Universität Leipzig
Fakultät für Mathematik und Informatik
Institut für Informatik

Untersuchungen von Ontologiesprachen mit modelltheoretischer Semantik

Eine Einführung in die formalen Semantiken von Common Logic, RDF/RDFS und Extended RDF

Diplomarbeit

Leipzig, Juli 2009

vorgelegt von
Hummel, Axel
Studiengang Informatik

Betreuender Hochschullehrer:
Prof. Dr. Heinrich Herre
Institut für Medizinische Informatik,
Statistik und Epidemiologie (IMISE)

Kurzfassung

Wissensrepräsentation als Teilgebiet der künstlichen Intelligenz beschäftigt sich mit der formalen Beschreibung von Wissen, um eine automatisierte Verarbeitung von Informationen zu ermöglichen. Mit der Einführung des Semantic Web und der damit verbundenen webbasierten Wissensmodellierung in Form von Ontologien, hat dieses Forschungsgebiet in den letzten Jahren einen starken Aufschwung erfahren. Für die Formulierung derartiger Ontologien werden formale Sprachen verwendet. Diese Ontologiesprachen sind Gegenstand der vorliegenden Arbeit. Stellvertretend für die Vielzahl der existierenden Ontologiesprachen werden Common Logic, das Resource Description Framework (RDF), das Resource Description Framework Schema (RDFS) sowie Extended RDF (ERDF) ausführlich behandelt.

Es wird gezeigt, dass Common Logic unter Verwendung der CL-Sequenznamen unendliche Konjunktionen und Disjunktionen erlaubt und aufgrund dieser Ausdrucksstärke nicht kompakt ist. Des Weiteren wird auf die Unterschiede zwischen unsegregierten CL-Dialekten und segregierten CL-Dialekten eingegangen. Die syntaktischen Freiheiten von Common Logic sind wesentlich größer als in der klassischen Prädikatenlogik erster Stufe. Dadurch besitzt Common Logic zusätzliche Modellierungsfähigkeiten, die ebenfalls in dieser Arbeit diskutiert werden. So ist es beispielsweise möglich, die Russellsche Klasse in Common Logic zu formulieren. Die Semantik dieser Klasse ist ein weiterer Untersuchungsgegenstand dieser Arbeit.

Angesichts der Ausdrucksmächtigkeit von Common Logic, lassen sich sowohl RDF-Dokumente als auch RDFS-Dokumente nach Common Logic übertragen. Eine derartige Methode, die man als Einbettung bezeichnet, wird im Rahmen dieser Arbeit ausführlich erläutert. Weiterhin wird ERDF als Vertreter der zahlreichen Erweiterungen von RDF/RDFS vorgestellt und dessen Einbettung in Common Logic beschrieben. ERDF erlaubt die Modellierung von unvollständigen Informationen, kann jedoch keine Widersprüche darstellen. Daher wird dieses Framework zu Generalized ERDF erweitert und eine parakonsistente Semantik vorgeschlagen, die neben partiellen Informationen auch widersprüchliche Aussagen zulässt.

Danksagung

Ich möchte an erster Stelle meinem Betreuer Herrn Professor Heinrich Herre für seine Unterstützung und Hilfestellung während der Anfertigung dieser Arbeit danken. Er hatte stets ein offenes Ohr für meine Fragen und Probleme.

Weiterhin bedanke ich mich bei den Mitgliedern der Forschungsgruppe OntoMed am Institut für Medizinische Informatik, Statistik und Epidemiologie für die gute Zusammenarbeit während meines Praktikums. In besonderer Weise danke ich meinem damaligen Betreuer Frank Loebe. Er hat die erfolgreiche Absolvierung meines Praktikums ermöglicht, mein Interesse an Common Logic und anderen formalen Sprachen geweckt, mich in die Methoden und Prinzipien des Verfassens von wissenschaftlichen Texten auf Basis von LaTeX eingeführt und damit den Grundstein für diese Arbeit gelegt.

Großer Dank gebührt Herrn Gobsch für das Korrekturlesen und die Verifizierung der formalen Beweise sowie meinen Mitbewohnern, die mich stets unterstützten.

Abschließend möchte ich mich bei allen Personen bedanken, die mir während des Studiums geholfen haben. Dazu zählen in erster Linie meine Eltern, ohne deren finanzielle Zuwendungen mein Studium nicht möglich gewesen wäre.
Ein Dankeschön auch an meine Kommilitonen Mathias, Martin, Karsten, Sören, Steffen, Raik, Jörn, Enrico und Aron für die gemeinsame Zeit an der Universität.
Herrn Professor Manfred Droste und Herrn HDoz. Dr. Dietrich Kuske danke ich für ihre interessanten Vorlesungen, aufgrund derer ich mich für den Studienschwerpunkt “Theoretische Informatik” entschieden habe.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Die Forschungsdisziplin künstliche Intelligenz verfolgt das Ziel, menschliches Verhalten im Computer zu simulieren, um dadurch die automatisierte Problemlösung komplexer Sachverhalte sowie die Kommunikation zwischen Mensch und Maschine zu verbessern. Für die Repräsentation des dafür notwendigen Wissens werden in der künstlichen Intelligenz geeignete Systeme untersucht. Das Semantic Web setzt für die verteilte Organisation von Wissen sogenannte Ontologien ein, die mithilfe einer formalen Sprache spezifiziert werden. Mit der rasant wachsenden Anzahl verfügbarer Ontologien infolge des Semantic Web wurden zahlreiche neue Ontologiesprachen entwickelt. In diesem Kontext entstand das Common Logic Framework, das seit Oktober 2007 ein offizieller ISO Standard ist [48].

1.1 Motivation und Zielsetzung

Wir, die Forschungsgruppe Ontologies in Medicine and Life Sciences (Onto-Med)[1], haben großes Interesse, Common Logic für die formale Beschreibung von Ontologien zu verwenden. Um ein besseres Verständnis von Common Logic zu erhalten und die im ISO Standard angesprochenen Eigenschaften der Semantik nachvollziehen zu können, wurde diese Arbeit angefertigt. Zusätzlich soll das Verhältnis zur Prädikatenlogik erster Stufe im Rahmen der Diplomarbeit untersucht werden. Die Analyse der Beziehung zwischen Common Logic und den offiziellen Semantic Web Sprachen RDF und RDFS ist ein weiteres Ziel dieser Diplomarbeit. Darüber hinaus gehört die Untersuchung von webbasierten Ontologiesprachen, die RDF/RDFS erweitern, zu unseren Forschungsschwerpunkten. Als Vertreter dieser formalen Sprachen wird Extended RDF [5] ausgewählt und in dieser Arbeit eingehend studiert.

1.2 Gliederung der Arbeit

Die Diplomarbeit ist folgendermaßen gegliedert:
Das nachfolgende Kapitel enthält die grundlegenden Begriffe der Arbeit. Es werden die notwendigen mathematischen Fachtermini und Notationen sowie die Konzepte Ontologie und Semantic Web erläutert.

[1] http://www.onto-med.de/

Das dritte Kapitel definiert die Syntax und Semantik der Prädikatenlogik erster Stufe. Zusätzlich wird das prädikatenlogische Universum charakterisiert und die Verwendung der Prädikatenlogik zur formalen Beschreibung von Ontologien diskutiert.

Anschließend betrachten wir das Common Logic Framework. Als erstes wird die Syntax von Common Logic eingeführt und mit der prädikatenlogischen Syntax in Beziehung gesetzt. Danach führen wir die Semantik von Common Logic ein und untersuchen das CL-Universum. Im Anschluss daran werden die Eigenschaften von Common Logic analysiert. Dabei gehen wir auf die verschiedenen Dialektarten, die CL-Sequenznamen, die Endlichkeits- und Kompaktheitseigenschaft sowie die Semantik der Russellschen Klasse ein.

Im fünften Abschnitt der Diplomarbeit werden die Semantic Web Sprachen RDF und RDFS behandelt. Nachdem wir deren Syntax und Semantik definiert sowie das RDFS-Universum analysiert haben, werden wir zeigen, dass sich diese Ontologiesprachen vollständig in Common Logic einbetten lassen.

Stellvertretend für die zahlreichen Erweiterungsvorschläge von RDFS untersuchen wir danach Extended RDF. Auch hier wird zunächst die Syntax und Semantik eingeführt und das ERDF-Universum betrachtet. Obwohl diese Ontologiesprache auf Partial Logic [42] basiert und die Modellierung von unvollständigen Informationen erlaubt, ist es möglich ERDF-Dokumente nach Common Logic zu übertragen. Eine derartige Einbettung von ERDF in Common Logic ist ebenfalls Schwerpunkt dieses Kapitels.

Der siebente Abschnitt thematisiert widersprüchliche Informationen. Nachdem wir die Ursachen und mögliche Lösungsstrategien für inkonsistente Ontologien überblicksartig dargestellt haben, erweitern wir ERDF zu Generalized Extended RDF. Dabei wird eine parakonsistente Semantik vorgeschlagen, welche die formale Beschreibung von inkonsistenten Ontologien erlaubt und auf den stabil erzeugten Modellen von Heinrich Herre und Gerd Wagner [43] basiert.

In Kapitel acht werden die Hauptresultate der Diplomarbeit zusammengefasst sowie Themen für zukünftige Arbeiten vorgestellt.

2 Grundbegriffe

Dieses Kapitel erläutert grundlegende Begriffe und Notationen der Diplomarbeit. Wir beginnen mit der Definition allgemeiner mathematischer Begriffe. Danach werden die Grundbegriffe der mathematischen Logik eingeführt. Abschließend widmen wir den Schlagworten *Ontologie* und *Semantic Web* jeweils ein eigenes Unterkapitel, um die Hauptgedanken zu skizzieren, die sich hinter diesen beiden Begriffen verbergen.

2.1 Mathematische Grundbegriffe

Dieser Abschnitt definiert mathematische Begriffe, die in den nachfolgenden Kapiteln verwendet werden.

Generell wird das Zeichen □ verwendet, um das Ende von Definitionen und Beispielen anzuzeigen. Formale Beweise werden mit dem Symbol ■ abgeschlossen. Nachstehend führen wir die grundlegenden Notationen der Mengentheorie ein.

Die Elementbeziehung bezeichnen wir mit $\in$, die Mengeninklusion mit $\subseteq$, die Vereinigung zweier Mengen mit $\cup$, den Schnitt zweier Mengen mit $\cap$ und die Differenz zweier Mengen mit $\setminus$. Die Menge der natürlichen Zahlen wird durch das Symbol $\mathbb{N}$ und die leere Menge durch $\emptyset$ gekennzeichnet. Für eine Menge M sei M^n die Menge aller n-Tupel von M, wobei $n \in \mathbb{N}$ gilt. Zur Darstellung eines n-Tupels verwenden wir die Schreibweise $\langle a_1, a_2, \ldots, a_n \rangle$. Für das leere Tupel $\langle\rangle$ vereinbaren wir die Notation ε. Die Länge eines Tupels $\langle a_1, a_2, \ldots, a_n \rangle$ wird durch $|\langle a_1, a_2, \ldots, a_n \rangle|$ dargestellt. Die Menge aller Tupel von M bezeichnen wir mit M^*, also $M^* = \bigcup_{n \in \mathbb{N}} M^n$.
Nun folgen weitere Definitionen.

Definition 2.1 (Einschränkung einer Abbildung)
Es sei $f : D \to W$ eine Abbildung und $E \subseteq D$ eine Teilmenge von D.
Die Abbildung $\widehat{f} : E \to W$, die definiert ist als $\widehat{f}(x) := f(x)$ für alle $x \in E$ heißt *Einschränkung* von f auf E.

□

Die Einschränkung der Abbildung $f : D \to W$ auf die Menge $E \subseteq D$ bezeichnen wir mit $f|_E$.

Definition 2.2 (Potenzmenge, kartesisches Produkt)
Seien M und N zwei Mengen, dann bezeichnet

(a) $\mathcal{P}(M) = \{N \mid N \subseteq M\}$ die *Potenzmenge* von M und

(b) $M \times N = \{\langle m, n\rangle \mid m \in M, n \in N\}$ das *kartesische Produkt* von M und N.

□

Definition 2.3 (Halbordnung)
Sei M eine Menge. Eine zweistellige Relation R heißt *Halbordnung* auf M genau dann, wenn R die folgenden Eigenschaften erfüllt:

1. Für alle $a \in M$ gilt: $\langle a, a\rangle \in R$.
   (Reflexivität)
2. Für alle $a, b, c \in M$ gilt: wenn $\langle a, b\rangle \in R$ und $\langle b, c\rangle \in R$, so ist $\langle a, c\rangle \in R$.
   (Transitivität)
3. Für alle $a, b \in M$ gilt: wenn $\langle a, b\rangle \in R$ und $\langle b, a\rangle \in R$, so ist $a = b$.
   (Antisymmetrie)

□

Definition 2.4 (inverse Relation)
Seien M_1 und M_2 zwei Mengen. Die zweistelligen Relationen $R \subseteq M_1 \times M_2$ und $R^{-1} \subseteq M_2 \times M_1$ heißen *invers* zueinander, falls für alle $a \in M_1$ und $b \in M_2$ gilt: $\langle a, b\rangle \in R$ genau dann, wenn $\langle b, a\rangle \in R^{-1}$.

□

In der Mathematik werden Mengen von Symbolen, die entweder Relationen, Funktionen oder Konstanten repräsentieren, zusammen mit der Argumentanzahl als *Signaturen* bezeichnet.

Definition 2.5 (Signatur)
Eine *Signatur* $\Sigma = \langle \mathbb{F}, \mathbb{R}, \mathbb{K}, ar\rangle$ besteht aus:

- einer Menge $\mathbb{F}$ von *Funktionssymbolen*,
- einer Menge $\mathbb{R}$ von *Relationssymbolen*,
- einer Menge $\mathbb{K}$ von *Konstantensymbolen* und
- einer Abbildung $ar : \mathbb{F} \cup \mathbb{R} \to \mathbb{N}$.

□

Die Abbildung ar ordnet jedem Funktionssymbol und jedem Relationssymbol eine feste Stelligkeit zu und wird deshalb auch *Stelligkeitsabbildung* oder *Aritätsabbildung* genannt. Die Konstantensymbole werden als nullstellige Funktionssymbole interpretiert, d. h. $\mathbb{K} \subseteq \mathbb{F}$ und $ar(c) = 0$ für alle $c \in \mathbb{K}$.

Die Bedeutung der Symbole einer Signatur wird mithilfe von *Strukturen* festgelegt.

Definition 2.6 (Σ-Struktur)
Sei $\Sigma = \langle \mathbb{F}, \mathbb{R}, \mathbb{K}, ar \rangle$ eine Signatur.
Eine Σ-*Struktur* $\mathcal{A} = \left\langle U, \left(f^{\mathcal{A}}\right)_{f \in \mathbb{F}}, \left(R^{\mathcal{A}}\right)_{R \in \mathbb{R}}, \left(c^{\mathcal{A}}\right)_{c \in \mathbb{K}} \right\rangle$ wird festgelegt durch:

- eine *Grundmenge* U und
- eine Abbildung δ mit dem Definitionsbereich $\mathbb{F} \cup \mathbb{R} \cup \mathbb{K}$, so dass gilt:
  - Für jedes Funktionssymbol $f \in \mathbb{F}$ ist $\delta(f) = f^{\mathcal{A}}$ und $f^{\mathcal{A}} : U^{ar(f)} \to U$ ist eine $ar(f)$-stellige Funktion.
  - Für jedes Relationssymbol $R \in \mathbb{R}$ ist $\delta(R) = R^{\mathcal{A}}$ und $R^{\mathcal{A}} \subseteq U^{ar(R)}$ ist eine $ar(R)$-stellige Relation über U.
  - Für jedes Konstantensymbol $c \in \mathbb{K}$ ist $\delta(c) = c^{\mathcal{A}}$ und $c^{\mathcal{A}} \in U$.

□

Derartige Σ-Strukturen werden auch als *relationale Strukturen* bezeichnet.

Abschließend wollen wir für Σ-Strukturen die nachfolgende Schreibweise einführen.

Definition 2.7 (Identität zweier Σ-Strukturen)
Seien $\Sigma = \langle \mathbb{F}, \mathbb{R}, \mathbb{K}, ar \rangle$ eine Signatur sowie $\mathcal{A} = \left\langle U, \left(f^{\mathcal{A}}\right)_{f \in \mathbb{F}}, \left(R^{\mathcal{A}}\right)_{R \in \mathbb{R}}, \left(c^{\mathcal{A}}\right)_{c \in \mathbb{K}} \right\rangle$ und $\mathcal{B} = \left\langle \widehat{U}, \left(f^{\mathcal{B}}\right)_{f \in \mathbb{F}}, \left(R^{\mathcal{B}}\right)_{R \in \mathbb{R}}, \left(c^{\mathcal{B}}\right)_{c \in \mathbb{K}} \right\rangle$ zwei Σ-Strukturen.
Die beiden Σ-Strukturen $\mathcal{A}$ und $\mathcal{B}$ sind identisch genau dann, wenn die folgenden Bedingungen erfüllt sind:

1. $U = \widehat{U}$,
2. $f^{\mathcal{A}}\left(u_1, u_2, \ldots, u_{ar(f)}\right) = f^{\mathcal{B}}\left(u_1, u_2, \ldots, u_{ar(f)}\right)$ für alle $f \in \mathbb{F}$ und für alle $u_1, u_2, \ldots, u_{ar(f)} \in U$,
3. $R^{\mathcal{A}} = R^{\mathcal{B}}$ für alle $R \in \mathbb{R}$ und
4. $c^{\mathcal{A}} = c^{\mathcal{B}}$ für alle $c \in \mathbb{K}$.

□

Falls zwei Σ-Strukturen $\mathcal{A}$ und $\mathcal{B}$ identisch sind, dann schreiben wir dafür $\mathcal{A} = \mathcal{B}$.

2.2 Mathematische Logik

Die Wissenschaft des vernünftigen Schlussfolgerns bezeichnet man als *Logik*. Die Logik analysiert das schlussfolgernde Denken und beschreibt dessen Formen. Weiterhin beschäftigt sich die Logik mit der Formulierung von Prinzipien für die Gültigkeit von logischen Schlüssen. Falls bei diesen Betrachtungen von dem Inhalt bzw. dem Sinn von Aussagen abstrahiert wird, dann spricht man von *formaler Logik*. Für die Informatik ist besonders die *mathematische Logik* von großer Bedeutung.

Diese ist ein Teilgebiet der formalen Logik und untersucht das logische Schlussfolgern in künstlich definierten Sprachen, die auch *logische Sprachen* genannt werden.

In diesem Abschnitt werden grundlegende Begriffe der mathematischen Logik formal definiert. Die angegebenen Definitionen werden aus [29] übernommen.

Definition 2.8 (formale Sprache, aus [29])
Eine *formale Sprache* $\mathcal{FS} = \langle \mathcal{AL}, \mathcal{AUS} \rangle$ ist definiert durch:

- eine Menge $\mathcal{AL}$ von atomaren Symbolen, die *Alphabet* genannt wird und
- eine Menge $\mathcal{AUS} \subseteq \mathcal{AL}^*$, welche als *Ausdrucksmenge* bezeichnet wird.

□

Das Alphabet einer formalen Sprache legt also die Menge der Zeichen fest, aus denen komplexe Zeichenketten gebildet werden können. Die Ausdrucksmenge besteht aus der Menge von Zeichenketten über dem Alphabet, die als gültige *Wörter* der formalen Sprache akzeptiert werden. Da die Erzeugung gültiger Wörter in der Regel nach festgelegten Vorschriften erfolgt, beinhaltet die Menge $\mathcal{AUS}$ im Allgemeinen nur einen Teil der Zeichenfolgen aus $\mathcal{AL}^*$. In der Informatik werden die Elemente der Menge $\mathcal{AUS}$ häufig als *Formeln* oder *logische Formeln* bezeichnet. In diesem Zusammenhang spricht man bei der Menge $\mathcal{AUS}$ von *Formelmenge*.

Eine formale Sprache beschreibt die Struktur von Formeln, legt aber nicht deren Bedeutung fest. Um Formeln deuten zu können, werden diese mit einem *Interpretationsbereich* verknüpft.

Definition 2.9 (logische Sprache, aus [29])
Eine *logische Sprache* $\mathcal{LS} = \langle \mathcal{FS}, \mathcal{IB}, \models \rangle$ besteht aus:

- einer formalen Sprache $\mathcal{FS} = \langle \mathcal{AL}, \mathcal{AUS} \rangle$,
- einer Menge $\mathcal{IB}$, die *Interpretationsbereich* genannt wird und
- einer Relation $\models \subseteq \mathcal{IB} \times \mathcal{AUS}$.

□

Die Elemente eines Interpretationsbereichs werden *Interpretationen* genannt. Die Relation $\models$ bezeichnet man in der Literatur als *Semantikrelation*, *Modellrelation* oder *Erfüllbarkeitsrelation*.

Sei $I \in \mathcal{IB}$ eine Interpretation und $F \in \mathcal{AUS}$ eine Formel. Für $\langle I, F \rangle \in \models$ werden wir auch $I \models F$ schreiben. In diesem Zusammenhang spricht man üblicherweise davon, dass die Interpretation I die Formel F *erfüllt*. Weiterhin vereinbaren wir für eine logische Sprache $\mathcal{LS} = \langle \mathcal{FS}, \mathcal{IB}, \models \rangle$ auch die Schreibweise $\mathcal{LS} = \langle \langle \mathcal{AL}, \mathcal{AUS} \rangle, \mathcal{IB}, \models \rangle$, wobei $\mathcal{FS} = \langle \mathcal{AL}, \mathcal{AUS} \rangle$ eine formale Sprache bezeichnet.

Bei der Analyse von logischen Sprachen unterscheidet man zwischen der Syntax und der Semantik einer logischen Sprache. Unter der *Syntax* versteht man Vorschriften, die definieren wie aus den Symbolen eines Alphabets wohlgeformte Ausdrücke gebildet werden. Die Syntax einer logischen Sprache wird durch die zugehörige formale Sprache festgelegt. Jedoch ist nicht nur die Syntax einer logischen Formel relevant, sondern auch deren Bedeutung. Diese wird durch die *Semantik* festgelegt. Die Zuweisung von Interpretationen zu Formeln nach festgelegten Regeln mittels der Semantikrelation ist eine Möglichkeit, um wohlgeformten Ausdrücken eine Semantik zuzuordnen. Die einzelnen Interpretationen beschreiben Sachverhalte der Anschauung oder des Denkens auf einer abstrakten Art und Weise und müssen nicht der Realität entsprechen. Diejenigen Interpretationen, die einer konkreten Formel zugeordnet werden, bezeichnet man als *Modelle* dieser Formel und die Semantik dementsprechend als *modelltheoretische Semantik*. Die formale Definition der Interpretationen ist abhängig von der jeweiligen logischen Sprache. In dieser Arbeit werden logische Sprachen untersucht, die mithilfe der Mengentheorie interpretiert werden, d.h. deren Interpretationsbereiche aus *mengentheoretischen Strukturen* bestehen.

