermOf` verbindet analog dazu einen allgemeineren Terminus mit seinen spezialisierten Termini.

**Beispiel:**

TermConcept\_OWL-Standard ist eine Spezialisierung des Terms TermConcept\_Standard. Diese beiden Instanzen von TermConcept werden also mittels der Property isNarrowerTermOf verbunden.

**OWL-Code:**

```
<TermConcept rdf:ID="TermConcept_OWL-Standard">
  <isNarrowerTermOf rdf:resource="#TermConcept_Standard"/>
  ...
</TermConcept>
```

**2.4.4 Type-Token-Relationen**

Während die Type-Token-Relationen isTypeOf und isTokenOf im Klassenmodell zwischen Individuen der Klassen TermConcept und TermForm stattfinden, ist dies im Instanzenmodell zwischen Instanzen von TermForm und TermOccurrence der Fall. Dieser Unterschied wird dadurch bedingt, dass die Termverwendungsinstanzen im Instanzenmodell in der Klasse TermOccurrence enthalten sind und nicht, wie im Klassenmodell, zur Klasse TermConcept gehören.

**Beispiel:**

In der Termverwendungsinstanz URI\_TermInstance\_Link\_1 wird die Schreibweise *Link* verwendet; *Link* ist wiederum eine Instanz der Klasse TermForm. URI\_TermInstance\_Link\_1 und *Link* sind also durch die Property isTokenOf verbunden.

**OWL-Code:**

```
<TermOccurrence rdf:ID="URI_TermInstance_Link_1">
  <isTokenOf>
    <TermForm rdf:ID="Link">
      <isTypeOf rdf:resource="#URI_TermInstance_Link_1"/>
      ...
    </TermForm>
  </isTokenOf>
</TermOccurrence>
```

**2.4.5 Lexikalisierungsrelationen**

Konzepte, die durch eine bestimmte Schreibweise realisiert werden, werden durch die ObjectProperty isLexicalizedAs mit der entsprechenden Instanz von TermForm verbunden. In der umgekehrten Richtung setzt die dazu inverse Property lexicalizes jede konkrete orthographische Variante mit den entsprechenden Konzepten, d. h. Instanzen der Klasse TermConcept, in Beziehung.

**Beispiel:**

Das Konzept *Link* kann durch die orthographischen Varianten *Link* und *link* realisiert werden. Die Instanz TermConcept\_Link der Klasse TermConcept wird also durch die Property isLexicalizedAs mit den TermForm-Instanzen *Link* und *link* verknüpft.

**OWL-Code:**

```
<TermConcept rdf:ID="TermConcept_Link">
  ...
  <isLexicalizedAs>
    <TermForm rdf:ID="link">
      <lexicalizes rdf:resource="#TermConcept_Link"/>
      ...
    </TermForm>
  </isLexicalizedAs>
</TermConcept>
```

**2.4.6 Verwendungsrelationen**

Die Verbindung von Termini und ihren Verwendungsinstanzen, die im Klassenmodell durch das Anlegen einer Termverwendungsinstanz in der entsprechenden Term-Klasse realisiert wurde (vgl. Kap. 1.3.2), muss im Instanzenmodell etwas anders gelöst werden, da in OWL DL Instanzen nicht noch einmal instanziiert werden können.

Stattdessen werden hierfür die zueinander inversen ObjectProperties occursIn und isOccurrenceOf definiert, die je einen Term und eine Verwendungsinstanz im Text miteinander verbinden, wenn der entsprechende Term in dieser Textstelle auftritt.

**Beispiel:**

Die Verwendungsinstanz URI\_TermInstance\_Link\_1 ist ein konkretes Vorkommen des Konzepts *Link*. Deshalb wird die TermOccurrence-Instanz URI\_TermInstance\_Link\_1 mit der TermConcept-Instanz TermConcept\_Link mittels der Property isOccurrenceOf verbunden.

**OWL-Code:**

```
<TermOccurrence rdf:ID="URI_TermInstance_Link_1">
  <isOccurrenceOf rdf:resource="#TermConcept_Link"/>
</TermOccurrence>
```

**2.4.7 Visualisierung und Übersicht der Relationen**

Analog zur Visualisierung des Klassenmodells (vgl. Abbildung 1) zeigt Abbildung 2 eine Übersicht der im Instanzenmodell verwendeten Klassen und Instanzen. Von den verwendeten Properties werden in der Abbildung lediglich die Type-Token-, Lexikalisierungs- und Verwendungs-Relationen berücksichtigt.

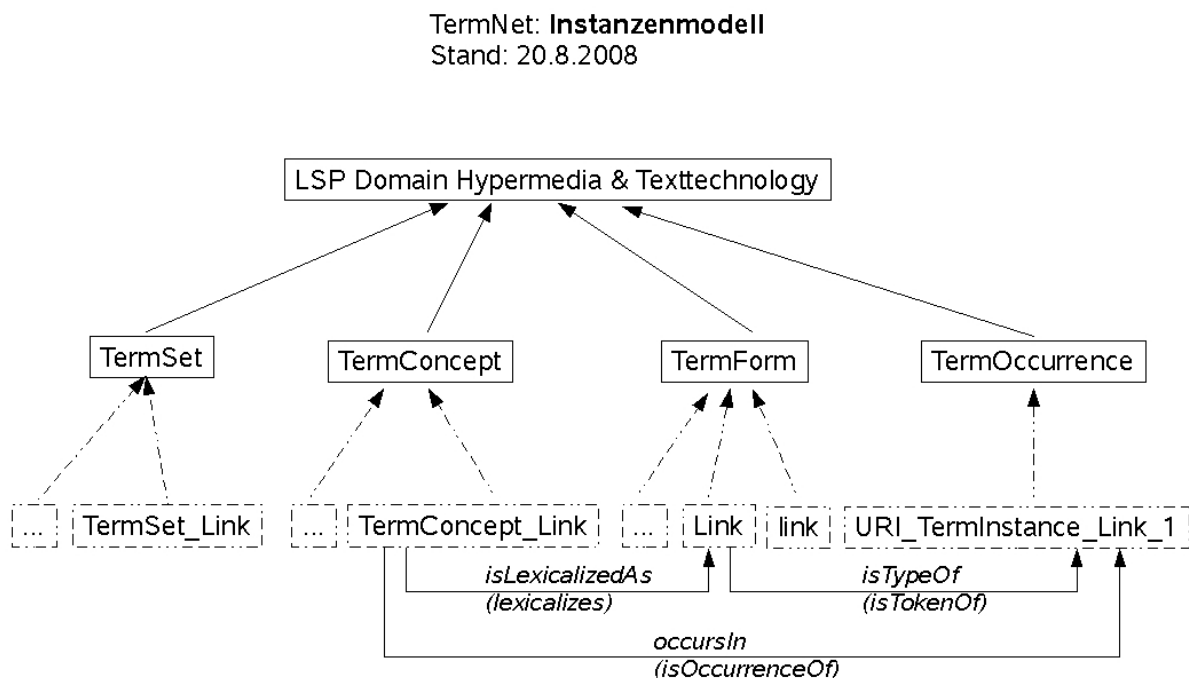


Abbildung 2: Visualisierung des Instanzenmodells

Die folgende Tabelle listet alle Relationen des Instanzenmodells mitsamt ihren Definitions- und Wertebereichen auf; zusätzlich werden eventuell vorhandene inverse Relationen sowie die Anzahl der Anlegungen solcher Relationen zwischen je zwei Individuen genannt.