In der folgenden Definition wird der bereits informal eingeführte Begriff eines Modells formal festgelegt.

Definition 2.10 (Modellmenge)
Sei $\mathcal{LS} = \langle\langle \mathcal{AL}, \mathcal{AUS}\rangle, \mathcal{IB}, \models\rangle$ eine logische Sprache und $F \subseteq \mathcal{AUS}$ eine Formelmenge von $\mathcal{LS}$.
Wir definieren: $I \models F$ genau dann, wenn $I \models \varphi$ für alle $\varphi \in F$.
Die Menge $Mod(F) = \{I \mid I \in \mathcal{IB} \text{ und } I \models F\}$ heißt *Modellmenge* von F.
□

Die Modellmenge $Mod(F)$ beinhaltet also alle Interpretationen des Interpretationsbereichs, die alle Formeln der Formelmenge F erfüllen.

Mithilfe der Modellmenge lassen sich die folgenden Bezeichnungen für Formelmengen festlegen.

Definition 2.11 (Kontradiktion, erfüllbar, Tautologie)
Sei $\mathcal{LS} = \langle\langle \mathcal{AL}, \mathcal{AUS}\rangle, \mathcal{IB}, \models\rangle$ eine logische Sprache und $F \subseteq \mathcal{AUS}$ eine Formelmenge von $\mathcal{LS}$.

(a) F ist eine *Kontradiktion*, falls $Mod(F) = \emptyset$.

(b) F ist *erfüllbar*, falls $Mod(F) \neq \emptyset$.

(c) F ist eine *Tautologie*, falls $Mod(F) = \mathcal{IB}$.

□

Unter Verwendung der Modellmenge wird die semantisch fundierte Folgerungsrelation eingeführt.

Definition 2.12 (Folgerungsrelation)
Sei $\mathcal{LS} = \langle\langle \mathcal{AL}, \mathcal{AUS}\rangle, \mathcal{IB}, \models\rangle$ eine logische Sprache. Weiterhin seien $F \subseteq \mathcal{AUS}$ eine Formelmenge und $\varphi \in \mathcal{AUS}$ eine Formel von $\mathcal{LS}$.
Es gilt: $F \models_{\mathcal{IB}} \varphi$ genau dann, wenn $Mod(F) \subseteq Mod(\varphi)$. Die Relation $\models_{\mathcal{IB}}$ bezeichnet man als *Folgerungsrelation*.
Die Menge $\mathcal{C}_{\mathcal{IB}}(F) = \{\varphi \mid F \models_{\mathcal{IB}} \varphi\}$ heißt *Folgerungsmenge* der Formelmenge F. □

Falls $F \models_{\mathcal{IB}} \varphi$, dann sagt man auch: aus F *folgt* φ. Dies ist genau dann der Fall, wenn jedes Modell von F auch ein Modell von φ ist.
Bei der Charakterisierung von logischen Sprachen werden häufig die Eigenschaften der Folgerungsrelation bzw. der Folgerungsmenge untersucht. In dieser Arbeit sind die nachfolgend aufgeführten Eigenschaften von Bedeutung. Es werden die Definitionen von Jens Dietrich und Heinrich Herre [21] verwendet.

Definition 2.13 (Eigenschaften der Folgerungsrelation, aus [21])
Sei $\mathcal{LS} = \langle\langle \mathcal{AL}, \mathcal{AUS}\rangle, \mathcal{IB}, \models\rangle$ eine logische Sprache.

(a) Falls für alle $F, G \subseteq \mathcal{AUS}$ gilt: wenn $F \subseteq G$, so ist auch $\mathcal{C}_{\mathcal{IB}}(F) \subseteq \mathcal{C}_{\mathcal{IB}}(G)$, dann heißt $\models_{\mathcal{IB}}$ *monoton*.

(b) Seien $F \subseteq \mathcal{AUS}$ und $\varphi \in \mathcal{AUS}$. Die Folgerungsrelation $\models_{\mathcal{IB}}$ wird als *kompakt* bezeichnet, falls $\varphi \in \mathcal{C}_{\mathcal{IB}}(F)$ die Existenz einer endlichen Formelmenge $G \subseteq F$ mit $\varphi \in \mathcal{C}_{\mathcal{IB}}(G)$ impliziert.

(c) Seien $F, G \subseteq \mathcal{AUS}$. Falls $F \subseteq G \subseteq \mathcal{C}_{\mathcal{IB}}(F)$ die Gleichung $\mathcal{C}_{\mathcal{IB}}(F) = \mathcal{C}_{\mathcal{IB}}(G)$ impliziert, dann heißt $\models_{\mathcal{IB}}$ *kumulativ*.

□

Abschließend führen wir den *Endlichkeitssatz* ein, welcher in Kapitel 4.6 von Bedeutung sein wird.

Satz 2.14 (Endlichkeitssatz, aus [29])
Sei $F \subseteq \mathcal{AUS}$ eine Formelmenge.
Wenn jede endliche Teilmenge von F ein Modell hat, dann besitzt auch F ein Modell.

2.3 Ontologien

Nachdem im vorangegangenen Abschnitt wichtige Begriffe der mathematischen Logik eingeführt wurden, werden wir in diesem Teil der Diplomarbeit den Fachausdruck *Ontologie* detailliert untersuchen. Als Erstes beschreiben wir, was unter einer Ontologie zu verstehen ist und verdeutlichen dies mithilfe eines komplexen Beispiels. Danach werden Anwendungsgebiete und Klassifikationsmöglichkeiten für Ontologien erläutert.

2.3.1 Der Ontologiebegriff

Die Bezeichnung *Ontologie* stammt von den beiden griechischen Wörtern *ontos* (zu deutsch: Sein) und *logos* (zu deutsch: Lehre, Wort) ab und lässt sich als Lehre vom Sein bzw. Lehre vom Seienden übersetzen. Ursprünglich bezeichnet Ontologie eine Disziplin der Philosophie, welche die Grundstrukturen der Realität analysiert.

Der Begriff Ontologie im Sinne der Informatik wurde von Thomas R. Gruber [32, S. 199] als *"eine explizite formale Spezifikation einer Konzeptualisierung"* (*"An ontology is an explicit specification of a conceptualization."*) charakterisiert. Als *Konzeptualisierung* bezeichnet Thomas R. Gruber eine abstrakte und vereinfachte Sicht auf einen zu repräsentierenden Bereich der Welt. Eine Konzeptualisierung besteht aus den Objekten, Konzepten und sonstigen Dingen dieses Weltausschnitts sowie den Beziehungen zwischen diesen [32, S. 199].

Der Sichtweise von Thomas R. Gruber folgend, vereinbaren wir die nachstehende Definition:

Definition 2.15 (Ontologie)
Eine *Ontologie* ist eine Spezifikation von Begriffen sowie deren Beziehungen untereinander und stellt damit eine abstrakte Beschreibung eines abgegrenzten Wissensbereichs dar. Als Bestandteile einer Ontologie unterscheidet man *Klassen*, *Instanzen*, *Relationen* und *Axiome*.

□

Unter einer *Klasse* verstehen wir eine Menge von Objekten des zu charakterisierenden Wissensbereichs mit gemeinsamen Eigenschaften. Betrachtet man zum Beispiel die Domäne Deutschland, dann ließe sich die Klasse `Stadt` definieren, die alle deutschen Städte beinhaltet.

Instanzen repräsentieren einzelne Objekte einer Ontologie. In unserem Beispiel wäre die Stadt `Leipzig` eine solche Instanz, die als ein Element der Klasse `Stadt` charakterisiert werden könnte.

Eine Ontologie beschreibt außerdem die Beziehungen zwischen einzelnen Instanzen oder Klassen. Diese Beziehungen werden als *Relationen* bezeichnet. Man könnte das Beispiel um die Klasse `Mensch` erweitern, die alle Menschen, welche in Deutschland leben, umfasst. Zwischen den Instanzen dieser beiden Klassen wäre eine Relation mit dem Namen `wohntIn` denkbar, die den Elementen der Klasse `Mensch` jeweils ihren Wohnort zuweist.

Sehr wichtige Relationen in Ontologien sind die *istOberklasseVon*-Beziehung sowie die *istUnterklasseVon*-Beziehung. Unter Zuhilfenahme dieser Relationen können die einzelnen Klassen einer Ontologie hierarchisch gegliedert werden. Derartige hierarchische Klassifikationen bezeichnet man in der Literatur als *Taxonomien*. Das Beispiel von oben fortführend, definieren wir die Klasse `Hauptstadt`, welche nur eine einzige Instanz beinhaltet, nämlich die Hauptstadt Deutschlands. Diese Klasse ist eine *Unterklasse* der allgemeineren Klasse `Stadt`. Demgegenüber wird die Klasse `Stadt` als *Oberklasse* der spezialisierteren Klasse `Hauptstadt` bezeichnet.

Des Weiteren beinhaltet eine Ontologie häufig eine Reihe von *Axiomen*. Dies sind allgemeingültige Aussagen über den jeweiligen Wissensbereich der Ontologie. Eine solche Aussage über Menschen und Städte in Deutschland ist beispielsweise die Bedingung, dass jeder Mensch einen Wohnort haben muss, also jede Instanz der Klasse `Mensch` mit einer Instanz der Klasse `Stadt` durch die zweistellige Relation `wohntIn` verbunden ist.

2.3.2 Eine Beispielontologie

Im nachfolgenden Abschnitt wollen wir uns ein konkretes Beispiel für eine Ontologie ansehen. Diese Beispielontologie beschreibt familiäre Beziehungen zwischen Menschen. Sie soll der Veranschaulichung dienen und erhebt daher keinen Anspruch auf Vollständigkeit. Des Weiteren wird die Ontologie bewusst in natürlicher Sprache formuliert.
Die Beispielontologie soll den folgenden Sachverhalt formal wiedergeben:

- Die zu beschreibende Domäne beschäftigt sich ausschließlich mit Menschen, d.h. jedes Objekt ist ein Mensch.
- Jeder Mensch ist entweder ein Mann oder eine Frau.
- Ein Mann ist ein Vater genau dann, wenn er mindestens einen Sohn hat.
- Ein Mann ist ein Sohn genau dann, wenn er einen Vater und eine Mutter hat.
- Eine Frau ist eine Mutter genau dann, wenn sie mindestens einen Sohn hat.
- Eine Mutter heißt kinderreich genau dann, wenn sie mindestens drei Söhne hat.
- Zwei Söhne heißen Brüder genau dann, wenn sie den selben Vater und die selbe Mutter haben.
- Jeder Sohn hat genau einen Vater und genau eine Mutter.
- Ein Sohn kann nicht sein eigener Vater sein.
- Thomas ist ein Mann, dessen Vater Paul heißt.
- Die Mutter von Thomas trägt den Namen Julia.

Die Begriffe Mensch, Mann, Frau, Vater, Sohn, Mutter und kinderreiche Mutter werden wir durch Klassen modellieren, so dass die Beispielontologie aus insgesamt sieben Klassen aufgebaut ist.

Weiterhin lassen sich die folgenden Relationen erkennen:

- Eine Relation zwischen den beiden Klassen `Vater` und `Sohn`, die Vater-Sohn-Beziehung,
- eine Relation zwischen den zwei Klassen `Sohn` und `Mutter`, die Sohn-Mutter-Beziehung,

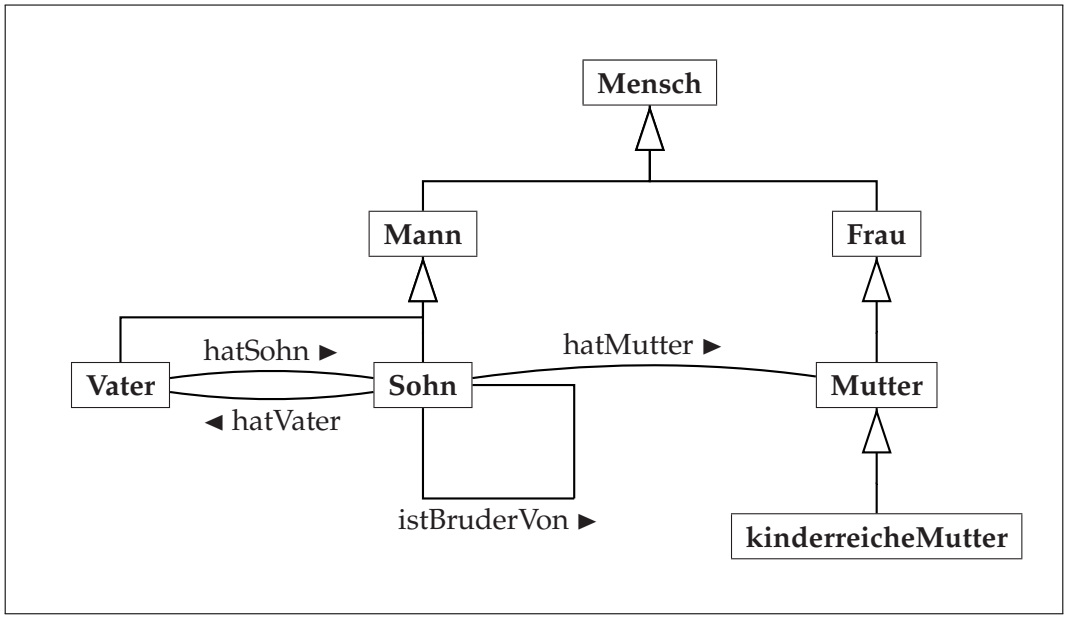

Abbildung 2.1: Klassendiagramm der Beispielontologie

- eine Relation innerhalb der Klasse `Sohn`, die Bruder-Beziehung und
- zwei Relationen, welche die Generalisierungs- bzw. Spezialisierungsbeziehungen zwischen den einzelnen Klassen modellieren.

Die Vater-Sohn-Beziehung der Beispielontologie ist in beiden Leserichtungen von Interesse und wird daher mithilfe der beiden zweistelligen Relationen `hatSohn` und `hatVater` modelliert. Die Sohn-Mutter-Beziehung wird durch die zweistellige Relation `hatMutter` dargestellt. Die binäre Relation `istBruderVon` bezeichnet die Bruder-Beziehung innerhalb der Elemente der Klasse `Sohn`. Die Generalisierungsbeziehungen werden mithilfe der Relation `istOberklasseVon` ausgedrückt. Analog dazu werden die Spezialisierungsrelationen unter Verwendung der Relation `istUnterklasseVon` formuliert. Eine Übersicht über die spezifizierten Klassen und deren Beziehungen untereinander zeigt die Abbildung 2.1 in Form eines UML-Klassendiagramms [54], [55].

Weiterhin enthält die Ontologie die drei Instanzen `Thomas`, `Paul` und `Julia`. Das UML-Objektdiagramm [54], [55] in Abbildung 2.2 auf der nächsten Seite zeigt die Klassenzugehörigkeiten der einzelnen Instanzen. Für jede Instanz ist die spezialisierteste Klasse angegeben, welche die entsprechende Instanz beinhaltet. Darüber hinaus sind die Relationen der Instanzen untereinander in dem Objektdiagramm dargestellt.

Unter Verwendung der beschriebenen Überlegungen konstruieren wir die Ontologie, welche den Sachverhalt am Beginn dieses Abschnitts modelliert, folgender-

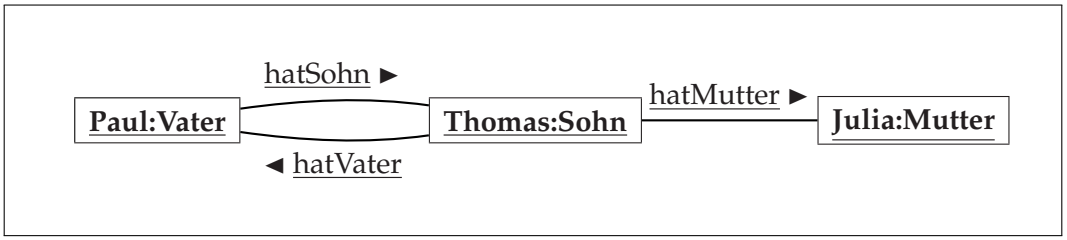

Abbildung 2.2: Objektdiagramm der Beispielontologie

maßen:

Beispiel 2.16 (Beispielontologie über familiäre Beziehungen)

1. Die Ontologie beinhaltet die Klassen `Mensch`, `Mann`, `Frau`, `Vater`, `Sohn`, `Mutter` und `kinderreicheMutter`.
   Die einzelnen Klassen sind wie folgt definiert:
   a) Die Klasse `Mensch` beinhaltet alle Instanzen der Ontologie.
   b) Die Klasse `Vater` umfasst alle Instanzen der Klasse `Mann`, die mit mindestens einer Instanz der Klasse `Sohn` durch die Relation `hatSohn` verbunden sind.
   c) Die Klasse `Sohn` besteht aus allen Instanzen der Klasse `Mann`, die
      i. mit mindestens einer Instanz der Klasse `Vater` durch die Relation `hatVater` und
      ii. mit mindestens einer Instanz der Klasse `Mutter` durch die Relation `hatMutter`

      in Beziehung stehen.
   d) Die Klasse `Mutter` enthält alle Instanzen der Klasse `Frau`, die mit mindestens einer Instanz der Klasse `Sohn` durch die Relation `hatMutter` verknüpft sind.
   e) Die Klasse `kinderreicheMutter` besteht aus allen Instanzen der Klasse `Mutter`, die mit mindestens drei paarweise verschiedenen Instanzen der Klasse `Sohn` durch die Relation `hatMutter` verbunden sind.

2. Als Instanzen enthält die Ontologie die Menschen `Thomas`, `Paul` und `Julia`. Für diese Instanzen gelten folgende Klassenzugehörigkeiten:
   a) Die Instanz `Thomas` ist ein Element der Klasse `Sohn`.
   b) Die Instanz `Paul` ist ein Element der Klasse `Vater`.
   c) Die Instanz `Julia` ist ein Element der Klasse `Mutter`.

3. Weiter beschreibt die Ontologie die binären Relationen `istOberklasseVon`, `istUnterklasseVon`, `hatSohn`, `hatVater`, `hatMutter` und `istBruderVon`. Für diese Relationen werden die nachfolgenden Eigenschaften gefordert.
   a) Die Relation `istOberklasseVon`
      i. ist reflexiv, transitiv, antisymmetrisch und
      ii. legt die folgenden Klassenbeziehungen fest:
         A. Die Klasse `Mensch` `istOberklasseVon` der Klasse `Mann`.
         B. Die Klasse `Mensch` `istOberklasseVon` der Klasse `Frau`.
         C. Die Klasse `Mann` `istOberklasseVon` der Klasse `Vater`.
         D. Die Klasse `Mann` `istOberklasseVon` der Klasse `Sohn`.
         E. Die Klasse `Frau` `istOberklasseVon` der Klasse `Mutter`.
         F. Die Klasse `Mutter` `istOberklasseVon` der Klasse `kinderreicheMutter`.
   b) Die Relation `istUnterklasseVon`
      i. ist reflexiv, transitiv, antisymmetrisch und
      ii. definiert die nachfolgenden Beziehungen:
         A. Die Klasse `Vater` `istUnterklasseVon` der Klasse `Mann`.
         B. Die Klasse `Sohn` `istUnterklasseVon` der Klasse `Mann`.
         C. Die Klasse `kinderreicheMutter` `istUnterklasseVon` der Klasse `Mutter`.
         D. Die Klasse `Mutter` `istUnterklasseVon` der Klasse `Frau`.
         E. Die Klasse `Mann` `istUnterklasseVon` der Klasse `Mensch`.
         F. Die Klasse `Frau` `istUnterklasseVon` der Klasse `Mensch`.
   c) Die Relation `hatSohn`
      i. ist eine Teilmenge des kartesischen Produkts der Klassen `Vater` und `Sohn`, also $\texttt{hatSohn} \subseteq \texttt{Vater} \times \texttt{Sohn}$ und
      ii. verbindet die Instanz `Paul` mit der Instanz `Thomas`, also $\langle \texttt{Paul}, \texttt{Thomas} \rangle \in \texttt{hatSohn}$.
   d) Die Relation `hatVater`
      i. ist eine Teilmenge des kartesischen Produkts der Klassen `Sohn` und `Vater`, also $\texttt{hatVater} \subseteq \texttt{Sohn} \times \texttt{Vater}$ und
      ii. verbindet die Instanz `Thomas` mit der Instanz `Paul`, also $\langle \texttt{Thomas}, \texttt{Paul} \rangle \in \texttt{hatVater}$.
   e) Die Relation `hatMutter`

i. ist eine Teilmenge des kartesischen Produkts der Klassen `Sohn` und `Mutter`, also `hatMutter` $\subseteq$ `Sohn` $\times$ `Mutter` und

ii. setzt die Instanz `Thomas` mit der Instanz `Julia` in Beziehung, also $\langle$`Thomas`, `Julia`$\rangle \in$ `hatMutter`.

f) Die Relation `istBruderVon` ist eine Teilmenge des kartesischen Produkts der Klasse `Sohn`, also `istBruderVon` $\subseteq$ `Sohn` $\times$ `Sohn`.

4. Außerdem beinhaltet die Ontologie die folgenden Axiome:

a) Falls die Klasse Kl_1 mit der Klasse Kl_2 durch die zweistellige Relation `istOberklasseVon` in Beziehung steht, dann ist jede Instanz der Klasse Kl_2 auch eine Instanz der Klasse Kl_1.

b) Falls die Klasse Kl_1 mit der Klasse Kl_2 durch die zweistellige Relation `istUnterklasseVon` in Beziehung steht, dann ist jede Instanz der Klasse Kl_1 auch eine Instanz der Klasse Kl_2.

c) Jede Instanz der Klasse `Mensch` ist entweder ein Element der Klasse `Mann` oder ein Element der Klasse `Frau`.

d) Wenn zwei Instanzen der Klasse `Sohn` durch die Relation `istBruderVon` miteinander in Verbindung stehen, dann sind sie durch die Relationen `hatVater` und `hatMutter` mit den selben Instanzen der Klassen `Vater` und `Mutter` verbunden.

e) Jede Instanz der Klasse `Sohn` ist mit genau einer Instanz der Klasse `Vater` durch die Relation `hatVater` verknüpft.

f) Jede Instanz der Klasse `Sohn` wird mit genau einer Instanz der Klasse `Mutter` durch die Relation `hatMutter` verbunden.

g) Es existiert keine Instanz der Klasse `Sohn`, so dass diese mit sich selbst durch die Relation `hatVater` in Beziehung steht.

h) Die beiden Relationen `istOberklasseVon` und `istUnterklasseVon` sind invers zueinander.

i) Die beiden Relationen `hatSohn` und `hatVater` sind invers zueinander. □

2.3.3 Anwendungsgebiete für Ontologien

Ontologien werden heutzutage in der Informatik in vielfältigen Bereichen eingesetzt. Michael Gruninger und Jintae Lee [33] nennen die allgemeine Kommunikation, das automatische Schließen sowie die Wissensorganisation und die Wiederverwendung von Wissen als Anwendungsfelder für Ontologien.