Property	Domain	Range	Charakteristika	inverse Property	Anzahl
<i>Lexikalische Relationen</i>					
isAbbreviationOf	TermConcept	TermConcept		isExpansionOf	26
isExpansionOf	TermConcept	TermConcept		isAbbreviationOf	26
<i>Konzeptuelle Relationen</i>					
isHypernymOf	TermSet	TermSet	transitiv	isHyponymOf	143
isHyponymOf	TermSet	TermSet	transitiv	isHypernymOf	143
isMeronymOf	TermSet	TermSet			36
isHolonymOf	TermSet	TermSet			36
<i>Member-Relationen</i>					
hasMember	TermSet	TermConcept		isMemberOf	429
isMemberOf	TermConcept	TermSet		hasMember	429
<i>Type-Token-Relationen</i>					
isTypeOf	TermForm	TermOccurrence		isTokenOf	471
isTokenOf	TermOccurrence	TermForm		isTypeOf	471
<i>Lexikalisierungsrelationen</i>					
lexicalizes	TermForm	TermConcept		isLexicalizedAs	471
isLexicalizedAs	TermConcept	TermForm		lexicalizes	471
<i>Verwendungsrelationen</i>					
occursIn	TermConcept	TermOccurrence		isOccurrenceOf	471
isOccurrenceOf	TermOccurrence	TermConcept		occursIn	471

### 3 OWL Full-Modell

Das OWL Full-Modell des TermNet wird aus dem Instanzenmodell erzeugt, indem die Datei in Protégé geladen wird und in den Einstellungen die verwendete OWL-Subsprache auf „OWL Full“ gestellt wird. Anschließend können alle Klassen zu Metaklassen umgewandelt werden, indem im Klasseneditor unter „Asserted Conditions“ die Klasse `owl:Class` als „necessary“ hinzugefügt wird.

#### 3.1 Klassen

Die Klassenhierarchie des OWL Full-Modells entspricht daher zunächst der des Instanzenmodells. Um die erweiterten Möglichkeiten von OWL Full auszuschöpfen, wurden nachträglich die einzelnen TermConcept-Klassen als Subklassen von TermOccurrence eingeführt. Jede dieser Subklassen enthält die entsprechenden URI-Terminstanzen als Individuen, wie es auch im Klassenmodell der Fall ist (vgl. Kap. 1.3.2).

Letztlich wurde die Klassenhierarchie zwischen den TermConcepts wieder eingeführt, so dass die endgültige Klassenstruktur wieder mit der des Klassenmodells übereinstimmt (vgl. Kap. 1.2.1).

#### 3.2 Instanzen

Die Instanzen im OWL Full-Modell entsprechen denen des Instanzenmodells (vgl. Kap. 2.3), da an diesen bei der Überführung nach OWL Full keine Veränderungen vorgenommen wurden.

### 3.3 Properties

Die aus dem Instanzenmodell übernommenen Properties `lexicalizes` und `isLexicalizedAs` wurden aus dem OWL Full-Modell entfernt, da die Lexikalisierungsrelationen implizit durch die Instanziierung der `TermConcept`-Klassen mit den URI-Terminstanzen realisiert wurden (vgl. Kap. 3.1).

Weiterhin wurden die Properties `isBroaderTermOf` und `isNarrowerTermOf` entfernt, da die Hierarchie zwischen den `TermConcepts` wie im Klassenmodell durch die `subClass`-Relation wiederhergestellt wurde (vgl. Kap. 3.1). Auf die im Instanzenmodell eingeführte Relation `isDisjointwith` konnte ebenfalls verzichtet werden, da die Disjunktivität zwischen `TermConcepts` wie im Klassenmodell durch das OWL-Konstrukt `<owl:disjointwith>` wiedereingeführt wurde (vgl. Kap. 1.2.2).

## B) GermaTermNet

Eine der Zielsetzungen der Modellierung besteht darin, einen Entwurf des `TermNet` vorzulegen, der sich mit relativ geringem Aufwand an allgemeinsprachliche Wortnetze wie `WordNet` oder `GermaNet` anschließen lässt. Um die in diesem Projekt erarbeitete Modellierung bezüglich dieser Verknüpfung zu testen, sind die Definition neuer Relationen sowie weitere Modifikationen der Modellierung nötig.

## 4 Hybridmodell

Das Hybridmodell des `GermaTermModell` vereint das Klassenmodell des `TermNet` (vgl. Kap. 1) mit dem Instanzenmodell von `GermaNet` (vgl. Kunze et al. 2007).

### 4.1 Vereinigung der beiden Ressourcen

Um die beiden Ontologien `TermNet` (TN) und `GermaNet` (GN) zusammenzuführen, ist deren Integration in eine gemeinsame Ontologie `GermaTermNet` (GTN) nötig. OWL bietet eine `Import`-Funktion, die den Vorteil hat, dass Änderungen an den separaten Dateien automatisch in die gemeinsame Datei, die die anderen Ontologien importiert, übernommen werden.

Falls in den importierten Ontologien gleichlautende Klassen- oder Relationennamen vorkommen, ist es unumgänglich, diese Klassen und Relationen mit dem Präfix eines Namespaces zu versehen. So wird z. B. aus einer Klasse `TermSet_Link` die Klasse `tn:TermSet_Link`, wobei `tn` das Namespace-Präfix für die Ontologie `TermNet` bezeichnet. Dieses Präfix ist frei wählbar und wird im OWL-Editor „Protégé“<sup>4</sup> (für das Projekt wurde Protégé 3.4 beta verwendet) beim Import einer Ontologie festgelegt. Die einzige Voraussetzung bei der Festlegung des Präfixes ist die Einzigartigkeit innerhalb der Ontologie und deren importierten Ontologien.

Neben dem Präfix des Namespaces muss in „Protégé“ beim Import einer Ontologie in den Metadaten auch der URI festgelegt werden. Dieser steht per default auf `http://www.owl-ontologies.com/unnamed.owl#` und muss auf `http://www.owl-ontologies.com/<Name>.owl#` (`<Name>` wird hierbei durch den Namen der Ontologie ersetzt) geändert werden, sobald mehr als eine Ontologie importiert werden soll – andernfalls überschreibt die zweite Ontologie die Daten der zuvor importierten Datei.

### 4.2 Plugin-Relationen

Wie in Kunze et al. 2007 beschrieben sind für die Integration von TN und GN weitere Relationen (sog. Plugin-Relationen) nötig, welche die Klassen aus `TermNet` mit den entsprechenden Individuen aus `GermaNet` zueinander in Beziehung setzen. Hierfür wurden drei `ObjectProperties` (Konstrukt: `<owl:ObjectProperty>`) definiert, die in den folgenden Unterkapiteln beschrieben sind. Jede der drei Plugin-Relationen besitzt außerdem eine inverse `ObjectProperty`, die sich im Namen durch den Zusatz `inverseOf` vor dem eigentlichen Namen der Relationen abgrenzt.

<sup>4</sup> <http://protege.stanford.edu/>

Die Plugin-Relationen sind im Hybridmodell des GermaTermNet jeweils als `hasValue`-Restrictions modelliert (vgl. Kap. Fehler: Referenz nicht gefunden), da sie von Klassen ausgehend zu Individuen verweisen und das Anlegen von Properties zwischen Klassen in OWL DL nur auf diese Weise möglich ist.

Da die inverse Verbindung zwischen Individuum und Klasse nicht möglich ist, ohne nach OWL Full zu springen (vgl. Lungen/Storrer 2006), bestehen diese Restrictions nur in der Richtung von TN-Termini-Klassen nach GN-Synset-Individuen.

#### 4.2.1 attachedToNearSynonym

Entspricht ein Terminus aus TN einem Synset aus GN, so wird die `TermConcept`-Klasse mit dem Synset-Individuum mittels der ObjectProperty `attachedToNearSynonym` verbunden.

##### Beispiel:

Der TN-Terminus *Verweis* entspricht im GermaNet dem Synset *Link*. Somit sind die Klasse `tn:TermConcept_Verweis` und das Individuum `gn:Artefakt.5373`, welches das Synset beschreibt, das die Lexical Unit *Link* enthält, durch die Property `attachedToNearSynonym` verknüpft.