Sowohl bei der Kommunikation zwischen Menschen, als auch bei der Kommunikation zwischen Maschinen und der Mensch-Maschine-Kommunikation ist es

wichtig, dass die einzelnen Kommunikationsteilnehmer die zu übertragenden Daten identisch interpretieren. Der Einsatz von Ontologien ist eine Möglichkeit um einheitliche und eindeutige Interpretationsvorschriften für Daten sicher zu stellen.

Weiterhin sind Ontologien für das automatische Schließen von großer Bedeutung, da aus *explizit* modellierten Wissen mithilfe einer Ontologie zusätzliches *implizites* Wissen abgeleitet werden kann. Unter Verwendung der Ontologie aus Beispiel 2.16 lässt sich aus dem Wissen, dass `Thomas` eine Instanz der Klasse `Sohn` ist, beispielsweise folgern, dass `Thomas` auch eine Instanz der Klasse `Mann` ist. Aufgrund der formalen Definition von Ontologien sind solche Schlussfolgerungen auch automatisiert, d.h. mithilfe von Maschinen möglich.

Bei der Wiederverwendung und der Organisation von Wissen werden Ontologien vor allem für eine formale Spezifikation von *Metadaten* eingesetzt. *Ontologiebasierte Metadaten* ermöglichen zum Beispiel sehr präzise Suchanfragen. Als Metadaten werden Daten bezeichnet, die Informationen über andere Daten repräsentieren. Wird beispielsweise der Inhalt eines Buches als eine Menge von Daten aufgefasst, dann sind Informationen wie der Name des Autors oder das Erscheinungsjahr des Buches Metadaten. Im Wissensmanagement werden Metadaten für die Strukturierung großer Wissensbestände und die Verbesserung der Wiederauffindbarkeit bzw. Wiederverwendbarkeit von Wissen verwendet. Für eine detaillierte Einführung in das Themengebiet Wissensmanagement und die Verwendung von Ontologien in dieser Anwendungsdisziplin, wird auf den Artikel "Wissensmanagement mit Ontologien und Metadaten"[68] von Steffen Staab verwiesen.

Aufgrund dieser Einsatzmöglichkeiten finden Ontologien in der Informatik vor allem in den Bereichen künstliche Intelligenz, Datenbanken, Informationssysteme und Softwaretechnik Verwendung. Weitere Anwendungsgebiete für Ontologien sind die Medizin und die Biologie.

Die Ontologie aus Beispiel 2.16 demonstriert, dass es prinzipiell möglich ist, Ontologien mithilfe der natürlichen Sprache zu formulieren. Allerdings werden in der Informatik für die Beschreibung von Ontologien spezielle künstliche Sprachen verwendet. Diese *Ontologiesprachen* sind logische Sprachen, wodurch eine maschinelle Verarbeitung des in der Ontologie modellierten Wissens ermöglicht wird.

2.3.4 Klassifizierung von Ontologien

Ontologien lassen sich nach verschiedenen Gesichtspunkten klassifizieren. Sehr oft werden Ontologien anhand ihres Umfangs in *leichtgewichtige Ontologien* (lightweight ontologies) und *schwergewichtige Ontologien* (heavyweight ontologies) unterteilt. Nach [30] beschreiben leichtgewichtige Ontologien Klassen und deren Eigenschaften. Zusätzlich werden auch Beziehungen zwischen den einzelnen Klassen spezifiziert. Schwergewichtige Ontologien verfeinern leichtgewichtige Ontologien, indem sie zusätzlich Axiome und Einschränkungen formulieren. Diese gro-

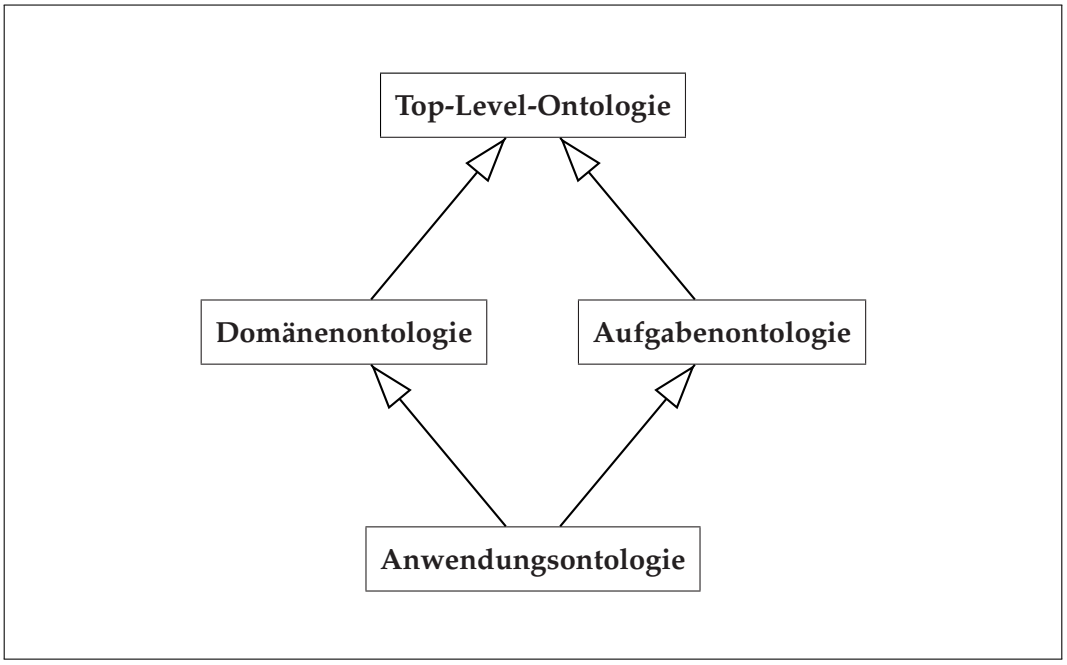

Abbildung 2.3: Ontologiearten nach Nicola Guarino

be Zweiteilung wird häufig weiter untergliedert. Eine Möglichkeit für eine feinere Strukturierung wird von Ora Lassila und Deborah McGuinness in [50] vorgeschlagen.

Mike Uschold und Michael Gruninger [71] gruppieren Ontologien anhand ihres Formalisierungsgrades. Dabei unterscheiden sie zwischen informalen Ontologien, semiinformalen Ontologien, semiformalen Ontologien und formalen Ontologien. Als *informale Ontologien* werden Ontologien bezeichnet, die in natürlicher Sprache spezifiziert sind. Falls eine natürliche Sprache nach festgelegten Regeln eingeschränkt wird, dann heißen Ontologien, die unter Verwendung dieser Sprache beschrieben werden, *semiinformal*. Eine Ontologie wird als *semiformal* bezeichnet, falls sie in einer künstlichen, formal definierten Sprache formuliert ist. *Formale Ontologien* sind Ontologien, die mithilfe einer logischen Sprache definiert werden.

Außerdem werden Ontologien nach ihrem Allgemeinheitsgrad strukturiert, um signalisieren zu können, wie universell eine Ontologie ist. Nicola Guarino führt in [34] die Bezeichnungen Top-Level-Ontologie (top-level ontology), Domänenontologie (domain ontology), Aufgabenontologie (task ontology), Anwendungsontologie (application ontology) ein. Abbildung 2.3 stellt die von Nicola Guarino charakterisierten Ontologiearten und deren Beziehungen untereinander dar.

Top-Level-Ontologien spezifizieren sehr allgemeine Begriffe und sind daher unabhängig von Anwendungsgebieten oder Aufgaben einsetzbar. Typische Top-Level-Ontologien modellieren Konzepte wie Raum und Zeit oder beschreiben universelle

Prozesse. Als Beispiel für eine Top-Level-Ontologie sei die General Formal Ontology [41] erwähnt, die an der Universität Leipzig von der Forschungsgruppe OntoMed[2] entwickelt wurde.

Ontologien, die Begriffe einer abgegrenzten Domäne wie beispielsweise Fachtermini in der Medizin definieren, werden als *Domänenontologien* bezeichnet und ausschließlich in der jeweiligen Domäne verwendet. Sehr oft basieren Domänenontologien auf Klassen von Top-Level-Ontologien und verfeinern diese. Häufig werden die Domänenontologien nach ihren Anwendungsdisziplinen benannt. So sind in [31] die Bezeichnungen Wissensrepräsentationsontologie, linguistische Ontologie, medizinische Ontologie und Chemieontologie zu finden. Entsprechend dazu werden Ontologien, die den Sachverhalt von allgemeinen Aufgaben beschreiben als *Aufgabenontologien* deklariert.

Des Weiteren wird die Bezeichnung *Anwendungsontologie* eingeführt, welche alle Ontologien umfasst, die sowohl von einer Domäne als auch von einer konkreten Aufgabe abhängig sind. Anwendungsontologien besitzen den geringsten Wiederverwendungsgrad, da sie sehr spezielles Fachwissen beschreiben.

2.4 Das Semantic Web

Nach Tim Berners-Lee und anderen [14] sind Ontologien eine Schlüsseltechnologie des Semantic Web. Daher wurden für die Realisierung des Semantic Web spezielle Ontologiesprachen entwickelt, deren semantische Untersuchung ein Bestandteil dieser Diplomarbeit ist. Dieser Abschnitt hat das Ziel eine kurze Einführung in die Welt des Semantic Web zu geben. Zunächst werden wir die Probleme des World Wide Web erläutern, die zur Entwicklung der Vision des Semantic Web geführt haben. Anschließend folgt ein kurzer Überblick über die Grundidee des Semantic Web und dessen Realisierung.

2.4.1 Probleme des World Wide Web

Die Informationen und Daten im heutigen *World Wide Web* sind fast ausschließlich für die Verwendung des Menschen vorgesehen. Unsere Computer können die Informationen des World Wide Web mittels Webbrowser zwar für Menschen geeignet präsentieren, aber eine maschinelle Verarbeitung ist derzeit nur sehr unzureichend möglich. Dies führt, aufgrund der gewaltigen Datenmengen des World Wide Web, zu einer Reihe von Problemen [44].

Ein allgemein bekanntes Beispiel ist die automatisierte Suche nach bestimmten Informationen mithilfe einer Suchmaschine. Derartige Recherchen sind gegenwärtig nur auf syntaktischer Ebene möglich, d.h. sie basieren auf dem reinen Vergleich von Zeichenketten. Interessiert man sich beispielsweise für Informationen über die

[2] http://www.onto-med.de/

Insektengattung der Hummeln und beginnt mithilfe einer Suchmaschine eine Internetrecherche zum Thema Hummel, dann ist das Ergebnis in der Regel ungenügend. Einerseits werden Dokumente angeboten, die keine relevanten Informationen zu dem Thema enthalten, sondern Firmen, Orte oder Personen beschreiben, die ebenfalls den Namen Hummel tragen. Andererseits werden wissenswerte Materialien nicht gefunden, weil diese das Suchwort Hummel nicht enthalten, da sie beispielsweise in einer anderen Sprache vorliegen.

Ein weiteres Problem ist die sehr große Heterogenität der verfügbaren Informationen, verursacht durch die dezentrale Struktur des World Wide Web. Angenommen Frau Müller möchte in der nächsten Woche die Sprechstunde eines Tierarztes aufsuchen, welcher seine Praxis in der Nähe ihres Wohnortes betreibt. Dann wäre es wünschenswert, dass automatisch aus den Informationen des World Wide Web alle Tierärzte in der Nähe ihres Wohnortes gefunden und einheitlich mit Adresse und Sprechzeiten aufgelistet werden. Automatisiert ist dies derzeit nicht möglich.

Weiterhin ist es heutzutage unmöglich aus Informationen, die im World Wide Web aufgefunden werden, auf maschinellem Weg neues Wissen zu schlussfolgern. So wäre es für Frau Müller durchaus sinnvoll, sich nicht einfach eine Liste der möglichen Tierärzte erstellen zu lassen, sondern diese Liste zusätzlich mit ihren täglichen Arbeitszeiten abzugleichen. Als Ergebnis könnte dann eine Aufstellung erzeugt werden, die nur diejenigen Tierärzte anzeigt, deren Sprechzeiten mit den Arbeitszeiten von Frau Müller vereinbar sind. Dies kann das heutige World Wide Web nicht leisten.

2.4.2 Idee und Realisierung des Semantic Web

Die in Abschnitt 2.4.1 dargestellten Szenarien sollen mithilfe des *Semantic Web* [14] Wirklichkeit werden. Die Grundidee besteht darin, die Informationen des World Wide Web mit zusätzlichen Metadaten anzureichern, die eine maschinelle Verarbeitung und Auswertung der Daten des World Wide Web ermöglichen sollen. Dies bedeutet, dass Computer die Informationen des World Wide Web nicht nur graphisch aufbereiten und anzeigen können, sondern auch in der Lage sind, diese aktiv zu verarbeiten.

Nach Auffassung des *World Wide Web Consortium (W3C)*[3] ist das Semantic Web keine separate Infrastruktur, sondern eine Erweiterung des bisherigen World Wide Web. Daran angelehnt soll auch das Semantic Web dezentral organisiert sein. Zur Absicherung der *Interoperabilität* zwischen heterogenen Anwendungen und Plattformen werden zentral vorgegebene offene Standards erarbeitet.

Die Vision des Semantic Web soll mithilfe von speziellen Technologien realisiert werden, deren Zusammenwirken unter Verwendung einer *Schichtenarchitektur* spezifiziert wird. Der Aufbau dieser Schichtenarchitektur ist in Abbildung 2.4 darge-

[3] http://www.w3.org/

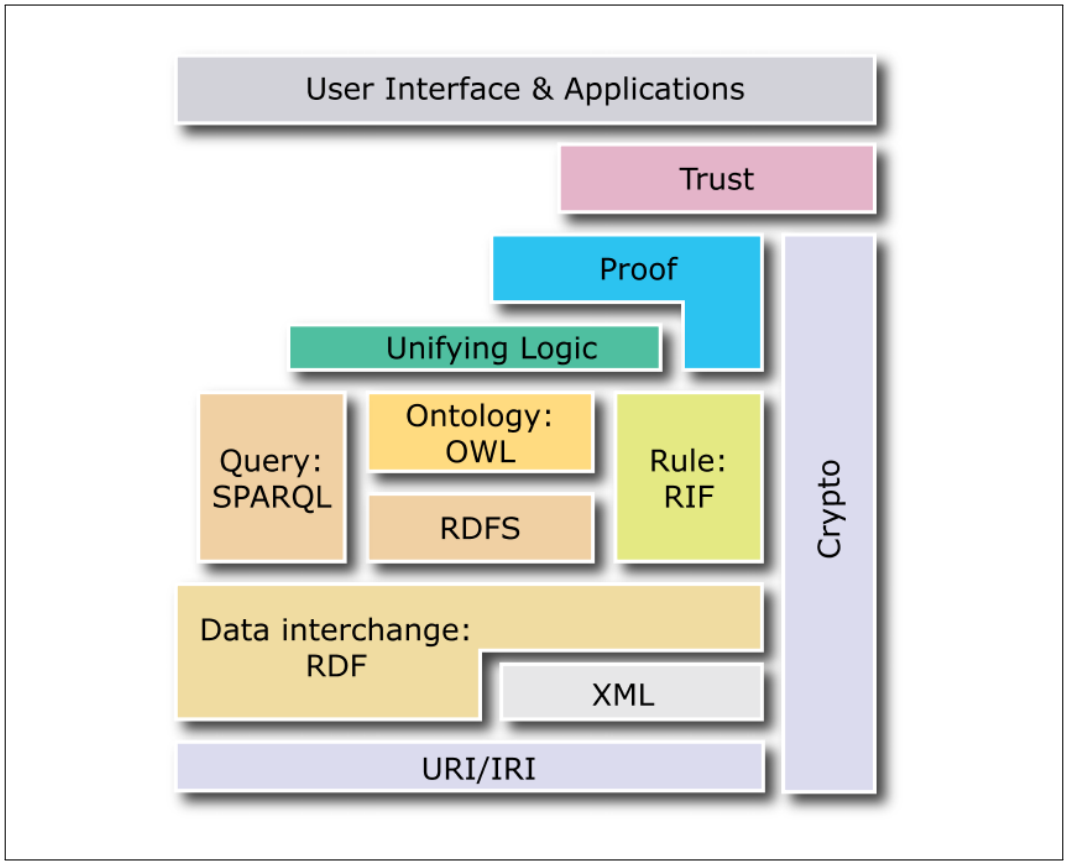

Abbildung 2.4: Die Schichtenarchitektur des Semantic Web [72].

stellt [72]. Die Anwendung des Schichtenprinzips ermöglicht einer Technologie die Nutzung von Funktionen, die durch darunter liegende Schichten bereitgestellt werden.

Die einzelnen Technologien zur Realisierung des Semantic Web werden im folgenden Teil dieses Kapitels kurz erläutert.

Die Basistechnologie des Semantic Web bilden die *Uniform Resource Identifiers (URIs)* [12] und dessen Verallgemeinerung, die *Internationalized Resource Identifiers (IRIs)* [22]. Diese Schicht stellt einen Mechanismus zur Erzeugung von weltweit eindeutigen Bezeichnern für die Benennung von Ressourcen zur Verfügung. Mithilfe von IRIs sind einheitliche Bezeichner sowohl für physikalische Ressourcen wie Menschen oder Bücher als auch für abstrakte Ressourcen wie Webseiten möglich. Die bisher am häufigsten verwendeten URIs sind die *Uniform Resource Locators (URLs)* [12]. Diese werden eingesetzt um web-adressierbare Ressourcen zu bezeichnen.

Die W3C-Recommendation der *Extensible Markup Language (XML)* [18] befin-

det sich oberhalb der URI/IRI-Schicht. Diese Auszeichnungssprache wird für die Beschreibung von Metadaten in Textdokumenten verwendet, indem einzelne Teile eines Dokuments mithilfe von *XML-Tags* geklammert werden. Die Bezeichner der XML-Tags sind frei wählbar. Um dadurch entstehende Mehrdeutigkeiten zu vermeiden und die gemeinsame Nutzung von XML-Dokumenten aus verschiedenen Quellen zu ermöglichen, werden URIs als Bestandteil der Namensgebung von XML-Tags eingesetzt, unter dessen Verwendung die *XML-Namensräume* [17] erzeugt werden. Die Strukturierung von XML-Dokumenten ist rein syntaktischer Art. Ohne zusätzliche Informationen kann ein Computer beispielsweise keine Beziehungen zwischen einzelnen Tags in einem XML-Dokument herstellen.

XML ermöglicht lediglich eine hierarchische Strukturierung einzelner Dokumente. Um Informationen über verschiedene Ressourcen miteinander in Beziehung setzen zu können und diese maschinenlesbar beschreiben zu können, wird das *Resource Description Framework (RDF)*[4] eingesetzt. RDF ist eine logische Sprache, welche die Formulierung von Aussagen in Form von *RDF-Tripeln* ermöglicht. Ein RDF-Tripel setzt ein *Subjekt* und ein *Objekt* durch ein *Prädikat* miteinander in Beziehung. Eine Menge von RDF-Tripeln beschreibt einen gerichteten Graphen, dessen Kanten die Prädikate symbolisieren und mit den Bezeichnern der Prädikate beschriftet sind. RDF setzt auf der URI/IRI-Schicht auf, um global eindeutige Bezeichner für Ressourcen und deren Beziehungen bereit zu stellen.

Das *Resource Description Framework Schema (RDFS)* [19] befindet sich direkt über der RDF-Schicht und erweitert die Ausdrucksstärke von RDF. In RDF können lediglich Aussagen über einzelne Objekte formuliert werden. RDFS erlaubt zusätzlich die Modellierung von *terminologischem Wissen*, d.h. es können auch Aussagen über Klassen und Beziehungen zwischen Klassen verfasst werden. Aufgrund dieser Ausdrucksmächtigkeit wird RDFS in der Literatur als Ontologiesprache für leichtgewichtige Ontologien bezeichnet.

Für die formale Beschreibung von komplexen Ontologien wird die *Web Ontology Language (OWL)*[5] eingesetzt. OWL basiert technisch auf der RDF-Syntax, erweitert jedoch die Ausdrucksmächtigkeit von RDFS um ein Vielfaches. OWL ermöglicht die Modellierung von schwergewichtigen Ontologien und repräsentiert die eigentliche Ontologiesprache des Semantic Web.

Für die Formulierung komplexer Anfragen hat das W3C die *SPARQL Protocol and RDF Query Language (SPARQL)* [60] entwickelt. SPARQL ist eine spezielle Anfragesprache für RDF-Dokumente und basiert auf RDF-Tripeln. Die Syntax von SPARQL ist angelehnt an die Syntax der weitverbreiteten Datenbanksprache *SQL (Structured Query Language)*. Die Semantik einer SPARQL-Anfrage wird unter Verwendung der *SPARQL-Algebra* definiert.

Eine weitere Komponente der Architektur des Semantic Web bilden Regeln. Un-

[4] http://www.w3.org/RDF/
[5] http://www.w3.org/2004/OWL/

ter Verwendung eines Regelformalismus ist es möglich aus bekanntem Wissen, beispielsweise in Form von Ontologien, neues Wissen abzuleiten. Im November 2005 wurde die W3C-Arbeitsgruppe *Rule Interchange Format Working Group*[6] gegründet, um ein standardisiertes Format für die Beschreibung und den Austausch von Regeln mit dem Namen *Rule Interchange Format (RIF)* zu erarbeiten. Bisher ist die Entwicklung von RIF noch nicht abgeschlossen, so dass zum jetzigen Zeitpunkt lediglich Entwürfe verfügbar sind. Es ist geplant das Rule Interchange Format als eine Familie von Sprachen, sogenannten *Dialekten*, zu realisieren, um der Vielzahl der existierenden Regelsprachen gerecht zu werden. Als gemeinsamer Basisdialekt für logische Regelformalismen wird der *RIF Basic Logic Dialect (RIF-BLD)* [16] vorgeschlagen.

Die verbleibenden Schichten des Semantic Web befinden sich derzeit in Planung und sind noch nicht realisiert. Das Ziel der logischen Schicht ist die Bereitstellung einer monotonen logischen Sprache [13]. Diese Logik hat die Aufgabe, die darunter liegenden Schichten mit ihren verschiedenen Ausprägungen zu vereinheitlichen und muss daher die Ausdrucksstärke der Schichten SPARQL, OWL und RIF übertreffen.

Unter Verwendung der Schicht mit der Bezeichnung *Proof* soll die Überprüfbarkeit von Aussagen gewährleistet werden. Dadurch werden automatische Schlüsse von Softwareagenten für andere Softwareanwendungen, aber auch den Menschen, nachvollziehbar. Diese Schicht muss also Methoden und Verfahren für die automatische Beweiserstellung und Beweisüberprüfung zur Verfügung stellen.

Auf den Schichten Proof und Crypto aufbauend, soll die *Trust-Schicht* eine Vertrauensbildung im Semantic Web ermöglichen. Diese Schicht behandelt Strategien, die analysieren wie stark man Informationen oder Informationsquellen vertraut.

Nachfolgend wollen wir die Komponente *Crypto* ansprechen. Die Crypto-Schicht verläuft parallel zu den anderen Schichten im Semantic Web. Diese Komponente hat die Aufgabe, Verschlüssellungstechniken für die einzelnen Schichten des Semantic Web bereit zu stellen und verkörpert damit den Grundbaustein einer Sicherheitsschicht für das Semantic Web.

Auf diese Schichtenarchitektur setzen die Softwareanwendungen unter Verwendung von wohldefinierten Schnittstellen auf. In Abbildung 2.4 wird dies durch das Modul mit der Bezeichnung *User Interface & Applications* symbolisiert.

[6] http://www.w3.org/2005/rules/wiki/RIF_Working_Group

3 Prädikatenlogik der ersten Stufe

Als Einstieg in die Welt der logischen Sprachen wählen wir die Prädikatenlogik der ersten Stufe. Diese logische Sprache ist nicht nur in der Mathematik und Informatik weit verbreitet, sondern spielt auch in der Linguistik und Philosophie eine wichtige Rolle. Zu Beginn dieses Kapitels definieren wir die formale Syntax und die formale Semantik der Prädikatenlogik erster Stufe. Darauf folgend wird das prädikatenlogische Universum näher untersucht. Mit der Modellierung der Ontologie aus Beispiel 2.16 auf Seite 20 und einer Diskussion der dadurch erkennbaren Vor- und Nachteile der Prädikatenlogik, werden wir diesen Teil der Diplomarbeit abschließen.

3.1 Syntax der Prädikatenlogik

Dieses Unterkapitel legt die Syntax der Prädikatenlogik erster Stufe fest. Wir orientieren uns dabei an den Definitionen von Hans-Joachim Goltz und Heinrich Herre [29], erweitern diese jedoch um ein zusätzliches Zeichen, mit dem Identitätsbeziehungen formuliert werden können, wie man es beispielsweise in [65] findet. Die so definierte Sprache wird als *Prädikatenlogik der ersten Stufe mit Identität* bezeichnet.

Als erstes legen wir die Menge der Basiszeichen fest, aus der komplexe Zeichenketten gebildet werden können.

Definition 3.1 (prädikatenlogisches Alphabet)
Sei $\Sigma = \langle \mathbb{F}, \mathbb{R}, \mathbb{K}, ar \rangle$ eine Signatur. Das *prädikatenlogische Alphabet* $Al(\Sigma, VAR) = VAR \cup JUN \cup QU \cup \mathbb{F} \cup \mathbb{R} \cup \mathbb{K} \cup Z \cup \{\ominus\}$ setzt sich aus den folgenden Symbolmengen zusammen:

- der Menge VAR von *Individuenvariablen*,
- der Menge $JUN = \{\neg, \wedge, \vee, \rightarrow, \leftrightarrow\}$ von *Junktoren*,
- der Menge $QU = \{\forall, \exists\}$ von *Quantoren*,
- der Menge $\mathbb{F}$ von Funktionssymbolen,
- der Menge $\mathbb{R}$ von Relationssymbolen,
- der Menge $\mathbb{K}$ von Konstantensymbolen,

- der Menge $Z = \{), (\}$ von technischen Hilfszeichen und
- dem Symbol $\ominus$.

□

Wir vereinbaren, dass die einzelnen Mengen des prädikatenlogischen Alphabets paarweise disjunkt sind. Da die Symbolmengen JUN, QU und Z sowie das Symbol $\ominus$ eines prädikatenlogischen Alphabets eindeutig festgelegt sind, können sich zwei prädikatenlogische Alphabete nur in den Symbolmenge VAR, $\mathbb{F}$, $\mathbb{R}$ und $\mathbb{K}$ unterscheiden. Es reicht also aus für ein konkretes prädikatenlogisches Alphabet $Al(\Sigma, VAR)$ die zugehörige Signatur Σ und die Menge der Individuenvariablen VAR anzugeben.

Anschließend definieren wir die Ausdrucksmenge der Prädikatenlogik. Dazu führen wir zunächst die Bezeichnung prädikatenlogischer Term ein.

Definition 3.2 (prädikatenlogischer Term)
Sei $Al(\Sigma, VAR)$ ein prädikatenlogisches Alphabet. Die Menge $Tm(\Sigma, VAR)$ der *prädikatenlogischen Terme* ist die kleinste Menge von Zeichenreihen über $Al(\Sigma, VAR)$, welche die folgenden Bedingungen erfüllt:

1. $VAR \subseteq Tm(\Sigma, VAR)$
2. $\mathbb{K} \subseteq Tm(\Sigma, VAR)$
3. Falls $t_1, t_2, \ldots, t_n \in Tm(\Sigma, VAR)$ und $f \in \mathbb{F}$ ein n-stelliges Funktionssymbol ist, dann gilt $f(t_1, t_2, \ldots, t_n) \in Tm(\Sigma, VAR)$.

□

Einen prädikatenlogischen Term, der unter Verwendung der 3. Bedingung konstruiert wird, bezeichnen wir als *prädikatenlogischen Funktionsterm*.

Definition 3.3 (prädikatenlogische Atomformel)
Es seien $t_1, t_2, \ldots, t_n \in Tm(\Sigma, VAR)$ prädikatenlogische Terme und $R \in \mathbb{R}$ ein n-stelliges Relationssymbol, dann sind folgende Zeichenreihen *prädikatenlogische Atomformeln*:

1. $R(t_1, t_2, \ldots, t_n)$ und
2. $(t_1 \ominus t_2)$

Die Menge aller prädikatenlogischen Atomformeln über $Al(\Sigma, VAR)$ bezeichnen wir mit $AtFm(\Sigma, VAR)$.

□

Nun vereinbaren wir wie die prädikatenlogischen Atomformeln zu komplexeren prädikatenlogischen Formeln erweitert werden können. Dazu führen wir die nachstehende Bezeichnung ein.
Eine Variable $x \in VAR$ heißt *vollfrei* in der Zeichenreihe F, falls x in F vorkommt, aber die Zeichenkette $\forall x$ oder $\exists x$ nicht in F enthalten ist.

Definition 3.4 (prädikatenlogische Formel)
Die Menge $Fm(\Sigma, VAR)$ der *prädikatenlogischen Formeln* ist die kleinste Menge von Zeichenreihen über dem prädikatenlogischen Alphabet $Al(\Sigma, VAR)$, welche die folgenden Bedingungen erfüllt:

1. $AtFm(\Sigma, VAR) \subseteq Fm(\Sigma, VAR)$
2. Falls $\varphi_1, \varphi_2 \in Fm(\Sigma, VAR)$ zwei prädikatenlogische Formeln sind, dann gehören auch die Zeichenreihen $\neg\varphi_1$, $(\varphi_1 \wedge \varphi_2)$, $(\varphi_1 \vee \varphi_2)$, $(\varphi_1 \rightarrow \varphi_2)$ und $(\varphi_1 \leftrightarrow \varphi_2)$ zu $Fm(\Sigma, VAR)$.
3. Falls $\varphi \in Fm(\Sigma, VAR)$, $x \in VAR$ und x kommt in φ vollfrei vor, dann gehören auch die Zeichenreihen $\forall x\varphi$ und $\exists x\varphi$ zu $Fm(\Sigma, VAR)$.

□

Bemerkung 3.5
Wenn aus dem Kontext heraus klar ist, dass es sich um Bezeichnungen der Prädikatenlogik handelt, dann verzichten wir in der Regel auf den Namenszusatz prädikatenlogisch *und schreiben beispielsweise Atomformel bzw. Formel anstelle von prädikatenlogische Atomformel bzw. prädikatenlogische Formel.*

Am Ende dieses Unterkapitels führen wir den Begriff der freien Variable ein. Dazu vereinbaren wir die folgende Symbolik: Für eine Formel $\varphi \in Fm(\Sigma, VAR)$ bezeichnet $Var(\varphi)$ die Menge der Variablen aus VAR die in φ vorkommen.

Definition 3.6 (Menge der freien Variablen)
Seien $\varphi, \psi_1, \psi_2 \in Fm(\Sigma, VAR)$ prädikatenlogische Formeln. Die Menge $fVar(\varphi)$ der *freien Variablen* von φ wird wie folgt induktiv definiert:

1. Falls φ eine Atomformel ist, dann gilt $fVar(\varphi) = Var(\varphi)$.
2. Falls $\varphi = \neg\psi$, dann gilt $fVar(\varphi) = fVar(\psi)$.
3. Falls $\varphi = (\psi_1 \wedge \psi_2)$, dann gilt $fVar(\varphi) = fVar(\psi_1) \cup fVar(\psi_2)$.
4. Falls $\varphi = (\psi_1 \vee \psi_2)$, dann gilt $fVar(\varphi) = fVar(\psi_1) \cup fVar(\psi_2)$.
5. Falls $\varphi = (\psi_1 \rightarrow \psi_2)$, dann gilt $fVar(\varphi) = fVar(\psi_1) \cup fVar(\psi_2)$.
6. Falls $\varphi = (\psi_1 \leftrightarrow \psi_2)$, dann gilt $fVar(\varphi) = fVar(\psi_1) \cup fVar(\psi_2)$.
7. Falls $\varphi = \forall x\psi_1$, dann gilt $fVar(\varphi) = fVar(\psi_1) \setminus \{x\}$.
8. Falls $\varphi = \exists x\psi_1$, dann gilt $fVar(\varphi) = fVar(\psi_1) \setminus \{x\}$.

□

In diesem Teil der Diplomarbeit haben wir die *formale Sprache der Prädikatenlogik* $\mathcal{FS}_{FOL}(\Sigma, VAR) = \langle Al(\Sigma, VAR), Fm(\Sigma, VAR)\rangle$ definiert. Im nächsten Abschnitt wollen wir diese zu einer logischen Sprache erweitern. Dazu muss die Semantik der prädikatenlogischen Formeln festgelegt werden.

3.2 Semantik der Prädikatenlogik

Um die formale Semantik der prädikatenlogischen Formeln definieren zu können, müssen wir den Interpretationsbereich und die Erfüllbarkeitsrelation festlegen. Zusätzlich werden wir eine *Variablenbelegung* einführen, welche die Interpretation der Individuenvariablen vorgibt. Die hier angegebenen Definitionen basieren ebenfalls auf [29].

Definition 3.7 (prädikatenlogische Variablenbelegung)
Sei $\Sigma = \langle \mathbb{F}, \mathbb{R}, \mathbb{K}, ar \rangle$ eine Signatur und $\mathcal{A} = \left\langle U, \left(f^{\mathcal{A}}\right)_{f \in \mathbb{F}}, \left(R^{\mathcal{A}}\right)_{R \in \mathbb{R}}, \left(c^{\mathcal{A}}\right)_{c \in \mathbb{K}} \right\rangle$ eine Σ-Struktur. Weiterhin sei VAR eine Menge von Individuenvariablen. Eine Abbildung $\mu : VAR \to U$ bezeichnen wir als *prädikatenlogische Variablenbelegung*. □

Eine prädikatenlogische Variablenbelegung ordnet also jeder Variable aus der Menge VAR ein Element aus der Grundmenge U von $\mathcal{A}$ zu. Als nächstes erweitern wir diese Variablenbelegung, um damit jedem prädikatenlogischen Term ein Element aus U zuzuweisen.

Definition 3.8 (erweiterte prädikatenlogische Variablenbelegung)
Sei $\Sigma = \langle \mathbb{F}, \mathbb{R}, \mathbb{K}, ar \rangle$ eine Signatur und $\mathcal{A} = \left\langle U, \left(f^{\mathcal{A}}\right)_{f \in \mathbb{F}}, \left(R^{\mathcal{A}}\right)_{R \in \mathbb{R}}, \left(c^{\mathcal{A}}\right)_{c \in \mathbb{K}} \right\rangle$ eine Σ-Struktur. Weiterhin sei VAR eine Menge von Variablen und $\mu : VAR \to U$ eine prädikatenlogische Variablenbelegung.
Die *erweiterte prädikatenlogische Variablenbelegung* $\mu^* : Tm(\Sigma, VAR) \to U$ ist induktiv wie folgt definiert:

1. Sei $x \in VAR$ eine Individuenvariable, dann gilt: $\mu^*(x) = \mu(x)$.
2. Sei $c \in \mathbb{K}$ ein Konstantensymbol, dann gilt: $\mu^*(c) = c^{\mathcal{A}}$.
3. Sei $f(t_1, t_2, \ldots, t_n) \in Tm(\Sigma, VAR)$ ein Funktionsterm, dann gilt:
   $\mu^*\big(f(t_1, t_2, \ldots, t_n)\big) = f^{\mathcal{A}}\big(\mu^*(t_1), \mu^*(t_2), \ldots, \mu^*(t_n)\big)$.

□

Nach obiger Konstruktionsvorschrift ist leicht zu erkennen, dass die Abbildung $\mu^* : Tm(\Sigma, VAR) \to U$ eindeutig definiert ist. Es gilt also die folgende Beobachtung:

Beobachtung 3.9
Sei $\mathcal{A} = \left\langle U, \left(f^{\mathcal{A}}\right)_{f \in \mathbb{F}}, \left(R^{\mathcal{A}}\right)_{R \in \mathbb{R}}, \left(c^{\mathcal{A}}\right)_{c \in \mathbb{K}} \right\rangle$ eine relationale Struktur über der Signatur $\Sigma = \langle \mathbb{F}, \mathbb{R}, \mathbb{K}, ar \rangle$. Des Weiteren sei VAR eine Menge von Variablen und $\mu : VAR \to U$ eine prädikatenlogische Variablenbelegung.
Dann ist die erweiterte prädikatenlogische Variablenbelegung $\mu^* : Tm(\Sigma, VAR) \to U$ eindeutig definiert.

Als nächstes führen wir den Begriff der prädikatenlogischen Interpretation ein.

Definition 3.10 (prädikatenlogische Interpretation)
Sei $Al(\Sigma, VAR)$ ein prädikatenlogisches Alphabet. Eine *prädikatenlogische Interpretation* $I = \langle \mathcal{A}, \mu \rangle$ über dem Alphabet $Al(\Sigma, VAR)$ besteht aus:

- einer Σ-Struktur $\mathcal{A} = \left\langle U, \left(f^{\mathcal{A}}\right)_{f \in \mathbb{F}}, \left(R^{\mathcal{A}}\right)_{R \in \mathbb{R}}, \left(c^{\mathcal{A}}\right)_{c \in \mathbb{K}} \right\rangle$ und
- einer prädikatenlogischen Variablenbelegung $\mu : VAR \to U$.

□

Die Menge aller prädikatenlogischen Interpretationen über dem prädikatenlogischen Alphabet $Al(\Sigma, VAR)$ heißt *prädikatenlogischer Interpretationsbereich* über dem Alphabet $Al(\Sigma, VAR)$ und wird mit $\mathcal{IB}_{FOL}(\Sigma, VAR)$ bezeichnet.

Anschließend müssen wir noch definieren, wie die prädikatenlogischen Formeln unter Verwendung des Interpretationsbereichs gedeutet werden. Dazu führen wir zunächst den Begriff der prädikatenlogischen Modifikation ein.

Definition 3.11 (prädikatenlogische Modifikation)
Seien $I, K \in \mathcal{IB}_{FOL}(\Sigma, VAR)$ zwei prädikatenlogische Interpretationen über dem Alphabet $Al(\Sigma, VAR)$ mit $I = \langle \mathcal{A}_I, \mu_I \rangle$ und $K = \langle \mathcal{A}_K, \mu_K \rangle$. Weiterhin sei $Var \subseteq VAR$ eine Menge von Individuenvariablen.
Die Interpretation K ist eine *prädikatenlogische Modifikation* der Interpretation I bezüglich der Menge Var, falls die nachfolgenden Bedingungen erfüllt sind.

1. $\mathcal{A}_I = \mathcal{A}_K$
2. $\mu_I(x) = \mu_K(x)$ für alle $x \in VAR \setminus Var$

□

Falls K eine prädikatenlogische Modifikation von I ist, dann unterscheiden sich die beiden Interpretationen I und K lediglich darin, wie sie die Variablen der Menge Var interpretieren. Die Menge aller prädikatenlogischer Modifikationen von I bezüglich Var bezeichnen wir mit $Modi_{FOL}^{Var}(I)$.

Definition 3.12 (prädikatenlogische Erfüllbarkeitsrelation)
Sei $\Sigma = \langle \mathbb{F}, \mathbb{R}, \mathbb{K}, ar \rangle$ eine Signatur und $\mathcal{A} = \left\langle U, \left(f^{\mathcal{A}}\right)_{f \in \mathbb{F}}, \left(R^{\mathcal{A}}\right)_{R \in \mathbb{R}}, \left(c^{\mathcal{A}}\right)_{c \in \mathbb{K}} \right\rangle$ eine Σ-Struktur. Weiter seien $I \in \mathcal{IB}_{FOL}(\Sigma, VAR)$ eine prädikatenlogische Interpretation über $Al(\Sigma, VAR)$ mit $I = \langle \mathcal{A}, \mu \rangle$ und $\varphi, \psi_1, \psi_2 \in Fm(\Sigma, VAR)$ prädikatenlogische Formeln.
Die *prädikatenlogische Erfüllbarkeitsrelation* wird induktiv definiert:

1. Sei $\varphi = R(t_1, t_2, \ldots, t_n)$ eine Atomformel, dann gilt: $I \models \varphi$ genau dann, wenn $\left\langle \mu^*(t_1), \mu^*(t_2), \ldots, \mu^*(t_n) \right\rangle \in R^{\mathcal{A}}$.
2. Sei $\varphi = (t_1 \ominus t_2)$ eine Atomformel, dann gilt: $I \models \varphi$ genau dann, wenn $\mu^*(t_1) = \mu^*(t_2)$.

3. Sei $\varphi = (\psi_1 \wedge \psi_2)$ eine prädikatenlogische Formel, dann gilt: $I \models \varphi$ genau dann, wenn $I \models \psi_1$ und $I \models \psi_2$.

4. Sei $\varphi = (\psi_1 \vee \psi_2)$ eine prädikatenlogische Formel, dann gilt: $I \models \varphi$ genau dann, wenn $I \models \psi_1$ oder $I \models \psi_2$.

5. Sei $\varphi = \neg\psi_1$ eine prädikatenlogische Formel, dann gilt: $I \models \varphi$ genau dann, wenn nicht $I \models \psi_1$.

6. Sei $\varphi = (\psi_1 \rightarrow \psi_2)$ eine prädikatenlogische Formel, dann gilt: $I \models \varphi$ genau dann, wenn $I \models (\neg\psi_1 \vee \psi_2)$.

7. Sei $\varphi = (\psi_1 \leftrightarrow \psi_2)$ eine prädikatenlogische Formel, dann gilt: $I \models \varphi$ genau dann, wenn $I \models (\psi_1 \rightarrow \psi_2)$ und $I \models (\psi_2 \rightarrow \psi_1)$.

8. Sei $\varphi = \exists x\psi_1$ eine prädikatenlogische Formel, dann gilt: $I \models \varphi$ genau dann, wenn eine prädikatenlogische Modifikation $K \in Modi_{FOL}^{Var}(I)$ von I bezüglich $Var = \{x\}$ existiert, so dass $K \models \psi_1$.

9. Sei $\varphi = \forall x\psi_1$ eine prädikatenlogische Formel, dann gilt: $I \models \varphi$ genau dann, wenn für alle prädikatenlogischen Modifikationen $K \in Modi_{FOL}^{Var}(I)$ von I bezüglich $Var = \{x\}$ gilt $K \models \psi_1$.

□

Nachdem wir die prädikatenlogische Erfüllbarkeitsrelation definiert haben, ist die Prädikatenlogik der ersten Stufe mit Identität vollständig als logische Sprache charakterisiert, die wir als $FOL = \left\langle \mathcal{FS}_{FOL}(\Sigma, VAR), \mathcal{IB}_{FOL}(\Sigma, VAR), \models \right\rangle$ bezeichnen wollen.

Der nächste Schritt besteht darin, den in Definition 2.12 auf Seite 16 eingeführten Begriff der Folgerungsrelation für die Prädikatenlogik der ersten Stufe zu konkretisieren. Dafür erweitern wir zunächst die Definition der prädikatenlogischen Erfüllbarkeitsrelation von einzelnen Formeln auf Formelmengen.

Definition 3.13 (prädikatenlogische Theorie)
Sei $FOL = \left\langle \langle Al(\Sigma, VAR), Fm(\Sigma, VAR) \rangle, \mathcal{IB}_{FOL}(\Sigma, VAR), \models \right\rangle$ die logische Sprache der Prädikatenlogik erster Stufe.

(a) Eine Menge $X \subseteq Fm(\Sigma, VAR)$ von prädikatenlogischen Formeln bezeichnen wir als *prädikatenlogische Theorie*.

(b) Es seien $I \in \mathcal{IB}_{FOL}(\Sigma, VAR)$ eine prädikatenlogische Interpretation und $X \subseteq Fm(\Sigma, VAR)$ eine prädikatenlogische Theorie.
Wir definieren $I \models X$ genau dann, wenn $I \models \varphi$ für alle $\varphi \in X$.

□

Nun sind wir in der Lage, die prädikatenlogische Folgerungsrelation formal zu definieren.

Definition 3.14 (Modellmenge, Folgerungsrelation, Folgerungsmenge)
Sei $\varphi \in Fm(\Sigma, VAR)$ eine prädikatenlogische Formel und $X \subseteq Fm(\Sigma, VAR)$ eine prädikatenlogische Theorie. Weiterhin sei $I \in \mathcal{IB}_{FOL}(\Sigma, VAR)$ eine prädikatenlogische Interpretation.

(a) Die *prädikatenlogische Modellmenge* von φ bezeichnen wir mit $Mod_{FOL}(\varphi)$ und definieren diese wie folgt: $Mod_{FOL}(\varphi) = \{I \in \mathcal{IB}_{FOL}(\Sigma, VAR) \mid I \models \varphi\}$.