##### OWL-Code:

```
<rdf:Description rdf:about="http://www.owl-ontologies.com/TermNet.owl#
  TermConcept_Verweis">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:hasValue rdf:resource="http://www.owl-ontologies.com/GermaNet-
        instances.owl#nArtefakt.5373"/>
      <owl:onProperty rdf:resource="#attachedToNearSynonym"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</rdf:Description>
```

#### 4.2.2 attachedToGeneralConcept

Die ObjectProperty `attachedToGeneralConcept` verbindet alle TN-TermConcepts *A* mit einem GN-Synset *E*, wenn die folgenden Bedingungen erfüllt sind:

- `TermConcept A` gehört zum `TermSet B`, welches ein Hyponym des `TermSets C` ist.
- `TermSet C` beinhaltet das `TermConcept D`, das mit dem GN-Synset *E* in einer `attachedToNearSynonym`-Relation steht.

##### Beispiel:

Das TN-TermSet `tn:TermSet_unidirektionaler_Link (B)` ist ein Hyponym der Klasse `tn:TermSet_Link (C)`, deren Member `tn:TermConcept_Link (D)` wiederum mit dem GN-Individuum `gn:Artefakt.5373 (E)` (Synset, das die Lexical Unit *Link* beinhaltet) mittels der Property `attachedToNearSynonym` verbunden ist. Die Klasse `tn:TermConcept_unidirektionaler_Link (A)` gehört wiederum zum `TermSet tn:TermSet_unidirektionaler_Link (B)`, weshalb sie mit `gn:Artefakt.5373 (E)` mittels `attachedToGeneralConcept` verknüpft ist.

##### OWL-Code:

```
<rdf:Description rdf:about="http://www.owl-ontologies.com/TermNet.owl#
  TermConcept_unidirektionaler_Link">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#attachedToGeneralConcept"/>
      <owl:hasValue rdf:resource="http://www.owl-ontologies.com/GermaNet-
        instances.owl#nArtefakt.5373"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</rdf:Description>
```

### 4.2.3 attachedToHolonym

Die ObjectProperty `attachedToHolonym` verbindet alle TN-TermConcepts *A* mit einem GN-Synset *E*, wenn die folgenden Bedingungen erfüllt sind:

- TermConcept *A* gehört zum TermSet *B*, welches das TermSet *C* als Holonym besitzt.
- TermSet *C* beinhaltet das TermConcept *D*, das mit dem GN-Synset *E* in einer `attachedToNearSynonym`-Relation steht.

#### Beispiel:

Zum TN-TermSet `tn:TermSet_Anker` (*B*) gehört der Term `tn:TermConcept_Anker` (*A*), und `tn:TermSet_Anker` (*B*) besitzt die Klasse `tn:TermSet_Link` (*C*) als Holonym. Deren Member `tn:TermConcept_Link` (*D*) ist wiederum an das GN-Individuum `gn:Artefakt.5373` (*E*) (Synset mit der Lexical Unit *Link*) durch die `attachedToNearSynonym`-Relation gebunden. Somit sind auch `tn:TermConcept_Anker` (*A*) und `gn:Artefakt.5373` (*E*) durch die Property `attachedToHolonym` verknüpft.

#### OWL-Code:

```
<rdf:Description rdf:about="http://www.owl-ontologies.com/TermNet.owl#
  TermConcept_Anker">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:hasValue rdf:resource="http://www.owl-ontologies.com/GermaNet-
        instances.owl#nArtefakt.5373"/>
      <owl:onProperty rdf:resource="#attachedToHolonym"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</rdf:Description>
```

### 4.2.4 Übersicht der Plugin-Relationen

Die modellierten Plugin-Relationen werden der Übersichtlichkeit halber zusammen mit der Häufigkeit ihrer Vorkommen in Restrictions in der nachfolgenden Tabelle zusammengefasst.