(b) Analog zu (a) bezeichnen wir die *prädikatenlogische Modellmenge* von X mit $Mod_{FOL}(X)$ und definieren $Mod_{FOL}(X) = \{I \in \mathcal{IB}_{FOL}(\Sigma, VAR) \mid I \models X\}$.

(c) Die *prädikatenlogische Folgerungsrelation* $\models_{FOL}$ wird wie folgt definiert: $X \models_{FOL} \varphi$ genau dann, wenn $Mod_{FOL}(X) \subseteq Mod_{FOL}(\varphi)$.

(d) Die *prädikatenlogische Folgerungsmenge* von X bezeichnen wir mit $\mathcal{C}^{\models_{FOL}}(X)$ und legen diese als $\mathcal{C}^{\models_{FOL}}(X) = \{\varphi \in Fm(\Sigma, VAR) \mid X \models_{FOL} \varphi\}$ fest.

□

3.3 Charakterisierung des prädikatenlogischen Universums

Nachdem wir in den vorangegangenen Unterkapiteln der Diplomarbeit die Syntax und Semantik der Prädikatenlogik erster Stufe formal definiert haben, wollen wir in diesem Abschnitt den Aufbau des prädikatenlogischen Universums näher untersuchen.
Betrachten wir zunächst das folgende Anschauungsbeispiel:

Beispiel 3.15
Es sei $VAR = \{x_1, x_2\}$ eine Menge von Individuenvariablen. Des Weiteren sei $\Sigma = \langle \mathbb{F}, \mathbb{R}, \mathbb{K}, ar \rangle$ eine Signatur mit $\mathbb{F} = \{f\}$, $\mathbb{R} = \{R, S\}$, $\mathbb{K} = \{a, b, c\}$ und der Stelligkeitsabbildung $ar : \mathbb{F} \cup \mathbb{R} \to \mathbb{N}$, die wie folgt definiert ist: $ar(f) = 2$, $ar(R) = 1$ und $ar(S) = 2$.
Als mögliche Interpretation $I \in \mathcal{IB}_{FOL}(\Sigma, VAR)$ mit $I = \langle \mathcal{A}, \mu \rangle$ für das angegebene Vokabular wählen wir die Σ-Struktur $\mathcal{A} = \left\langle U, \left(f^{\mathcal{A}}\right)_{f \in \mathbb{F}}, \left(R^{\mathcal{A}}\right)_{R \in \mathbb{R}}, \left(c^{\mathcal{A}}\right)_{c \in \mathbb{K}} \right\rangle$ mit

1. der Grundmenge U mit $U = \{u_1, u_2, u_3\}$ und
2. der Abbildung δ, die wie folgt definiert ist:
   a) $\delta(f) = f^{\mathcal{A}}$ mit $f^{\mathcal{A}} : U^2 \to U$, so dass $f^{\mathcal{A}}(x, y) = u_3$ für alle $x, y \in U$ gilt,
   b) $\delta(R) = R^{\mathcal{A}}$ mit $R^{\mathcal{A}} = \{u_2, u_3\}$,
   c) $\delta(S) = S^{\mathcal{A}}$ mit $S^{\mathcal{A}} = \{\langle u_1, u_1 \rangle, \langle u_1, u_2 \rangle, \langle u_1, u_3 \rangle, \langle u_2, u_1 \rangle\}$ sowie
   d) $\delta(a) = u_1, \delta(b) = u_2$ und $\delta(c) = u_3$

und die Variablenbelegung $\mu : VAR \to U$ mit $\mu(x_1) = u_1$ und $\mu(x_2) = u_2$.

□

Abbildung 3.1 auf der nächsten Seite skizziert, in Anlehnung an [45, S. 42], den Aufbau des prädikatenlogischen Universums von Beispiel 3.15.

Es ist erkennbar, dass die prädikatenlogischen Bezeichner bereits auf der syntaktischen Ebene in vier paarweise disjunkte Mengen gegliedert werden. Man unterscheidet zwischen Individuenvariablen, Individuenkonstanten, Funktions- und Relationskonstanten. Jeder Name, welcher eine Funktion oder eine Relation bezeichnet, besitzt eine feste Stelligkeit, die durch die zugehörige Signatur festgelegt wird.

Die prädikatenlogischen Funktions-, Relations- und Konstantenbezeichner werden unter Verwendung einer relationalen Struktur interpretiert. In Beispiel 3.15 erfolgt dies mithilfe der Struktur $\mathcal{A}$. Die Variablensymbole werden mittels einer prädikatenlogischen Variablenbelegung gedeutet, in Abbildung 3.1 dargestellt durch die Funktion μ. Prinzipiell wird zwischen freien und gebundenen Variablen unterschieden, wobei eine gebundene Variable entweder existenzquantifiziert oder allquantifiziert ist. Da Variablenbezeichner ebenfalls über der Grundmenge einer Σ-Struktur gedeutet werden, umfasst das prädikatenlogische Universum lediglich eine Menge, die Grundmenge der zugehörigen Σ-Struktur.

In der Prädikatenlogik erster Stufe werden alle Variablensymbole als Bezeichner für Individuen behandelt. Dies bedeutet, dass es nicht möglich ist über Funktionen und Relationen zu quantifizieren. Die Reichweite der beiden prädikatenlogischen Quantoren erstreckt sich über die gesamte Grundmenge einer Σ-Struktur.

Namen, die Individuen benennen, werden gedeutet, indem diesen Namen Elemente aus der Grundmenge einer Σ-Struktur zugewiesen werden. Demgegenüber bezeichnen Funktionssymbole und Relationssymbole keine Elemente der Grundmenge, sondern diese werden als Tupel über der Grundmenge interpretiert, wobei die Länge der Tupel von der jeweiligen Stelligkeit der Symbole abhängig ist. Die Grundmenge eines prädikatenlogischen Universums enthält also lediglich die Elemente, die als Individuen angesehen werden.

3.4 Modellierung von Ontologien

Den Themenkomplex der Prädikatenlogik erster Stufe wollen wir mit einem Abschnitt beenden, welcher die Vor- und Nachteile der Prädikatenlogik bei der Modellierung von Ontologien beschreibt.

Der Hauptgrund für die Verwendung der Prädikatenlogik als Beschreibungssprache für Ontologien ist die hohe Ausdruckskraft der Prädikatenlogik erster Stufe. Dadurch lassen sich auch sehr komplexe Sachverhalte mithilfe der Prädikatenlogik formulieren. Die Ontologie aus Beispiel 2.16 auf Seite 20 kann problemlos unter Verwendung der Prädikatenlogik beschrieben werden, dargestellt in Abbildung 3.2 auf Seite 41 und Abbildung 3.3 auf Seite 42.
Benannte Instanzen wie `Thomas`, `Paul` oder `Julia` werden als Konstanten im prädi-

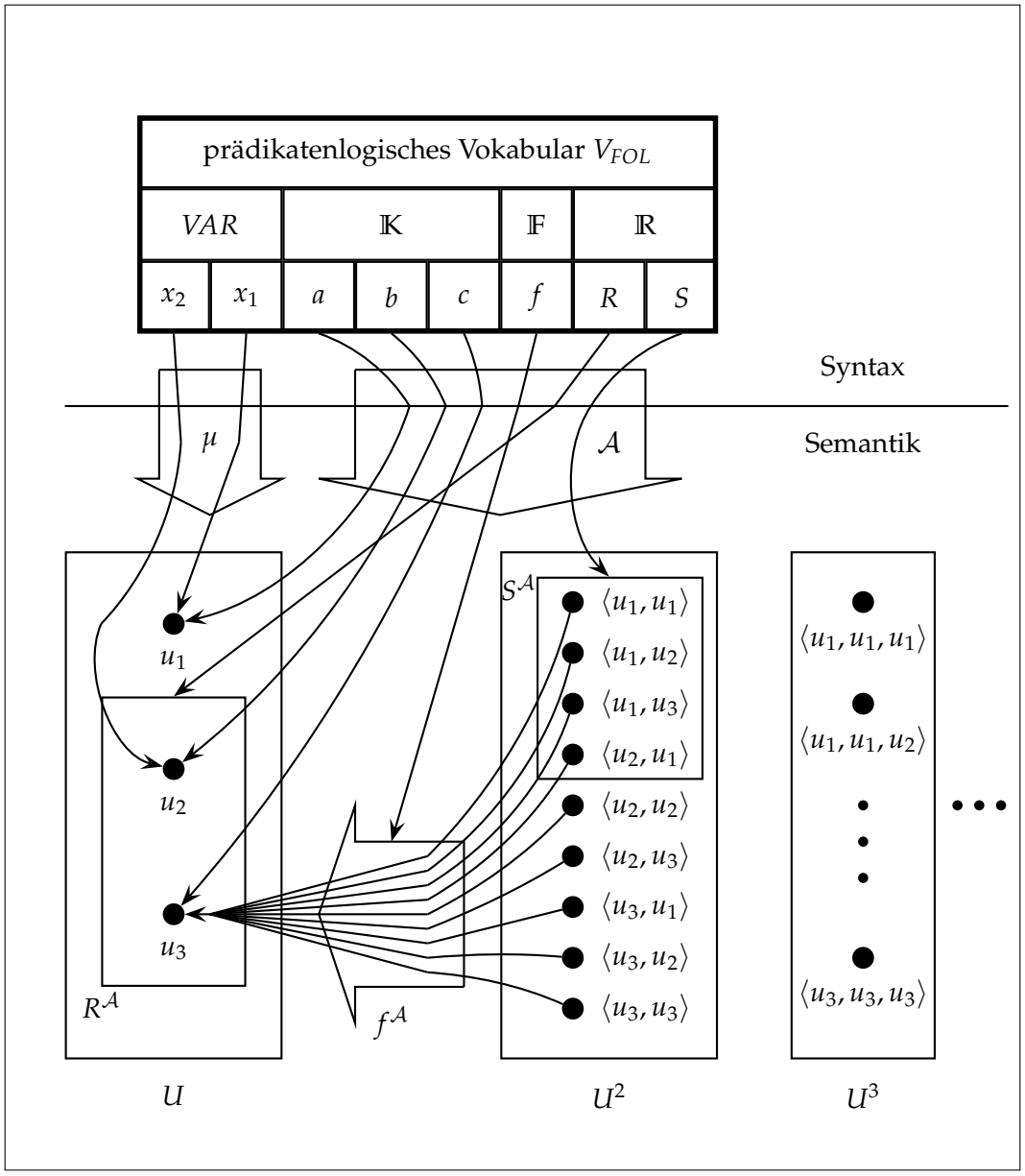

Abbildung 3.1: Schematische Darstellung des Universums von Beispiel 3.15

katenlogischen Universum modelliert. Die Klassen einer Ontologie werden durch einstellige Relationen dargestellt, wie beispielsweise die Klassen `Mensch` oder `Mann`. Es gilt zu beachten, dass in der Prädikatenlogik erster Stufe lediglich Relationen zwischen Instanzen modelliert werden können. Beziehungen zwischen Klassen lassen sich deshalb nicht direkt beschreiben, sondern werden indirekt über die Instanzen der jeweiligen Klassen festgelegt. So werden beispielsweise die hierarchischen Beziehungen zwischen den einzelnen Klassen der Beispielontologie nicht mithilfe der Relationen `istOberklasseVon` und `istUnterklasseVon` beschrieben, sondern diese Beziehungen werden direkt durch die prädikatenlogischen Formeln (3.9) bis (3.14) modelliert. Dieses Prinzip wird auch angewendet, um die Gleichheit von zwei Klassen oder Eigenschaften von Relationen auszudrücken. Ein Beispiel dafür ist Formel (3.31), die ausdrückt, dass die beiden Relationen `hatSohn` und `hatVater` invers zueinander sind. Die Beziehungen `hatSohn`, `hatVater`, `hatMutter` und `istBruderVon` werden durch binäre Relationen modelliert.

Ein weiterer Vorteil der Prädikatenlogik ist der hohe Bekanntheitsgrad dieser logischen Sprache. Die Prädikatenlogik erster Stufe ist neben der Aussagenlogik, die weit verbreitetste Logik überhaupt. Dies hat dazu geführt, dass unzählige Arbeiten über die Prädikatenlogik verfasst und ihre Eigenschaften sehr detailliert untersucht wurden.

Aufgrund des hohen Verbreitungsgrades der Prädikatenlogik, ist eine sehr große Anzahl von Anwendungsprogrammen verfügbar. So existieren zahlreiche, sehr effizient arbeitende maschinelle Theorembeweiser wie Vampire [61] oder SPASS [74]. Die meisten dieser Reasoner beinhalten graphische Benutzeroberflächen mit denen auch umfangreiche prädikatenlogische Theorien komfortabel editiert und verwaltet werden können. Als Beispiel sei auf die Prover9-Mace4 GUI[7] verwiesen.

Wir wollen jedoch anmerken, dass die Prädikatenlogik erster Stufe eine rein logische Sprache ist, das heißt es existieren beispielsweise keine Möglichkeiten um einzelne Formelmengen einer prädikatenlogischen Theorie zu Gruppen zusammenfassen zu können oder mit Kommentaren zu versehen.

Des Weiteren ist die Wiederverwendbarkeit von prädikatenlogischen Theorien sehr stark eingeschränkt, da es keinen Importmechanismus gibt. Prinzipiell ist nur das berühmte "Copy & Paste" möglich. Diese Methode kann jedoch zu Konflikten führen, beispielsweise bei der Verwendung von identischen Bezeichnern in verschiedenen prädikatenlogischen Theorien, die zusammengeführt werden sollen.

In der Prädikatenlogik ist die Nutzung von global eindeutigen Bezeichnern wie URIs für Klassen oder Relationen nicht üblich. In der Regel werden proprietäre Namen eingesetzt, die eine Wiederverwendbarkeit einzelner Klassen oder Konzepte unmöglich machen.

Abschließend sei noch erwähnt, dass die Prädikatenlogik keine speziell für die

[7] Die Prover9-Mace4 GUI ist verfügbar unter
http://www.cs.unm.edu/~mccune/prover9/gui/v05.html

Beschreibung von Ontologien entwickelte logische Sprache ist. Dies zeigt sich dadurch, dass es keine vordefinierten Komponenten für Ontologien gibt, wie beispielsweise eine elementare Vererbungsbeziehung und alles manuell definiert werden muss. Auch die Verwendung von Datentypen ist in der Prädikatenlogik nicht vorgesehen.

$$\forall x\, \texttt{Mensch}\,(x) \tag{3.1}$$

$$\forall x \Big(\texttt{Vater}\,(x) \leftrightarrow \big(\texttt{Mann}\,(x) \wedge \exists y\, (\texttt{Sohn}\,(y) \wedge \texttt{hatSohn}\,(x,y)) \big) \Big) \tag{3.2}$$

$$\begin{aligned} \forall x\, (\texttt{Sohn}\,(x) \leftrightarrow (\texttt{Mann}\,(x) \wedge \\ (\exists y_1\, (\texttt{Vater}\,(y_1) \wedge \texttt{hatVater}\,(x,y_1)) \wedge \\ \exists y_2\, (\texttt{Mutter}\,(y_2) \wedge \texttt{hatMutter}\,(x,y_2))))) \end{aligned} \tag{3.3}$$

$$\forall x \Big(\texttt{Mutter}\,(x) \leftrightarrow \big(\texttt{Frau}\,(x) \wedge \exists y\, (\texttt{Sohn}\,(y) \wedge \texttt{hatMutter}\,(y,x)) \big) \Big) \tag{3.4}$$

$$\begin{aligned} \forall x\, (\texttt{kinderreicheMutter}\,(x) \leftrightarrow (\texttt{Mutter}\,(x) \wedge \exists y_1 \exists y_2 \exists y_3\, (\\ \neg\, (y_1 \ominus y_2) \wedge \neg\, (y_1 \ominus y_3) \wedge \\ \neg\, (y_2 \ominus y_3) \wedge \texttt{Sohn}\,(y_1) \wedge \\ \texttt{Sohn}\,(y_2) \wedge \texttt{Sohn}\,(y_3) \wedge \\ \texttt{hatMutter}\,(y_1, x) \wedge \\ \texttt{hatMutter}\,(y_2, x) \wedge \\ \texttt{hatMutter}\,(y_3, x)\ )\ )\ ) \end{aligned} \tag{3.5}$$

$$\texttt{Sohn}\,(\texttt{Thomas}) \tag{3.6}$$

$$\texttt{Vater}\,(\texttt{Paul}) \tag{3.7}$$

$$\texttt{Mutter}\,(\texttt{Julia}) \tag{3.8}$$

$$\forall x \big(\texttt{Mann}\,(x) \rightarrow \texttt{Mensch}\,(x) \big) \tag{3.9}$$

$$\forall x \big(\texttt{Frau}\,(x) \rightarrow \texttt{Mensch}\,(x) \big) \tag{3.10}$$

$$\forall x \big(\texttt{Vater}\,(x) \rightarrow \texttt{Mann}\,(x) \big) \tag{3.11}$$

$$\forall x \big(\texttt{Sohn}\,(x) \rightarrow \texttt{Mann}\,(x) \big) \tag{3.12}$$

$$\forall x \big(\texttt{Mutter}\,(x) \rightarrow \texttt{Frau}\,(x) \big) \tag{3.13}$$

$$\forall x \big(\texttt{kinderreicheMutter}\,(x) \rightarrow \texttt{Mutter}\,(x) \big) \tag{3.14}$$

Abbildung 3.2: Prädikatenlogische Modellierung der Beispielontologie - Teil 1

$$\forall x \forall y \big(\texttt{hatSohn}\,(x,y) \rightarrow (\texttt{Vater}\,(x) \wedge \texttt{Sohn}\,(y)) \big) \tag{3.15}$$

$$\texttt{hatSohn}\,(\texttt{Paul},\texttt{Thomas}) \tag{3.16}$$

$$\forall x \forall y \big(\texttt{hatVater}\,(x,y) \rightarrow (\texttt{Sohn}\,(x) \wedge \texttt{Vater}\,(y)) \big) \tag{3.17}$$

$$\texttt{hatSohn}\,(\texttt{Thomas},\texttt{Paul}) \tag{3.18}$$

$$\forall x \forall y \big(\texttt{hatMutter}\,(x,y) \rightarrow (\texttt{Sohn}\,(x) \wedge \texttt{Mutter}\,(y)) \big) \tag{3.19}$$

$$\texttt{hatSohn}\,(\texttt{Thomas},\texttt{Julia}) \tag{3.20}$$

$$\forall x \forall y \big(\texttt{istBruderVon}\,(x,y) \rightarrow (\texttt{Sohn}\,(x) \wedge \texttt{Sohn}\,(y)) \big) \tag{3.21}$$

$$\forall x \big(\texttt{Mensch}\,(x) \rightarrow (\texttt{Mann}\,(x) \vee \texttt{Frau}\,(x)) \big) \tag{3.22}$$

$$\neg \exists x \big(\texttt{Mann}\,(x) \wedge \texttt{Frau}\,(x) \big) \tag{3.23}$$

$$\forall x \forall y \forall z_1 \forall z_2 \,((\texttt{istBruderVon}\,(x,y) \wedge (\texttt{hatVater}\,(x,z_1) \wedge \texttt{hatVater}\,(y,z_2))) \rightarrow (z_1 \circledminus z_2)) \tag{3.24}$$

$$\forall x \forall y \forall z_1 \forall z_2 \,((\texttt{istBruderVon}\,(x,y) \wedge (\texttt{hatMutter}\,(x,z_1) \wedge \texttt{hatMutter}\,(y,z_2))) \rightarrow (z_1 \circledminus z_2)) \tag{3.25}$$

$$\forall x \big(\texttt{Sohn}\,(x) \rightarrow \exists y \,(\texttt{Vater}\,(y) \wedge \texttt{hatVater}\,(x,y)) \big) \tag{3.26}$$

$$\forall x \forall y_1 \forall y_2 \,((\texttt{Sohn}\,(x) \wedge (\texttt{hatVater}\,(x,y_1) \wedge \texttt{hatVater}\,(x,y_2))) \rightarrow (y_1 \circledminus y_2)) \tag{3.27}$$

$$\forall x \big(\texttt{Sohn}\,(x) \rightarrow \exists y \,(\texttt{Mutter}\,(y) \wedge \texttt{hatMutter}\,(x,y)) \big) \tag{3.28}$$

$$\forall x \forall y_1 \forall y_2 \,((\texttt{Sohn}\,(x) \wedge (\texttt{hatMutter}\,(x,y_1) \wedge \texttt{hatMutter}\,(x,y_2))) \rightarrow (y_1 \circledminus y_2)) \tag{3.29}$$

$$\forall x \big(\texttt{Sohn}\,(x) \rightarrow \neg \texttt{hatVater}\,(x,x) \big) \tag{3.30}$$

$$\forall x \forall y \big(\texttt{hatSohn}\,(x,y) \leftrightarrow \texttt{hatVater}\,(y,x) \big) \tag{3.31}$$

Abbildung 3.3: Prädikatenlogische Modellierung der Beispielontologie - Teil 2

4 Common Logic

Nachdem wir im vorangegangenen Abschnitt die traditionelle Prädikatenlogik erster Stufe kennengelernt haben, wollen wir uns in diesem Kapitel dem noch sehr jungen Common Logic Framework zuwenden.
Common Logic ist das Ergebnis der Standardisierungsbemühungen des *Knowledge Interchange Formats (KIF)* [27] und der *Conceptual Graphs (CGs)* [67]. Im Jahre 2002 existierten für beide logische Sprachen Projekte mit dem Ziel, einen ANSI-Standard für den jeweiligen Repräsentationsformalismus zu entwickeln. Am 21. und 22. März 2002 fand in der Stanford University ein Treffen von Vertretern beider Projektgruppen statt. Auf diesem *Common Logic Standardization Meeting* [1] wurde entschieden, die beiden parallel laufenden Standardisierungsverfahren zu einem Projekt zu bündeln und einen gemeinsamen Standard für beide Sprachen zu entwickeln. Als neuer Projektname wurde die Bezeichnung *Common Logic (CL)* gewählt. Das Hauptziel dieses Projektes war die Entwicklung einer gemeinsamen formalen Semantik für beide Sprachen, basierend auf der Semantik der Prädikatenlogik erster Stufe. Um eine separate Entwicklung der Semantik für die beiden verschiedenen Notationen von KIF und der CGs zu vermeiden, entschied man sich dafür, eine *abstrakte Syntax* zu konstruieren und für diese die gemeinsame Semantik zu spezifizieren. Die formale Semantik der KIF-Syntax und der CG-Syntax soll anschließend unter Verwendung einer Funktion, welche die konkreten Notationen auf die abstrakten Syntaxkategorien abbildet, definiert werden [66].