Property	Domain	Range	Anzahl	Inverse
<code>attachedToNearSynonym</code>	<code>tn_TermConcept</code>	<code>gn_Synset</code>	27	<code>inverseOfAttachedToNearSynonym</code>
<code>attachedToGeneralConcept</code>	<code>tn_TermConcept</code>	<code>gn_Synset</code>	103	<code>inverseOfAttachedToGeneralConcept</code>
<code>attachedToHolonym</code>	<code>tn_TermConcept</code>	<code>gn_Synset</code>	20	<code>inverseOfAttachedToHolonym</code>

## 5 Klassenmodell

Das GermaTermNet-Klassenmodell vereint das Klassenmodell des TermNet (vgl. Kap. 1) mit dem Klassenmodell des GermaNet. Hier sind, genau wie im Hybridmodell des GermaTermNet (vgl. Kap. 4), die drei Plugin-Relationen `attachedToSynonym`, `attachedToGeneralConcept` und `attachedToHolonym` als Restrictions modelliert; da diese Relationen jedoch nicht zwischen Klasse und Individuum, sondern zwischen Klasse und Klasse angelegt wurden, wurde hierfür jeweils wie im Klassenmodell des TermNet (vgl. Kap. Fehler: Referenz nicht gefunden) die `allValuesFrom`-Restriction verwendet.

Im Gegensatz zum Hybridmodell konnten beim Klassenmodell die Plugin-Relationen in beiden Richtungen angelegt werden, d. h. auch die inversen Relationen `inverseOfAttachedToNearSynonym`, `inverseOfAttachedToGeneralConcept` und `inverseOfattachedToHolonym` wurden als Restrictions vom Typ `allValuesFrom` erstellt.

### 5.1 attachedToNearSynonym

Entspricht ein Terminus aus TN einem Synset aus GN, so wird die Klasse des Terms mit der Klasse des Synsets mittels der ObjectProperty `attachedToNearSynonym` verbunden.

**Beispiel:**

Der TN-Terminus *Verweis* entspricht im GermaNet dem Synset *Link*. Somit sind die Klasse `tn:TermConcept_Verweis` und die Synset-Klasse `gn:Artefakt.5373`, welche die Lexical Unit *Link* beinhaltet, durch die Property `attachedToNearSynonym` verknüpft.

**OWL-Code:**

```
<rdf:Description rdf:about="http://www.owl-ontologies.com/TermNet.owl#
TermConcept_Verweis">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom rdf:resource="http://www.owl-ontologies.com/GermaNet-
instances.owl#nArtefakt.5373"/>
      <owl:onProperty rdf:resource="#attachedToNearSynonym"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</rdf:Description>
```

## 5.2 attachedToGeneralConcept

Die ObjectProperty `attachedToGeneralConcept` verbindet alle TN-TermConcepts *A* mit einem GN-Synset *E*, wenn die folgenden Bedingungen erfüllt sind:

- TermConcept *A* gehört zum TermSet *B*, welches ein Hyponym des TermSets *C* ist.
- TermSet *C* beinhaltet das TermConcept *D*, das mit dem GN-Synset *E* in einer `attachedToNearSynonym`-Relation steht.

**Beispiel:**

Das TN-TermSet `tn:TermSet_unidirektionaler_Link` (*B*) ist ein Hyponym der Klasse `tn:TermSet_Link` (*C*), deren Member `tn:TermConcept_Link` (*D*) wiederum mit der GN-Klasse `gn:Artefakt.5373` (*E*) (Synset, das die Lexical Unit *Link* beinhaltet) mittels der Property `attachedToNearSynonym` verbunden ist. Die Klasse `tn:TermConcept_unidirektionaler_Link` (*A*) gehört wiederum zum TermSet `tn:TermSet_unidirektionaler_Link` (*B*), weshalb sie mit `gn:Artefakt.5373` (*E*) mittels `attachedToGeneralConcept` verknüpft ist.

**OWL-Code:**

```
<rdf:Description rdf:about="http://www.owl-ontologies.com/TermNet.owl#
TermConcept_unidirektionaler_Link">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom rdf:resource="http://www.owl-ontologies.com/GermaNet-
instances.owl#nArtefakt.5373"/>
      <owl:onProperty rdf:resource="#attachedToGeneralConcept"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</rdf:Description>
```

## 5.3 attachedToHolonym

Die ObjectProperty `attachedToHolonym` verbindet alle TN-TermConcepts *A* mit einem GN-Synset *E*, wenn die folgenden Bedingungen erfüllt sind:

- TermConcept *A* gehört zum TermSet *B*, welches das TermSet *C* als Holonym besitzt.
- TermSet *C* beinhaltet das TermConcept *D*, das mit dem GN-Synset *E* in einer `attachedToNearSynonym`-Relation steht.