Seit Oktober 2007 ist Common Logic ein offizieller und frei verfügbarer ISO Standard [48]. Dieser Standard definiert eine abstrakte Syntax in Form von sogenannten *abstrakten Syntaxkategorien* und deren formale modelltheoretische Semantik. Des Weiteren beinhaltet der Common Logic Standard drei konkrete Notationen der abstrakten Syntaxkategorien, die als *Dialekte* bezeichnet werden. Das *Common Logic Interchange Format (CLIF)* ist einer dieser Common Logic Dialekte. Er basiert auf dem Knowledge Interchange Format und besitzt eine an LISP angelehnte Syntax. Ein weiterer Bestandteil des Common Logic Standards ist das *Conceptual Graph Interchange Format (CGIF)*. Dieser Dialekt stellt eine textuelle Repräsentation der Conceptual Graphs dar. Der dritte Common Logic Dialekt trägt den Namen *eXtended Common Logic Markup Language (XCL)* und ist eine XML-Notation der abstrakten Syntaxkategorien. Common Logic ist aufgrund der abstrakten Syntaxkategorien nicht als einzelne logische Sprache, sondern eher als eine Familie von logischen Sprachen aufzufassen, welche durch die Definition neuer Common Logic Dialekte

stetig erweitert werden kann. Die *IKRIS Knowledge Language (IKL)* [40] ist ein Beispiel für einen weiteren Common Logic Dialekt, welcher nicht Teil des aktuellen Common Logic ISO Standards ist.

Dieses Kapitel ist zunächst ähnlich aufgebaut wie der vorherige Abschnitt über die Prädikatenlogik. Wir beginnen mit der Definition der Syntax von Common Logic und führen anschließend die formale Semantik von Common Logic ein. Danach analysieren wir das Universum von Common Logic. Der vierte Teil dieses Kapitels stellt die beiden Dialektarten von Common Logic, die segregierten CL-Dialekte und die unsegregierten CL-Dialekte, vor. Anschließend beschäftigen wir uns ausführlich mit den CL-Sequenznamen. In den nachfolgenden Abschnitten zeigen wir zunächst, dass Common Logic nicht kompakt ist, bevor wir die syntaktischen Freiheiten von Common Logic an einigen Beispielen demonstrieren und die Semantik der Russellschen Klasse analysieren. Mit einem Überblick über weitere Eigenschaften und offene Fragestellungen zu Common Logic wird dieses Kapitel beendet.

4.1 Syntax von Common Logic

Wie bereits weiter vorn beschrieben, wollen wir in diesem Unterkapitel die formale Syntax von Common Logic nach [48] definieren. Wir werden die abstrakten Syntaxkategorien nacheinander einführen und deren Verhältnis zu den korrespondierenden prädikatenlogischen Konstruktionen erläutern. Um in der Lage zu sein, auch konkrete Sachverhalte in Common Logic ausdrücken und Beispiele angeben zu können, werden wir außerdem eine konkrete Notation der einzelnen Syntaxkategorien vereinbaren, welche sich sehr stark an dem Common Logic Interchange Format orientiert.

4.1.1 Elementare Syntaxkategorien

Wir werden die abstrakten Syntaxkategorien von unten nach oben definieren, das heißt wir beginnen mit den elementaren Syntaxkategorien. In Common Logic sind dies die CL-Namen, die CL-Sequenznamen und die CL-Kommentare.

Definition 4.1 (CL-Name, interpretierbarer CL-Name, quoted String)
Ein *CL-Name* ist eine Zeichenfolge über einem festgelegten Alphabet. Die Menge aller CL-Namen bezeichnen wir mit N. Jeder CL-Name ist entweder ein *interpretierbarer CL-Name* oder ein *quoted String*.

□

In Common Logic sind CL-Namen im Allgemeinen elementare syntaktische Bausteine, deren innere Strukturen nicht näher spezifiziert werden. Einzelne Common

Logic Dialekte können zusätzliche Anforderungen an CL-Namen definieren. Prinzipiell kann man sich einen CL-Namen als eine Folge von Unicode-Schriftzeichen vorstellen.

Ähnlich wie CLIF, unterscheiden wir bei CL-Namen zwischen interpretierbaren CL-Namen und quoted Strings. Wir werden letztgenannte Bezeichner als Zeichenketten darstellen, welche durch die Hochkommata ’ umklammert werden.

In Common Logic werden alle Bezeichner, ausgenommen der quoted Strings, einheitlich als interpretierbare Namen behandelt. Dies ist ein großer Unterschied zur Prädikatenlogik erster Stufe. Dort werden die interpretierbaren Namen anhand ihrer Gestalt in die paarweise disjunkten Mengen der Individuenvariablen, Individuenkonstanten, Funktionskonstanten und Relationskonstanten unterteilt. Daher ist es in der Prädikatenlogik möglich, schon anhand der syntaktischen Zeichenfolge zu erkennen, welche Art von Ding ein prädikatenlogischer Name bezeichnet, also ob er beispielsweise eine Relation oder ein Individuum benennt. Solche Folgerungen sind in Common Logic nicht möglich.

Die Prädikatenlogik beinhaltet keine Bezeichner, welche mit quoted Strings vergleichbar sind. Am ehesten kann man sich quoted Strings als spezielle Konstanten vorstellen, die jedoch in allen Interpretationen identisch gedeutet werden.

Falls aus dem Kontext eindeutig ersichtlich ist, dass es sich bei Bezeichnern um CL-Namen handelt, dann werden wir diese auch als *Namen* bezeichnen. Analoge Abkürzungen verwenden wir ebenso für die anderen Syntaxkategorien.

Die CL-Sequenznamen sind eine weitere elementare Syntaxkategorie von Common Logic, welche nachfolgend formal definiert werden.

Definition 4.2 (CL-Sequenzname)
Die Menge der *CL-Sequenznamen* über einem Alphabet ist eine Menge von Zeichenketten, welche wir mit SEQ bezeichnen. Die beiden Mengen N und SEQ müssen in jedem Common Logic Dialekt disjunkt sein.

□

Einzelne Common Logic Dialekte haben die Möglichkeit zusätzliche Bedingungen für CL-Sequenznamen zu fordern. Die Sequenznamen in Common Logic werden verwendet um beliebige endliche Argumentsequenzen, beispielsweise von Relationen, zu beschreiben. In der Prädikatenlogik erster Stufe existiert keine Syntaxkategorie, welche mit den Sequenznamen vergleichbar ist.
Die Menge der CL-Namen und CL-Sequenznamen heißt *CL-Vokabular* und wird mit $V_{CL} = N \dot{\cup} SEQ$ bezeichnet.

Definition 4.3 (CL-Kommentar)
Ein *CL-Kommentar* ist eine Zeichenkette über einem Alphabet.

□

CL-Kommentare sind in der Regel optionale Bestandteile anderer Syntaxkategorien von Common Logic, mit denen zusätzliche Informationen verwaltet werden.

Ein CL-Kommentar kann dabei aus beliebigen Daten bestehen, insbesondere aus anderen Syntaxkategorien von Common Logic.
Wir verwenden für die Darstellung von CL-Kommentaren die folgende Notation:

$$(\text{cl:comment "}text\text{"}) \tag{4.1}$$

Dabei symbolisiert *text* eine beliebige Zeichenfolge, welche das Symbol " nicht beinhaltet.

Eine zu CL-Kommentaren korrespondierende Syntaxkategorie existiert in der Prädikatenlogik nicht. Dort gibt es keine Möglichkeit logische Komponenten mit Kommentaren zu versehen. Allerdings haben Kommentare in Common Logic keine formale Bedeutung, das heißt ein logischer Ausdruck mit einem CL-Kommentar ist semantisch äquivalent zu dem entsprechenden Ausdruck ohne CL-Kommentar.

4.1.2 Terme

Unter Verwendung der elementaren Syntaxkategorien werden in Common Logic die Terme definiert. Dieser Abschnitt führt die Begriffe CL-Termsequenz und CL-Term formal ein.

Definition 4.4 (CL-Term, kommentierter CL-Term)
Als einen *CL-Term* bezeichnet man in Common Logic entweder einen CL-Namen, einen CL-Funktionsterm oder einen CL-Term, welcher mit einem CL-Kommentar versehen ist. CL-Terme mit CL-Kommentaren werden auch *kommentierte CL-Terme* genannt.

□

Die Menge der CL-Terme über dem CL-Vokabular V_{CL} nennen wir $Tm(V_{CL})$. Entsprechend führen wir für die Menge der CL-Funktionsterme über V_{CL} die Notation $FTm(V_{CL})$ und für die Menge der kommentierten CL-Terme über V_{CL} die Bezeichnung $KTm(V_{CL})$ ein.
Nach Definition 4.4 gilt also $Tm(V_{CL}) = N \cup FTm(V_{CL}) \cup KTm(V_{CL})$.

Auf CL-Funktionsterme werden wir im weiteren Verlauf dieses Unterkapitels noch ausführlich eingehen, deshalb führen wir an dieser Stelle keine Notation für diese Syntaxkategorie ein.
Sei $t \in Tm(V_{CL})$ ein CL-Term, dann vereinbaren wir für einen kommentierten CL-Term, welcher aus t unter Verwendung des CL-Kommentars $(\text{cl:comment "}text\text{"})$ konstruiert wird, die nachfolgende Schreibweise.

$$(\text{cl:comment "}text\text{" } t) \tag{4.2}$$

Die Syntaxkategorie Term existiert auch in der Prädikatenlogik. Da ein prädikatenlogischer Term entweder eine Individuenkonstante, eine Individuenvariable

oder ein prädikatenlogischer Funktionsterm ist, erweitert Common Logic bezüglich der Prädikatenlogik erster Stufe diese Syntaxkategorie. Einerseits ist es in Common Logic möglich Terme mit Kommentaren zu versehen. Andererseits werden in Common Logic alle Namen als Terme zugelassen, also auch Bezeichner für Relationen und Funktionen.

Definition 4.5 (CL-Termsequenz)
Eine *CL-Termsequenz* ist eine endliche Folge von CL-Termen und CL-Sequenznamen, die auch leer sein kann.

□

CL-Termsequenzen sind Folgen, welche als Argumente von Funktionen und Relationen in Common Logic auftreten können. Solche Termfolgen existieren auch in der Prädikatenlogik der ersten Stufe. Die CL-Termsequenzen sind jedoch umfangreicher als die prädikatenlogischen Termsequenzen, da es beispielsweise keine Sequenznamen in der Prädikatenlogik gibt.

Definition 4.6 (CL-Funktionsterm)
Ein *CL-Funktionsterm* besteht aus einem CL-Term, dem sogenannten *Operator* und einer CL-Termsequenz, die als *Argumentsequenz* bezeichnet wird. Die Elemente der CL-Termsequenz heißen *Argumente* des CL-Funktionsterms.

□

Nach Definition 4.5 ist es möglich, dass die CL-Termsequenz eines CL-Funktionsterms leer ist. Deshalb muss jeder Common Logic Dialekt sicher stellen, dass sich ein CL-Funktionsterm mit einer leeren Argumentsequenz syntaktisch von seinem Operator unterscheidet.

Sei $op \in Tm(V_{CL})$ ein CL-Term und arg eine CL-Termsequenz. Für einen CL-Funktionsterm definieren wir die Syntax wie folgt:

$$(op\ arg) \tag{4.3}$$

Die Prädikatenlogik erster Stufe definiert ebenfalls die Syntaxkategorie Funktionsterm. Prinzipiell gilt, dass ein prädikatenlogischer Funktionsterm stets auch als CL-Funktionsterm angesehen werden kann. Allerdings besitzen die Funktionssymbole der Prädikatenlogik eine feste Stelligkeit, welche durch die zugehörige Signatur vorgegeben wird. Dagegen ist die Anzahl der Argumente bei Funktionssymbolen in Common Logic variabel. Weiterhin ist es in Common Logic möglich, dass als Argumente eines CL-Funktionsterms auch Funktions- und Relationsnamen auftreten können. Die Prädikatenlogik erster Stufe gestattet derartige Konstruktionen nicht.

4.1.3 Formeln

Mithilfe der CL-Terme und CL-Termsequenzen werden, ähnlich der Prädikatenlogik, komplexe Formeln konstruiert. Die formalen Definitionen der verschiedenen Formelarten von Common Logic sind Inhalt dieses Unterkapitels.

Definition 4.7 (CL-Gleichung)
Eine *CL-Gleichung* in Common Logic besteht aus zwei CL-Termen, die *Argumente* der CL-Gleichung genannt werden.

□

Seien $t_1, t_2 \in Tm(V_{CL})$ zwei CL-Terme. Zur Darstellung einer CL-Gleichung mit den Argumenten t_1 und t_2 verwenden wir die nachfolgende Notation.

$$(\mathsf{equal}\ t_1\ t_2) \tag{4.4}$$

In der Prädikatenlogik erster Stufe mit Identität ist es ebenfalls möglich Gleichungen zu formulieren. Eine prädikatenlogische Gleichung wird, analog zu Common Logic, aus zwei Termen gebildet. Jede Gleichung, die in der Prädikatenlogik formulierbar ist, ist auch eine Gleichung im Sinne von Common Logic. Aufgrund der ausdrucksstärkeren CL-Terme sind in Common Logic zusätzliche Konstruktionen möglich, wie beispielsweise Relationsnamen oder kommentierte CL-Terme als Argumente einer CL-Gleichung.

Definition 4.8 (CL-Atomformel)
Eine *CL-Atomformel* ist zusammengesetzt aus einem CL-Term, dem *Prädikat* und einer CL-Termsequenz, welche als *Argumentsequenz* bezeichnet wird. Die Elemente der Argumentsequenz werden *Argumente* der CL-Atomformel genannt.

□

Analog zu Definition 4.6 auf der vorherigen Seite muss sichergestellt sein, dass sich ein Prädikat und eine CL-Atomformel mit einer leeren Argumentsequenz syntaktisch unterscheiden.

Es sei $R \in Tm(V_{CL})$ ein CL-Term und *arg* eine CL-Termsequenz, dann definieren wir die Syntax einer CL-Atomformel wie folgt:

$$(R\ arg) \tag{4.5}$$

Auch für CL-Atomformeln existieren in der Prädikatenlogik erster Stufe Konstruktionen, welche mit CL-Atomformeln vergleichbar sind. Allerdings gestattet Common Logic auch hier mehr Freiheiten als die Prädikatenlogik. So haben die Prädikate der Prädikatenlogik eine feste Arität, während die Stelligkeit der Prädikate in Common Logic variabel ist. Weiterhin unterscheiden sich die beiden logischen Sprachen in der Komplexität ihrer Prädikate. In Common Logic sind Terme als Prädikate zugelassen. Im Gegensatz dazu ist ein Prädikat in der Prädikatenlogik stets ein Relationssymbol. Des Weiteren ist es in Common Logic erlaubt, dass CL-Sequenznamen als Argumente einer CL-Atomformel auftreten.

Definition 4.9 (CL-Atom)
Ein *CL-Atom* in Common Logic ist entweder eine CL-Gleichung oder eine CL-Atomformel.

□

Nach dieser Definition sind CL-Atome mit prädikatenlogischen Atomformeln vergleichbar. Jede Atomformel der Prädikatenlogik kann daher als ein Atom in Common Logic angesehen werden.

Definition 4.10 (Boolesche CL-Formel)
Jede *Boolesche CL-Formel* besteht aus einer festgelegten Anzahl von CL-Formeln denen ein bestimmter Typ, *Boolescher Operator* genannt, zugeordnet wird. Die einzelnen CL-Formeln werden als *Komponenten* der Booleschen CL-Formel bezeichnet. Ihre Anzahl ist abhängig von dem Typ der Booleschen CL-Formel. Jeder Common Logic Dialekt sollte zwischen den folgenden fünf Typen von Booleschen CL-Formeln unterscheiden: Konjunktion, Disjunktion, Implikation, Äquivalenz und Negation. Boolesche CL-Formeln vom Typ *Konjunktion* oder *Disjunktion* können eine beliebige Anzahl von Komponenten haben. Bei einer *Implikation* oder *Äquivalenz* besteht die Boolesche CL-Formel aus genau zwei Komponenten. *Negationen* besitzen genau eine Komponente.

□

Seien $\varphi_1, \varphi_2, \ldots, \varphi_n$ CL-Formeln, dann legen wir die Notation der fünf Typen von Booleschen CL-Formeln wie folgt fest:

$(\mathsf{and}\ \varphi_1\ \varphi_2\ \ldots\ \varphi_n)$	(Konjunktion)	(4.6)
$(\mathsf{or}\ \varphi_1\ \varphi_2\ \ldots\ \varphi_n)$	(Disjunktion)	(4.7)
$(\mathsf{if}\ \varphi_1\ \varphi_2)$	(Implikation)	(4.8)
$(\mathsf{iff}\ \varphi_1\ \varphi_2)$	(Äquivalenz)	(4.9)
$(\mathsf{not}\ \varphi_1)$	(Negation)	(4.10)

Die Prädikatenlogik erster Stufe erlaubt ebenfalls die Konstruktion von Booleschen Formeln, wobei die gleichen Junktoren zur Verfügung stehen wie bei Common Logic. Allerdings haben alle prädikatenlogischen Junktoren eine feste Anzahl von Argumenten. Dies unterscheidet die prädikatenlogische Konjunktion und die prädikatenlogische Disjunktion von ihren korrespondierenden Junktoren in Common Logic, deren Stelligkeit variabel ist. Die übrigen prädikatenlogischen Junktoren – Implikation, Äquivalenz und Negation – besitzen die gleiche Arität wie die vergleichbaren Junktoren in Common Logic.

Definition 4.11 (quantifizierte CL-Formel)
Eine *quantifizierte CL-Formel* hat einen bestimmten Typ, den sogenannten *Quantor*, eine endliche Sequenz, die *Bindungssequenz* bestehend aus CL-Namen und CL-Sequenznamen, sowie eine CL-Formel, die *Rumpf* genannt wird. Die Elemente der Bindungssequenz müssen paarweise verschieden sein. Weiterhin sollte jeder Common Logic Dialekt zwischen dem *Allquantor* und dem *Existenzquantor* als Typ einer quantifizierten CL-Formel unterscheiden.

□

Ein CL-Name oder CL-Sequenzname, welcher ein Element der Bindungssequenz ist, wird in dem zugehörigen Rumpf als *gebunden* bezeichnet. Andererseits heißt ein CL-Name oder CL-Sequenzname *frei* in einem Rumpf, falls er in diesem nicht gebunden ist.

Seien V_{CL} ein CL-Vokabular und $a_1, a_2, \ldots, a_n \in V_{CL}$. Weiterhin fordern wir, dass $a_1, a_2, \ldots, a_n$ paarweise verschieden sind. Außerdem sei φ eine CL-Formel. Eine quantifizierte CL-Formel werden wir im restlichen Teil dieser Arbeit, wie nachfolgend angegeben, darstellen.

$$(\mathsf{forall}\ (a_1\ a_2\ \ldots\ a_n)\ \varphi) \qquad \text{Allquantor} \tag{4.11}$$

$$(\mathsf{exists}\ (a_1\ a_2\ \ldots\ a_n)\ \varphi) \qquad \text{Existenzquantor} \tag{4.12}$$

Quantifizierte Formeln findet man auch in der Prädikatenlogik erster Stufe. In beiden logischen Sprachen sind quantifizierte Formeln sehr ähnlich aufgebaut. Sowohl die Prädikatenlogik als auch Common Logic unterscheiden zwischen dem Allquantor und dem Existenzquantor. Allerdings erlaubt auch hier Common Logic mehr syntaktische Freiheiten als die Prädikatenlogik. So bindet ein prädikatenlogischer Quantor genau eine Individuenvariable. Im Gegensatz dazu ist die Anzahl der Elemente der Bindungssequenz in Common Logic variabel. Des Weiteren sind in Common Logic beliebige CL-Namen und auch CL-Sequenznamen als Elemente der Bindungssequenz möglich. Dies bedeutet, dass Quantifizierungen über Relations- und Funktionsbezeichner zugelassen sind. Also existieren in Common Logic, im Gegensatz zur Prädikatenlogik erster Stufe, nicht nur Individuenvariablen, sondern auch Relations- und Funktionsvariablen.

Definition 4.12 (irreguläre CL-Formel)
Eine *irreguläre CL-Formel* ist eine Zeichenkette, die den anderen Syntaxkategorien nicht zugeordnet werden kann, da es nicht möglich ist die formale Semantik dieser Zeichenkette mithilfe von Common Logic zu definieren.

□

Die Menge der irregulären CL-Formeln bezeichnen wir mit IFm.

Irreguläre CL-Formeln sind ein Mechanismus, der die Erweiterung von Common Logic mit zusätzlichen Konstruktionen wie nichtmonotonen Operatoren ermöglicht. In der aktuellen *Semantics of Business Vocabulary and Business Rules (SBVR)* Spezifikation [56] der Object Management Group[8] werden irreguläre CL-Formeln verwendet, um modallogische Ausdrücke in Common Logic darstellen zu können.

Wir werden in dieser Arbeit die irregulären CL-Formeln nicht näher untersuchen und verzichten daher auf die Einführung einer Notation für irreguläre CL-Formeln. Eine vergleichbare Syntaxkategorie existiert in der Prädikatenlogik erster Stufe nicht.

[8] http://www.omg.org/

Definition 4.13 (CL-Formel)
Eine *CL-Formel* ist entweder ein CL-Atom, eine Boolesche CL-Formel, eine quantifizierte CL-Formel, eine CL-Formel mit einem Kommentar, *kommentierte CL-Formel* genannt, oder eine irreguläre CL-Formel.
□

Es sei V_{CL} ein CL-Vokabular. Die Menge aller CL-Formeln über V_{CL} bezeichnen wir mit $Fm(V_{CL})$.
Im Folgenden sei $\varphi \in Fm(V_{CL})$ eine CL-Formel und (cl:comment "*text*") ein CL-Kommentar. Für die kommentierte CL-Formel φ führen wir die nachfolgende Notation ein.

$$(\text{cl:comment "}text\text{"}\ \varphi) \tag{4.13}$$

Die Syntaxkategorie der CL-Formeln ist vergleichbar mit der Menge der prädikatenlogischen Formeln. Eine prädikatenlogische Formel ist nach Definition 3.4 auf Seite 32 entweder eine prädikatenlogische Atomformel, eine Boolesche Formel oder eine quantifizierte Formel. Deshalb existiert zu jeder prädikatenlogischen Formel eine korrespondierende CL-Formel. Common Logic erweitert den Formelbegriff der Prädikatenlogik erster Stufe und erlaubt zusätzlich noch irreguläre CL-Formeln und kommentierte CL-Formeln.

4.1.4 Weitere Syntaxkategorien

In Common Logic sind CL-Formeln nicht die komplexesten Syntaxkategorien, sondern es existieren noch zahlreiche weitere Konstruktionen, die wir in diesem Teil der Diplomarbeit vorstellen werden.

Definition 4.14 (CL-Import)
Ein *CL-Import* beinhaltet einen CL-Namen, welcher einen CL-Text identifiziert.
□

Ein CL-Import ermöglicht die Einbindung von externen Common Logic Inhalten und erhöht damit die Wiederverwendbarkeit von Common Logic Dokumenten.
Es sei $bez \in N$ ein CL-Name, dann vereinbaren wir folgende Notation für einen CL-Import:

$$(\text{cl:imports } bez) \tag{4.14}$$

Die Prädikatenlogik erster Stufe in ihrer klassischen Ausprägung beinhaltet keinen Mechanismus um extern definierte prädikatenlogische Formeln einzubinden.

Definition 4.15 (CL-Modul)
Ein *CL-Modul* setzt sich aus einem CL-Namen, dem *Modulnamen*, einer optionalen Menge von CL-Namen und einem CL-Text, dem *Modultext* zusammen. Die optionale Menge von CL-Namen wird als *Ausschlussmenge* bezeichnet.
□

Unter Verwendung eines CL-Moduls ist es möglich, einen CL-Text in einem speziellen Kontext zu interpretieren, welcher durch den Modulnamen spezifiziert wird. Weiterhin kann das Interpretationsuniversum mithilfe der Ausschlussmenge eingeschränkt und damit der Wirkungsbereich der Quantoren verkleinert werden.

Es seien $a, a_1, a_2, \ldots, a_n \in N$ CL-Namen und $text$ ein CL-Text. Die Syntax eines CL-Moduls legen wir wie folgt fest:

$$(\text{cl:module } a\ (\text{cl:excludes } a_1\ a_2\ \ldots\ a_n)\ text) \tag{4.15}$$

In der Prädikatenlogik der ersten Stufe existiert keine Syntaxkategorie, die mit dem CL-Modul vergleichbar ist.

Definition 4.16 (CL-Ausdruck)
Ein *CL-Ausdruck* ist entweder ein CL-Modul, eine CL-Formel, ein CL-Import oder ein CL-Text mit einem CL-Kommentar. Für die letztgenannte Komponente verwenden wir die Bezeichnung *kommentierter CL-Text*.
□

Wir wollen an dieser Stelle anmerken, dass Definition 4.16 auf der entsprechenden Spezifikation des Common Logic Standards [48, Seite 8] basiert. Die korrespondierende Darstellung des CL-Ausdrucks als Klassendiagramm in UML-Notation [48, Figure 1, Seite 10] stimmt allerdings nicht mit dieser Definition überein. Laut Klassendiagramm ist ein CL-Ausdruck entweder ein CL-Modul, eine CL-Formel, ein CL-Import oder ein CL-Kommentar.

Sei $text$ ein CL-Text und $(\text{cl:comment "}a\text{"})$ ein CL-Kommentar. Für den kommentierten CL-Text $text$ vereinbaren wir folgende Notation:

$$(\text{cl:comment "}a\text{"}\ text) \tag{4.16}$$

In der Prädikatenlogik erster Stufe existiert keine Syntaxkategorie, welche mit einem CL-Ausdruck vergleichbar wäre. Der Grund dafür ist, dass die Prädikatenlogik neben prädikatenlogischen Formeln keine zusätzlichen Konstruktionen wie beispielsweise einen CL-Import unterstützt.

Definition 4.17 (CL-Text)
Ein *CL-Text* ist eine Menge von CL-Ausdrücken. Zusätzlich kann einem CL-Text ein CL-Name zugeordnet werden, welcher diesen CL-Text eindeutig identifiziert.
□

Es seien $A_1, A_2, \ldots, A_n$ CL-Ausdrücke und $bez \in N$ ein CL-Name.
Die Syntax des CL-Textes bestehend aus den CL-Ausdrücken $A_1, A_2, \ldots, A_n$ ist wie folgt definiert:

$$(\text{cl:text } A_1\ A_2\ \ldots\ A_n) \qquad \text{CL-Text} \tag{4.17}$$

$$(\text{cl:text } bez\ A_1\ A_2\ \ldots\ A_n) \qquad \text{CL-Text mit CL-Name} \tag{4.18}$$

Der Begriff CL-Text, als die allgemeinste Syntaxkategorie, ähnelt der Bezeichnung prädikatenlogische Theorie. Allerdings werden prädikatenlogische Theorien ausschließlich aus prädikatenlogischen Formeln gebildet. Ein CL-Text kann neben CL-Formeln weitere Konstruktionen beinhalten, namentlich CL-Module, CL-Importe oder kommentierte CL-Texte. Des Weiteren unterstützt Common Logic die Möglichkeit, einen CL-Text mit einem CL-Namen zu versehen. In der Prädikatenlogik ist es nicht möglich, einer prädikatenlogischen Theorie einen Bezeichner zuzuweisen, welcher diese Theorie identifiziert.

4.1.5 Zusammenfassung

Bevor wir die Ergebnisse dieses Kapitels zusammenfassen, wollen wir noch einmal darauf hinweisen, dass die weiter oben eingeführte Notation der abstrakten Syntaxkategorien sehr stark an das Common Logic Interchange Format angelehnt ist, aber an einigen Stellen von diesem abweicht. Deshalb stellt dieser Abschnitt *keine* formale Spezifikation des Common Logic Interchange Formats dar. Vielmehr werden die abstrakten Syntaxkategorien von Common Logic formal definiert und anhand der CLIF-ähnlichen Notation etwas anschaulicher dargestellt.
Das Hauptresultat dieses Abschnitts kann wie folgt formuliert werden:

Bemerkung 4.18
Für jede prädikatenlogische Syntaxkategorie existiert eine vergleichbare Syntaxkategorie in Common Logic.

Dieses Ergebnis lässt sich mithilfe der Anmerkungen zu den einzelnen Syntaxkategorien auf den vorangegangenen Seiten begründen. Obige Bemerkung bedeutet, dass jede syntaktisch gültige prädikatenlogische Formel auch in Common Logic auf eine vergleichbare Weise ausgedrückt werden kann.

Die Tabellen 4.1 auf der nächsten Seite und 4.2 auf Seite 55 geben einen zusammenfassenden Überblick über die einzelnen abstrakten Syntaxkategorien von Common Logic und stellen diesen die korrespondierenden Syntaxkategorien der Prädikatenlogik erster Stufe gegenüber. Wie in den beiden Tabellen erkennbar ist, existieren Syntaxkategorien in Common Logic, welche nicht vergleichbar mit syntaktischen Konstruktionen der Prädikatenlogik sind. Weiterhin erweitert Common Logic einige Syntaxkategorien der Prädikatenlogik und ermöglicht dadurch mehr syntaktische Freiheiten bei der Formulierung logischer Aussagen.

Im restlichen Teil der Diplomarbeit werden wir die formale Sprache von Common Logic mit $\mathcal{FS}_{CL}(V_{CL}, IFm)$ bezeichnen.

4.2 Formale Semantik

Nachdem wir die abstrakten Syntaxkategorien von Common Logic eingeführt und deren Beziehungen zur Prädikatenlogik erster Stufe erläutert haben, werden wir in

Common Logic Syntaxkategorie	**Beispiele**	**prädikatenlogische Syntaxkategorie**	**Beispiele**
interpretierbarer CL-Name	$x, x_1, x_2, y, z,$ $f, f_1, f_2, g,$ $R, R_1, R_2, S,$ `Mann`, $c, c_1,$ `Paul`	Variablensymbol	x, x_1, x_2, y, z
		Funktionssymbol	f, f_1, f_2, g
		Relationssymbol	$R, R_1, R_2, S,$ `Mann`
		Konstantensymbol	$c, c_1,$ `Paul`
quoted String	*'s'*	—	—
CL-Sequenzname	seq, seq_1, seq_2	—	—
CL-Kommentar	(cl:comment "a")	—	—
kommentierter CL-Term	(cl:comment "a" t)	—	—
CL-Funktionsterm	$(f\ x\ c)$, $(g$ *'s'*$)$, $(f\ (f\ R)\ seq)$, $(f\ f)$	prädikatenlogischer Funktionsterm	$g\,(x_1)$, $f\,(x, c)$, $g\,(g\,(x))$
CL-Term	x, R, f, *'s'*, $(f\ (f\ R)\ seq)$	prädikatenlogischer Term	x, c, $g\,(x_1)$, $f\,(x, c)$
CL-Gleichung	(equal $x\ c$), (equal $(g\ x_1)\ R$)	prädikatenlogische Gleichung	$(x \circledcirc c)$, $(g\,(x_1) \circledcirc c)$
CL-Atomformel	(`Mann` `Paul`), $(R\ R)$, $((g\ x_1)\ x\ seq)$	prädikatenlogische Atomformel[9]	`Mann` (`Paul`), $S\,(x, g\,(x_1))$
CL-Atom	(equal $(g\ x_1)\ R$), $((g\ x_1)\ x\ seq)$	prädikatenlogische Atomformel	$(x \circledcirc c)$, $S\,(x, g\,(x_1))$

Tabelle 4.1: Die abstrakten Syntaxkategorien von Common Logic – Teil 1

[9] Wir wollen an dieser Stelle anmerken, dass eine CL-Atomformel lediglich mit einem Teil der prädikatenlogischen Atomformeln vergleichbar ist. Für diese speziellen prädikatenlogischen Konstruktionen haben wir jedoch keine eigene Bezeichnung eingeführt.

Common Logic Syntaxkategorie	**Beispiele**	**prädikatenlogische Syntaxkategorie**	**Beispiele**
Boolesche CL-Formel	$(\mathsf{not}\,(\mathsf{equal}\ x\ c))$ $(\mathsf{and}\,(R\ c)\,(R\ x))$	Boolesche Formel	$\neg\,(x \circledcirc c)$ $(R\,(c) \wedge R\,(x))$
quantifizierte CL-Formel	$(\mathsf{forall}\,(x\ y)\,(S\ x\ y))$, $(\mathsf{forall}\,(R)\,(R\ x))$	quantifizierte Formel	$\forall x \forall y\ S\,(x, y)$, $\exists x\ R\,(x)$
irreguläre CL-Formel	$\diamond\varphi$, $(\Box\varphi_1 \wedge \Box\varphi_2)$	—	—
CL-Formel	$(\mathsf{cl{:}comment}$ "a" $(R\ x))$, $(\mathsf{forall}\,(R)\,(R\ x))$	prädikatenlogische Formel	$R\,(x)$, $\neg R\,(x)$, $\exists x\ R\,(x)$
CL-Import	$(\mathsf{cl{:}imports}\ bez)$	—	—
CL-Modul	$(\mathsf{cl{:}module}\ a$ $(\mathsf{cl{:}excludes}\ a_1\ \ldots\ a_n)$ $text)$	—	—
CL-Ausdruck	$(\mathsf{cl{:}imports}\ bez)$, $(R\ x)$	—	—
CL-Text	$(\mathsf{cl{:}text}\,(R\ x)\,(R\ y))$	prädikatenlogische Theorie	

Tabelle 4.2: Die abstrakten Syntaxkategorien von Common Logic – Teil 2

diesem Abschnitt die formale Semantik der einzelnen Syntaxkategorien, basierend auf [48], definieren.

Ähnlich der Prädikatenlogik, werden die CL-Texte über mathematischen Strukturen gedeutet, welche im konkreten Fall von Common Logic als semantische CL-Strukturen bezeichnet werden.

Definition 4.19 (semantische CL-Struktur)
Eine *semantische CL-Struktur* $\mathcal{S} = \langle UR, UD, rel, fun \rangle$ besteht aus:

- Einer Menge UR, die *Referenzbereich* genannt wird.
- Einer nichtleeren Menge $UD \subseteq UR$, dem *Gegenstandsbereich*.
- Einer Abbildung $rel : UR \to \mathcal{P}(UD^*)$.
- Einer Abbildung $fun : UR \to \mathcal{F}$, wobei $\mathcal{F}$ die Menge aller totalen Funktionen f mit $f : UD^* \to UD$ bezeichnet.

□

Analog zu Definition 2.7 auf Seite 13, definieren wir die Identität zweier semantischer CL-Strukturen wie folgt:

Definition 4.20 (Identität zweier semantischer CL-Strukturen)
Seien $\mathcal{S}_1 = \langle UR_1, UD_1, rel_1, fun_1 \rangle$ und $\mathcal{S}_2 = \langle UR_2, UD_2, rel_2, fun_2 \rangle$ zwei semantische CL-Strukturen. Die beiden CL-Strukturen $\mathcal{S}_1$ und $\mathcal{S}_2$ sind identisch genau dann, wenn die nachfolgenden Bedingungen erfüllt sind.

1. $UR_1 = UR_2$,
2. $UD_1 = UD_2$,
3. $rel_1(x) = rel_2(x)$ für alle $x \in UR_1$ und
4. $fun_1(x) = fun_2(x)$ für alle $x \in UR_1$.

□

Falls die beiden semantischen CL-Strukturen $\mathcal{S}_1$ und $\mathcal{S}_2$ identisch sind, dann schreiben wir $\mathcal{S}_1 = \mathcal{S}_2$.

Um Ausdrücke von Common Logic über diesen Strukturen deuten zu können, müssen die Bezeichner der CL-Texte mit den semantischen CL-Strukturen in Beziehung gesetzt werden. Dies erfolgt durch die Erweiterung von semantischen CL-Strukturen zu CL-Interpretationen.

Definition 4.21 (CL-Interpretation)
Eine *CL-Interpretation* $I = \langle \mathcal{S}, int, seq, irr \rangle$ über dem CL-Vokabular $V_{CL} = N \dot{\cup} SEQ$ und der Menge der irregulären CL-Formeln IFm besteht aus:

- einer semantischen CL-Struktur $\mathcal{S} = \langle UR, UD, rel, fun \rangle$,

- einer Abbildung $int : N \rightarrow UR$,
- einer Abbildung $seq : SEQ \rightarrow UD^*$ und
- einer Abbildung $irr : IFm \rightarrow \{TRUE, FALSE\}$.

□

Die Menge aller CL-Interpretationen über dem CL-Vokabular V_{CL} und der Menge der irregulären CL-Formeln IFm heißt *CL-Interpretationsbereich* über V_{CL} und IFm. Wir bezeichnen diese Menge mit $\mathcal{IB}_{CL}(V_{CL}, IFm)$.

Definition 4.22 (flache CL-Interpretation)
Es sei $I \in \mathcal{IB}_{CL}(V_{CL}, IFm)$ eine CL-Interpretation über V_{CL} und IFm mit $I = \langle \mathcal{S}, int, seq, irr \rangle$ und $\mathcal{S} = \langle UR, UD, rel, fun \rangle$.
Die CL-Interpretation I heißt *flache CL-Interpretation* genau dann, wenn $UD = UR$ gilt.

□

Um in der Lage zu sein, die Erfüllbarkeitsrelation für die Syntaxkategorien von Common Logic formal definieren zu können, müssen wir zunächst die folgenden Begriffe einführen.

Definition 4.23 (CL-Modifikation)
Seien $I, K \in \mathcal{IB}_{CL}(V_{CL}, IFm)$ zwei CL-Interpretationen über V_{CL} und IFm mit $I = \langle \mathcal{S}_I, int_I, seq_I, irr_I \rangle$ und $K = \langle \mathcal{S}_K, int_K, seq_K, irr_K \rangle$. Weiterhin sei $V \subseteq V_{CL}$ eine Menge von CL-Namen und CL-Sequenznamen.
Die CL-Interpretation K ist eine *CL-Modifikation* der CL-Interpretation I bezüglich der Menge V, falls die nachfolgenden Bedingungen erfüllt sind.

1. $\mathcal{S}_K = \mathcal{S}_I$, d.h. die beiden CL-Strukturen $\mathcal{S}_K$ und $\mathcal{S}_I$ sind identisch,
2. $int_K(n) = int_I(n)$ für alle $n \in N$ mit $n \notin V$,
3. $seq_K(s) = seq_I(s)$ für alle $s \in SEQ$ mit $s \notin V$ und
4. $irr_K(\varphi) = irr_I(\varphi)$ für alle $\varphi \in IFm$.

□

Falls K eine CL-Modifikation von I bezüglich der Menge V ist, dann unterscheiden sich die beiden CL-Interpretationen I und K lediglich darin, wie die Elemente der Menge V interpretiert werden. Die Menge aller CL-Modifikationen von I bezüglich V bezeichnen wir mit $Modi^V_{CL}(I)$.

Bevor wir die Einschränkung einer CL-Interpretation definieren, werden wir zunächst den Begriff der Bildmenge formal einführen.

Definition 4.24 (Bildmenge einer Abbildung)
Es sei $f : D \rightarrow W$ eine Abbildung.
Die *Bildmenge* von f ist definiert als $BM(f) = \{w \in W \mid \exists x \in D \text{ und } f(x) = w\}$.
□

Die Bildmenge einer Abbildung ist also stets eine Teilmenge des Wertebereichs und enthält genau die Elemente des Wertebereichs, die ein Urbild bezüglich der Abbildung f besitzen.

Definition 4.25 (Bildmenge einer CL-Interpretation)
Es seien $I \in \mathcal{IB}_{CL}(V_{CL}, IFm)$ eine CL-Interpretation über V_{CL} und IFm mit $I = \langle \mathcal{S}, int, seq, irr \rangle$ und $\mathcal{S} = \langle UR, UD, rel, fun \rangle$ die dazugehörige semantische CL-Struktur.
Die *Bildmenge* der CL-Interpretation I bezüglich der Menge $E \subseteq UD$ ist definiert als:

$$\mathcal{BM}_E(I) = \bigcup_{u \in UR} BM(fun(u)|_E) \tag{4.19}$$

Dabei bezeichnet $fun(u)|_E$ die Einschränkung der Abbildung $fun(u) \in \mathcal{F}$ auf die Menge E.

□

Nach Definition der Menge $\mathcal{F}$ (Definition 4.19) gilt $\mathcal{BM}_E(I) \subseteq UD$. Die Menge $\mathcal{BM}_E(I)$ beinhaltet also alle diejenigen Elemente $x \in UD$, so dass ein $u \in UR$ existiert und x ein Urbild bezüglich der Abbildung $fun(u)|_E$ besitzt.

Im Allgemeinen gilt die Bedingung $\mathcal{BM}_E(I) \subseteq E$ nicht. Dies bedeutet, dass die Abbildungen $fun(u)|_E$ in der Regel nicht bezüglich der Menge E abgeschlossen sind. Diesen Sachverhalt werden wir durch das nachfolgende Beispiel veranschaulichen.

Beispiel 4.26
Es sei $I \in \mathcal{IB}_{CL}(V_{CL}, IFm)$ eine CL-Interpretation über V_{CL} und IFm mit $I = \langle \mathcal{S}, int, seq, irr \rangle$. Die semantische CL-Struktur $\mathcal{S} = \langle UR, UD, rel, fun \rangle$ sei wie folgt definiert:

- $UR = UD = \{a, b, c\}$,
- $rel(a) = rel(b) = rel(c) = \{\langle a \rangle, \langle a, b \rangle\}$ und
- $fun(a) = fun(b) = fun(c) = f$, wobei die Abbildung $f : UD^* \to UD$ wie folgt definiert ist:

$$f(x) = \begin{cases} a & \text{für } x = a \\ b & \text{für } x = b \\ c & \text{sonst} \end{cases}$$

□

Wir wählen nun $E = \{a, b\}$. Dann gilt $\mathcal{BM}_E(I) = \{a, b, c\}$, da beispielsweise $f(aa) = c$ und damit $\mathcal{BM}_E(I) \not\subseteq E$.

Definition 4.27 (Einschränkung einer CL-Interpretation)
Es seien $I \in \mathcal{IB}_{CL}(V_{CL}, IFm)$ eine CL-Interpretation über $V_{CL} = N \dot{\cup} SEQ$ und IFm mit $I = \langle \mathcal{S}_I, int_I, seq_I, irr_I \rangle$ und $\mathcal{S}_I = \langle UR_I, UD_I, rel_I, fun_I \rangle$ die dazugehörige semantische CL-Struktur.
Die *Einschränkung* K der CL-Interpretation I bezüglich der Menge $E \subseteq UD_I$ ist definiert als:

1. $UR_K := UR_I$
2. $UD_K := E$
3. $rel_K(u) := rel_I(u) \cap E^*$ für alle $u \in UR_I$
4. $fun_K(u) := fun_I(u)|_{E^*}$ für alle $u \in UR_I$
5. $int_K(n) := int_I(n)$ für alle $n \in N$
6. Für alle $s \in SEQ$ gilt:
   $seq_K(s) := seq_I(s)$ falls $seq_I(s) \in E^*$ und
   $seq_K(s) := \varepsilon$ falls $seq_I(s) \notin E^*$ [10]
7. $irr_K(\varphi) := irr_I(\varphi)$ für alle $\varphi \in IFm$

□

Lemma 4.28 (Wohldefiniertheit der Einschränkung einer CL-Interpretation)
Sei $I \in \mathcal{IB}_{CL}(V_{CL}, IFm)$ eine CL-Interpretation über $V_{CL} = N \dot{\cup} SEQ$ und der Menge IFm mit $I = \langle \mathcal{S}_I, int_I, seq_I, irr_I \rangle$. Weiter sei $\mathcal{S}_I = \langle UR_I, UD_I, rel_I, fun_I \rangle$ die dazugehörige semantische CL-Struktur.
Falls für die Menge $E \subseteq UD_I$ die Bedingung $\mathcal{BM}_E(I) \subseteq E$ erfüllt ist, dann ist die Einschränkung K der CL-Interpretation I bezüglich E eine wohldefinierte CL-Interpretation über V_{CL} und IFm.

Beweis:
Sei $I = \langle \mathcal{S}_I, int_I, seq_I, irr_I \rangle$ eine CL-Interpretation und $\mathcal{S}_I = \langle UR_I, UD_I, rel_I, fun_I \rangle$ eine semantische CL-Struktur. Weiter sei $E \subseteq UD_I$ eine Menge mit $\mathcal{BM}_E(I) \subseteq E$.
Wir zeigen zuerst, dass $\mathcal{S}_K = \langle UR_K, UD_K, rel_K, fun_K \rangle$ eine semantische CL-Struktur ist, das heißt wir müssen nachweisen, dass $\mathcal{S}_K = \langle UR_K, UD_K, rel_K, fun_K \rangle$ die Bedingungen der Definition 4.19 auf Seite 56 erfüllt.
Nach Definition 4.27 gilt $UR_K = UR_I$. Weiter gilt nach Konstruktion $UD_K = E$ und damit $UD_K \subseteq UR_K$, da $E \subseteq UD_I \subseteq UR_I = UR_K$. Für rel_K gilt $rel_K(u) := rel_I(u) \cap E^*$ für alle $u \in UR_I$. Also $rel_K(u) \subseteq E^*$ für alle $u \in UR_K$ und damit ist $rel_K : UR_K \to \mathcal{P}(UD_K^*)$ eine wohldefinierte Abbildung.