**Beispiel:**

Zum TN-TermSet `tn:TermSet_Anker` (*B*) gehört der Term `tn:TermConcept_Anker` (*A*), und `tn:TermSet_Link` (*C*) ist ein Holonym von `tn:TermSet_Anker` (*B*). `tn:TermSet_Link` (*C*) besitzt wiederum das Member `tn:TermConcept_Link` (*D*), das an die GN-Klasse `gn:Artefakt.5373` (*E*) (Synset, das die Lexical Unit *Link* enthält) durch die `attachedToNearSynonym`-Relation gebunden. Somit sind auch `tn:TermConcept_Anker` (*A*) und `gn:Artefakt.5373` (*E*) durch die Property `attachedToHolonym` verknüpft.

**OWL-Code:**

```

<rdf:Description rdf:about="http://www.owl-ontologies.com/TermNet.owl#
TermConcept_Anker">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#attachedToHolonym"/>
      <owl:allValuesFrom rdf:resource="http://www.owl-ontologies.com/GermaNet-
instances.owl#nArtefakt.5373"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</rdf:Description>

```

## 6 Instanzenmodell

Das Instanzenmodell von GermaTermNet kombiniert das Instanzenmodell von TermNet (vgl. Kap. 2) mit dem Instanzenmodell von GermaNet zu einer Ressource. Da sowohl TermConcepts als auch Synsets in diesen Modellen als Individuen repräsentiert werden, konnten hier alle Plugin-Relationen direkt zwischen diesen Instanzen angelegt werden; auf die im Hybrid- (vgl. Kap. 4) und Klassenmodell (vgl. Kap. 5) verwendeten Restrictions konnte hier also verzichtet werden.

Auch beim Instanzenmodell konnten zusätzlich zu den Plugin-Relationen auch deren inverse Relationen `inverseOfAttachedToNearSynonym`, `inverseOfAttachedToGeneralConcept` und `inverseOfAttachedToHolonym` zwischen GN-Synsets und TN-TermConcepts angelegt werden.

### 6.1 attachedToNearSynonym

Entspricht ein Terminus aus TN einem Synset aus GN, so wird die Term-Instanz mit der entsprechenden Synset-Instanz mittels der ObjectProperty `attachedToNearSynonym` verbunden.

**Beispiel:**

Der TN-Terminus *Verweis* entspricht im GermaNet dem Synset *Link*. Somit sind die Klasse `tn:TermConcept_Verweis` und die Synset-Klasse `gn:Artefakt.5373`, welche die Lexical Unit *Link* beinhaltet, durch die Property `attachedToNearSynonym` verknüpft.

**OWL-Code:**

```

<rdf:Description rdf:about="http://www.owl-ontologies.com/TermNet.owl#
TermConcept_Verweis">
  <attachedToNearSynonym rdf:resource="http://www.owl-ontologies.com/GermaNet-
instances.owl#nArtefakt.5373"/>
</rdf:Description>

```

### 6.2 attachedToGeneralConcept

Die ObjectProperty `attachedToGeneralConcept` verbindet alle TN-TermConcepts *A* mit einem GN-Synset *E*, wenn die folgenden Bedingungen erfüllt sind:

- TermConcept *A* gehört zum TermSet *B*, welches ein Hyponym des TermSets *C* ist.
- TermSet *C* beinhaltet das TermConcept *D*, das mit dem GN-Synset *E* in einer `attachedToNearSynonym`-Relation steht.

**Beispiel:**

Das TN-TermSet `tn:TermSet_unidirektionaler_Link` (*B*) ist ein Hyponym der Klasse `tn:TermSet_Link` (*C*), deren Member `tn:TermConcept_Link` (*D*) wiederum mit der GN-Klasse `gn:Artefakt.5373` (*E*) (Synset, das die Lexical Unit *Link* beinhaltet) mittels der Property `attachedToNearSynonym` verbunden ist. Die Klasse `tn:TermConcept_unidirektionaler_Link` (*A*) gehört wiederum zum TermSet `tn:TermSet_unidirektionaler_Link` (*B*), weshalb sie mit `gn:Artefakt.5373` (*E*) mittels `attachedToGeneralConcept` verknüpft ist.

#### OWL-Code:

```
<rdf:Description rdf:about="http://www.owl-ontologies.com/TermNet.owl#
  TermConcept_unidirektionaler_Link">
  <attachedToGeneralConcept rdf:resource="http://www.owl-ontologies.com/
    GermaNet-instances.owl#nArtefakt.5373"/>
</rdf:Description>
```

### 6.3 attachedToHolonym

Die ObjectProperty `attachedToHolonym` verbindet alle TN-TermConcepts *A* mit einem GN-Synset *E*, wenn die folgenden Bedingungen erfüllt sind:

- TermConcept *A* gehört zum TermSet *B*, welches das TermSet *C* als Holonym besitzt.
- TermSet *C* beinhaltet das TermConcept *D*, das mit dem GN-Synset *E* in einer `attachedToNearSynonym`-Relation steht.

#### Beispiel:

Zum TN-TermSet `tn:TermSet_Anker` (*B*) gehört der Term `tn:TermConcept_Anker` (*A*), und `tn:TermSet_Link` (*C*) ist ein Holonym von `tn:TermSet_Anker` (*B*). `tn:TermSet_Link` (*C*) besitzt wiederum das Member `tn:TermConcept_Link` (*D*), das an die GN-Klasse `gn:Artefakt.5373` (*E*) (Synset, das die Lexical Unit *Link* enthält) durch die `attachedToNearSynonym`-Relation gebunden. Somit sind auch `tn:TermConcept_Anker` (*A*) und `gn:Artefakt.5373` (*E*) durch die Property `attachedToHolonym` verknüpft.

#### OWL-Code:

```
<rdf:Description rdf:about="http://www.owl-ontologies.com/TermNet.owl#
  TermConcept_Anker">
  <attachedToHolonym rdf:resource="http://www.owl-ontologies.com/GermaNet-
    instances.owl#nArtefakt.5373"/>
</rdf:Description>
```

## 7 OWL Full-Modell

Das GermaTermNet-OWL Full-Modell vereint das OWL Full-Modell des TermNet mit dem GermaNet-Metamodell. Wie auch im GTN-Instanzenmodell konnten die Plugin-Relationen `attachedToNearSynonym`, `attachedToGeneralConcept` und `attachedToHolonym` sowie deren Inverse direkt zwischen den entsprechenden Individuen angelegt werden.

Auf die genauere Beschreibung der Plugin-Relationen sowie die Code-Beispiele sei an dieser Stelle auf das Instanzenmodell des GermaTermNet verwiesen, da diese jeweils identisch sind (vgl. Kap. 6.1, 6.2 und 6.3).

## Literaturverzeichnis

Kunze, C. / Lemnitzer, L. / Längen, H. / Storrer, A. (2007): **Repräsentation und Verknüpfung allgemeinsprachlicher und terminologischer Wortnetze in OWL**. Zeitschrift für Sprachwissenschaft 26.

Längen, H. / Beißwenger, M. / Stockrahm, B. / Storrer, A. (im Erscheinen): **Modeling and Processing Wordnets in OWL**. In: Mehler, A. / Kühnberger, K.-U. / Lobin, H. / Längen, H. / Storrer, A. / Witt, A. (Eds.): Modeling, Learning and Processing of Text Technological Data Structures. Dordrecht: Springer.

Längen, H. / Storrer, A. (2006): **Domain ontologies and word nets in OWL: Modelling options**. In: Proceedings of the International Workshop "Ontologies in Text Technology" OTT'06, 28/29. September 2006, Osnabrück. pp. 3-10.