[10] Die hier verwendete Definition der Abbildung seq_K weicht von der offiziellen Definition ab. Nach dem Common Logic Standard [48, Seite 14] gilt $seq_K(s) = seq_I(s)$ für alle $s \in SEQ$. Diese Definition ist jedoch falsch [37]. Die obige Festlegung formuliert unser Verständnis der informalen Formulierung der Einschränkung von seq_I auf E^*.

Des Weiteren gilt nach Konstruktion aus Definition 4.27 $fun_K(u) = fun_I(u)|_{E^*}$ und damit $fun_K(u) : E^* \to UD_I$. Wir müssen nun zeigen, dass die Funktionen der Menge $\mathcal{F}$ bezüglich E abgeschlossen sind. Sei $u \in UR_K$ und $f_u = fun_K(u)$. Weiter sei $w \in E^*$ und $f_u(w) = a$. Dann gilt $a \in UD_I$ und nach 4.24 $a \in BM(f_u)$. Also $a \in \mathcal{BM}_E(I)$ und somit gilt auch $a \in E$, da $\mathcal{BM}_E(I) \subseteq E$. Also ist $fun_K : UR_K \to \mathcal{F}$, wobei $\mathcal{F}$ die Menge aller totalen Funktionen f mit $f : E^* \to E$ ist. Damit haben wir gezeigt, dass $\mathcal{S}_K = \langle UR_K, UD_K, rel_K, fun_K \rangle$ eine semantische CL-Struktur ist.
Weiterhin gilt nach Konstruktion aus Definition 4.27: $int_K : N \to UR_K$ und $seq_K : SEQ \to E^*$ sowie $irr_K : IFm \to \{TRUE, FALSE\}$.
Also ist $K = \langle \mathcal{S}_K, int_K, seq_K, irr_K \rangle$ eine CL-Interpretation über dem CL-Vokabular V_{CL} und der Menge IFm.

■

Nachfolgend definieren wir die Retraktion einer CL-Interpretation.

Definition 4.29 (Retraktion einer CL-Interpretation)
Es sei $I = \langle \mathcal{S}_I, int_I, seq_I, irr_I \rangle$ eine CL-Interpretation über $V_{CL} = N \dot\cup SEQ$ und IFm. Weiter sei $\mathcal{S}_I = \langle UR_I, UD_I, rel_I, fun_I \rangle$ die zu I gehörende semantische CL-Struktur und $M \subseteq N$ eine Menge von CL-Namen.
Die CL-Interpretation $K = \langle \mathcal{S}_K, int_K, seq_K, irr_K \rangle$ mit $\mathcal{S}_K = \langle UR_K, UD_K, rel_K, fun_K \rangle$ ist die *Retraktion* von I bezüglich M, falls K die Einschränkung von I bezüglich der Menge $E = UD_I \setminus \{int_I(v) \mid v \in M\}$ ist.

□

Die Retraktion der CL-Interpretation I bezüglich der Menge M bezeichnen wir mit $[I < M]$.

Als nächstes führen wir den Begriff der CL-Interpretationsabbildung ein. Dazu vereinbaren wir folgende Notation. Seien $s = \langle s_1, s_2, \ldots, s_n \rangle$ und $t = \langle t_1, t_2, \ldots, t_m \rangle$ zwei endliche Sequenzen, dann bezeichnet $s \bullet t = \langle s_1, s_2, \ldots, s_n, t_1, t_2, \ldots, t_m \rangle$ die durch Verkettung von s und t erzeugte Sequenz.

Definition 4.30 (CL-Interpretationsabbildung)
Es sei $V_{CL} = N \dot\cup SEQ$ ein CL-Vokabular. Weiterhin seien $I = \langle \mathcal{S}, int, seq, irr \rangle$ eine CL-Interpretation über V_{CL} und IFm sowie $\mathcal{S} = \langle UR, UD, rel, fun \rangle$ die zu I gehörende semantische CL-Struktur.
Die *CL-Interpretationsabbildung* int_I^* ist eine Abbildung, die wie folgt definiert ist:

1. Sei $'s' \in N$ ein quoted String, dann gilt: $int_I^*('s') = s$.
2. Sei $n \in N$ ein interpretierbarer CL-Name, dann gilt: $int_I^*(n) = int(n)$.
3. Sei $s \in SEQ$ ein CL-Sequenzname, dann gilt: $int_I^*(s) = seq(s)$.
4. Sei $arg = \langle a_1, a_2, \ldots, a_n \rangle$ eine CL-Termsequenz und $a_1 \in Tm(V_{CL})$ ein CL-Term, dann gilt: $int_I^*(arg) = \langle int_I^*(a_1) \rangle \bullet int_I^*(\langle a_2, \ldots, a_n \rangle)$.

5. Sei $arg = \langle a_1, a_2, a_3, \ldots, a_n \rangle$ eine CL-Termsequenz und $a_1 \in SEQ$ ein CL-Sequenzname, dann gilt: $int_I^*(arg) = int_I^*(a_1) \bullet int_I^*(\langle a_2, \ldots, a_n \rangle)$.

6. Sei $t = (op\ arg)$ ein CL-Funktionsterm mit dem Operator op und der Argumentsequenz arg , dann gilt: $int_I^*(t) = fun(int_I^*(op))(int_I^*(arg))$.

7. Sei $t = (\mathsf{cl{:}comment}\ "text"\ term)$ ein kommentierter CL-Term, dann gilt: $int_I^*(t) = int_I^*(term)$.

□

Die CL-Interpretationsabbildung int_I^* bildet jede CL-Termsequenz in eine endliche Sequenz des Referenzbereichs UR ab. Ähnlich der Beobachtung 3.9 auf Seite 33 ist die CL-Interpretationsabbildung eindeutig durch die Abbildungen int, seq und fun festgelegt.

Nachdem die Interpretation für einfache Common Logic Konstruktionen spezifiziert ist, wollen wir nun die Interpretation der komplexen Syntaxkategorien festlegen. Diese Interpretation wird unter Verwendung einer Erfüllbarkeitsrelation ähnlich der Prädikatenlogik erster Stufe definiert. Bevor wir diese Erfüllbarkeitsrelation formal einführen, vereinbaren wir die nachfolgende Definition.

Definition 4.31 (benanntes CL-Textobjekt)
Es seien $I = \langle \mathcal{S}, int, seq, irr \rangle$ eine CL-Interpretation über V_{CL} und IFm sowie $\mathcal{S} = \langle UR, UD, rel, fun \rangle$ die zu I gehörende semantische CL-Struktur.
Ein *benanntes CL-Textobjekt* $t = \langle NAME(t), TEXT(t) \rangle \in UR$ des Referenzbereichs ist ein Paar bestehend aus dem CL-Namen $NAME(t)$ und dem CL-Text $TEXT(t)$.
□

Benannte CL-Textobjekte sind Elemente des Referenzbereichs einer CL-Interpretation und symbolisieren die semantischen Gegenstücke von CL-Texten, denen ein CL-Name zur Identifizierung zugeordnet wurde.

Definition 4.32 (CL-Erfüllbarkeitsrelation)
Seien $I = \langle \mathcal{S}, int, seq, irr \rangle$ eine CL-Interpretation über V_{CL} und IFm sowie $\mathcal{S} = \langle UR, UD, rel, fun \rangle$ die zu I gehörende semantische CL-Struktur. Weiter bezeichne int_I^* die durch I eindeutig festgelegte CL-Interpretationsabbildung.
Die *CL-Erfüllbarkeitsrelation* $\models$ ist wie folgt induktiv definiert:

1. Sei $\varphi = (\mathsf{equal}\ t_1\ t_2)$ eine CL-Gleichung mit $t_1, t_2 \in Tm(V_{CL})$. Dann gilt $I \models \varphi$ genau dann, wenn $int_I^*(t_1) = int_I^*(t_2)$.

2. Sei $\varphi = (R\ arg)$ eine CL-Atomformel bestehend aus dem Prädikat R und der Argumentsequenz arg. Dann gilt $I \models \varphi$ genau dann, wenn $int_I^*(arg) \in rel(int_I^*(R))$.

3. Sei $\varphi = (\mathsf{not}\ \psi)$ eine Boolesche CL-Formel vom Typ Negation mit der Komponente $\psi \in Fm(V_{CL})$. Dann gilt $I \models \varphi$ genau dann, wenn nicht $I \models \psi$.

4. Sei $\varphi = (\mathsf{and}\ \psi_1\ \psi_2\ \ldots\ \psi_n)$ eine Boolesche CL-Formel vom Typ Konjunktion mit den Komponenten $\psi_1, \psi_2, \ldots, \psi_n \in Fm(V_{CL})$. Dann gilt $I \models \varphi$ genau dann, wenn $I \models \psi_i$ für alle $1 \leq i \leq n$.

5. Sei $\varphi = (\mathsf{and})$ eine Boolesche CL-Formel vom Typ Konjunktion ohne Komponenten. Dann gilt $I \models \varphi$.

6. Sei $\varphi = (\mathsf{or}\ \psi_1\ \psi_2\ \ldots\ \psi_n)$ eine Boolesche CL-Formel vom Typ Disjunktion mit den Komponenten $\psi_1, \psi_2, \ldots, \psi_n \in Fm(V_{CL})$. Dann gilt $I \models \varphi$ genau dann, wenn $I \models \psi_i$ für mindestens ein i mit $1 \leq i \leq n$.

7. Sei $\varphi = (\mathsf{or})$ eine Boolesche CL-Formel vom Typ Disjunktion ohne Komponenten. Dann gilt $I \not\models \varphi$.

8. Sei $\varphi = (\mathsf{if}\ \psi_1\ \psi_2)$ eine Boolesche CL-Formel vom Typ Implikation mit den Komponenten $\psi_1, \psi_2 \in Fm(V_{CL})$. Dann gilt $I \models \varphi$ genau dann, wenn $I \models (\mathsf{or}\ (\mathsf{not}\ \psi_1)\ \psi_2)$.

9. Sei $\varphi = (\mathsf{iff}\ \psi_1\ \psi_2)$ eine Boolesche CL-Formel vom Typ Äquivalenz mit den Komponenten $\psi_1, \psi_2 \in Fm(V_{CL})$. Dann gilt $I \models \varphi$ genau dann, wenn $I \models (\mathsf{if}\ \psi_1\ \psi_2)$ und $I \models (\mathsf{if}\ \psi_2\ \psi_1)$.

10. Sei $\varphi = (\mathsf{forall}\ (v_1\ v_2\ \ldots\ v_n)\ \psi)$ eine quantifizierte CL-Formel vom Typ Allquantor mit der Bindungssequenz $v_1\ v_2\ \ldots\ v_n \in V_{CL}^*$ und dem Rumpf $\psi \in Fm(V_{CL})$. Dann gilt $I \models \varphi$ genau dann, wenn für alle CL-Modifikationen $K \in Modi_{CL}^V(I)$ von I bezüglich $V = \{v_1, v_2, \ldots, v_n\}$ gilt $K \models \psi$.

11. Sei $\varphi = (\mathsf{exists}\ (v_1\ v_2\ \ldots\ v_n)\ \psi)$ eine quantifizierte CL-Formel vom Typ Existenzquantor mit der Bindungssequenz $v_1\ v_2\ \ldots\ v_n \in V_{CL}^*$ und dem Rumpf $\psi \in Fm(V_{CL})$. Dann gilt $I \models \varphi$ genau dann, wenn eine CL-Modifikation $K \in Modi_{CL}^V(I)$ von I bezüglich $V = \{v_1, v_2, \ldots, v_n\}$ existiert, so dass $K \models \psi$.

12. Sei $\varphi \in IFm$ eine irreguläre CL-Formel. Dann gilt $I \models \varphi$ genau dann, wenn $irr(\varphi) = TRUE$.

13. Sei $\varphi = (\mathsf{cl{:}comment}\ \texttt{"}text\texttt{"}\ \psi)$ eine kommentierte CL-Formel. Dann gilt $I \models \varphi$ genau dann, wenn $I \models \psi$.

14. Sei $\varphi = (\mathsf{cl{:}module}\ a\ (\mathsf{cl{:}excludes}\ a_1\ a_2\ \ldots\ a_n)\ txt)$ ein CL-Modul mit dem Modulnamen $a \in N$, der Ausschlussmenge $\{a_1, a_2, \ldots, a_n\} \subseteq N$ und dem Modultext txt. Dann gilt $I \models \varphi$ genau dann, wenn
    a) für die Retraktion von I bezüglich der Menge $M = \{a_1, a_2, \ldots, a_n\}$ gilt $[I < M] \models txt$,
    b) $rel(int_I^*(a)) = UD_{[I<M]}^*$ und

c) $\mathcal{BM}_{UD_{[I<M]}}(I) \subseteq UD_{[I<M]}$[11].

Die Menge $UD_{[I<M]}$ bezeichnet dabei den Gegenstandsbereich der Retraktion $[I < M]$.

15. Sei $\varphi = (\text{cl:imports } id)$ ein CL-Import mit dem CL-Namen $id \in N$. Dann gilt $I \models \varphi$ genau dann, wenn $I \models TEXT(int_I^*(id))$.

16. Sei $\varphi = (\text{cl:comment "}kom\text{"} \ text)$ ein kommentierter CL-Text mit dem CL-Text $text$. Dann gilt $I \models \varphi$ genau dann, wenn $I \models text$.

17. Sei $\varphi = (\text{cl:text } A_1\ A_2\ \ldots\ A_n)$ ein CL-Text bestehend aus den CL-Ausdrücken $A_1, A_2, \ldots, A_n$. Dann gilt $I \models \varphi$ genau dann, wenn $I \models A_i$ für alle $1 \leq i \leq n$.

18. Sei $\varphi = (\text{cl:text } id\ A_1\ A_2\ \ldots\ A_n)$ ein CL-Text mit dem CL-Namen $id \in N$ und den CL-Ausdrücken $A_1, A_2, \ldots, A_n$. Dann gilt $I \models \varphi$ genau dann, wenn

    a) ein benanntes CL-Textobjekt t in dem Referenzbereich UR von I existiert mit $TEXT(t) = (\text{cl:text } A_1\ A_2\ \ldots\ A_n)$ sowie $NAME(t) = id$ und

    b) $int_I^*(id) = t$.

□

Abschließend führen wir für Common Logic die semantisch fundierte Folgerungsrelation sowie die Äquivalenz von CL-Formeln und CL-Interpretationen ein.

Definition 4.33 (Modellmenge, Folgerungsrelation, Folgerungsmenge)
Sei $\varphi \in Fm(V_{CL})$ eine CL-Formel und $X \subseteq Fm(V_{CL})$ eine Menge von CL-Formeln. Weiterhin sei $I \in \mathcal{IB}_{CL}(V_{CL}, IFm)$ eine CL-Interpretation.

(a) Die *CL-Modellmenge* von φ bezeichnen wir mit $Mod_{CL}(\varphi)$ und definieren diese als $Mod_{CL}(\varphi) = \{I \in \mathcal{IB}_{CL}(V_{CL}, IFm) \mid I \models \varphi\}$.

(b) Für die CL-Formelmenge X definieren wir die *CL-Modellmenge* $Mod_{CL}(X)$ wie folgt $Mod_{CL}(X) = \{I \in \mathcal{IB}_{CL}(V_{CL}, IFm) \mid I \models X\}$[12].

(c) Die *CL-Folgerungsrelation* $\models_{CL}$ wird wie folgt definiert: $X \models_{CL} \varphi$ genau dann, wenn $Mod_{CL}(X) \subseteq Mod_{CL}(\varphi)$.

(d) Die *CL-Folgerungsmenge* von X bezeichnen wir mit $\mathcal{C}^{\models_{CL}}(X)$ und legen diese als $\mathcal{C}^{\models_{CL}}(X) = \{\varphi \in Fm(V_{CL}) \mid X \models_{CL} \varphi\}$ fest.

□

[11] Der Common Logic Standard fordert nur die ersten beiden Bedingungen. Dadurch ist die Wohldefiniertheit der Retraktion $[I < M]$, welche wir in Lemma 4.28 gezeigt haben, nicht gegeben. So ist nicht klar wie die Abbildung f aus Beispiel 4.26 für $f(aa)$ definiert ist, falls der Wertebereich von f auf die Menge $E = \{a, b\}$ eingeschränkt wird.

[12] Analog zu Definition 3.13 auf Seite 35 definieren wir für eine Menge von CL-Formeln X: $I \models X$ genau dann, wenn $I \models \varphi$ für alle $\varphi \in X$.

Definition 4.34 (semantische Äquivalenz zweier CL-Formeln)
Seien $\varphi, \psi \in Fm(V_{CL})$ zwei CL-Formeln. Dann heißen φ und ψ *semantisch äquivalent*, falls für alle CL-Interpretationen $I \in \mathcal{IB}_{CL}(V_{CL}, IFm)$ gilt $I \models \varphi$ genau dann, wenn $I \models \psi$.

□

Definition 4.35 (semantische Äquivalenz zweier CL-Interpretationen)
Seien $I, K \in \mathcal{IB}_{CL}(V_{CL}, IFm)$ zwei CL-Interpretationen. Dann heißen I und K *semantisch äquivalent*, falls für alle CL-Formeln $\varphi \in Fm(V_{CL})$ gilt $I \models \varphi$ genau dann, wenn $K \models \varphi$.

□

4.3 Charakterisierung des CL-Universums

Nachdem wir die Syntax und die Semantik von Common Logic formal definiert haben, werden wir in diesem Abschnitt das Universum von Common Logic analysieren und mit dem prädikatenlogischen Universum vergleichen. Für eine graphische Darstellung des CL-Universums wählen wir das nachfolgende Beispiel.

Beispiel 4.36
Es sei $V_{CL} = N \cup SEQ$ ein CL-Vokabular bestehend aus $N = \{R, a, 'id'\}$ und $SEQ = \{seq_1\}$. Bei den CL-Namen unterscheiden wir zwischen den interpretierbaren CL-Namen R und a sowie dem quoted String $'id'$. Weiter sei $IFm = \{\diamond\varphi\}$ die Menge der irregulären CL-Formeln.
Die CL-Interpretation $I \in \mathcal{IB}_{CL}(V_{CL}, IFm)$ über dem CL-Vokabular V_{CL} und der Menge IFm mit $I = \langle \mathcal{S}, int, seq, irr \rangle$ und die dazugehörige semantische CL-Struktur $\mathcal{S} = \langle UR, UD, rel, fun \rangle$ seien wie folgt definiert:

1. $UR = UD = \{u_1, u_2, id\}$
2. die Abbildung $rel : UR \to \mathcal{P}(UD^*)$ ist definiert als:
   - $rel(u_1) = \{\langle u_1, u_1 \rangle, \langle u_1, u_2 \rangle, \langle u_1, id \rangle, \langle u_2, u_1 \rangle\}$ und
   - $rel(u_2) = rel(id) = \{u_2, id\}$
3. die Abbildung $fun : UR \to \mathcal{F}$ mit $\mathcal{F} = \{f\}$ ist definiert als:
   - $fun(u_1) = fun(u_2) = fun(id) = f$ mit $f : UD^* \to UD$
4. die Abbildung $int : N \to UR$ ist definiert als:
   - $int(R) = u_1$, $int(a) = u_2$ und $int('id') = id$
5. die Abbildung $seq : SEQ \to UD^*$ ist definiert als:
   - $seq(seq_1) = \langle u_1, u_2 \rangle$
6. die Abbildung $irr : IFm \to \{TRUE, FALSE\}$ ist definiert als:

- $irr\,(\diamond\varphi) = TRUE$

□

Der Aufbau des CL-Universums von Beispiel 4.36 ist in Abbildung 4.1 skizziert. Im Gegensatz zur Prädikatenlogik der ersten Stufe werden die CL-Namen auf syntaktischer Ebene nicht in Relations-, Funktions- und Konstantenbezeichner sowie Variablen unterschieden. Dies bedeutet, dass jeder CL-Name sowohl eine Relation oder eine Funktion als auch ein Individuum bezeichnen kann.

Der erste Schritt bei der Interpretation von CL-Namen besteht darin, dass die Abbildung *int* jedem CL-Namen ein Element des Referenzbereichs zuordnet. Also enthält der Referenzbereich auch Elemente, die als Relationen oder Funktionen angesehen werden. Das ist ein großer Unterschied zur Prädikatenlogik, denn dort beinhaltet die Menge U keine Relations- und Funktionsbezeichner, sondern lediglich Individuen. Eine besondere Rolle spielen die quoted Strings, denn deren Abbildung *int* ist durch ihre syntaktische Gestalt fest vorgegeben und damit in jeder erfüllbaren CL-Interpretation identisch.

Das Universum von Common Logic besteht nicht, wie das prädikatenlogische Universum, aus einer einzelnen Menge, sondern es umfasst die beiden Mengen UR und UD. In Beispiel 4.36 sind die beiden Mengen identisch. Dies muss jedoch nicht bei jeder CL-Interpretation der Fall sein. Der Gegenstandsbereich UD entspricht dem Wirkungsbereich der Quantoren von Common Logic und beinhaltet diejenigen Elemente aus denen die Extensionen der Relationen und Funktionen definiert werden. Der Referenzbereich UR kann zusätzliche Elemente beinhalten.

Als zweiten Schritt ordnet die Abbildung *rel* jedem Element der Menge UR eine *relationale Extension* zu. In Abbildung 4.1 wird diese Funktion mithilfe von Strichlinien dargestellt. Dadurch wird in Common Logic zwischen einem Objekt, welches durch einen Relationsnamen bezeichnet wird, und seiner Extension unterschieden. Diese Zweistufigkeit ist in der Prädikatenlogik erster Stufe nicht vorhanden. Prädikatenlogische Relationssymbole werden direkt als relationale Extensionen interpretiert. In unserem Anschauungsbeispiel haben alle Tupel, die zu einer bestimmten Relation gehören, die gleiche Länge. Prinzipiell muss dies in Common Logic nicht der Fall sein, das heißt die Relationen können auch Tupel verschiedener Längen beinhalten. Dies spiegelt die variable Arität der Relationen von Common Logic wider.

Da jeder CL-Name auch als Funktionsbezeichner verwendet werden kann, besteht der zweite Interpretationsschritt von CL-Namen nicht nur darin, jedem CL-Namen eine relationale Extension zuzuordnen, sondern es wird jedem CL-Namen unter Verwendung der Abbildung fun zusätzlich eine funktionale Extension zugewiesen. In Abbildung 4.1 ist die Zuordnung der funktionalen Extension durch Punktlinien dargestellt. In der Prädikatenlogik erfolgt die Interpretation der Funktionsbezeichner direkt. Aufgrund der variablen Stelligkeit der Funktionssymbole in Common Logic und der Tatsache, dass die Funktionen in Common Logic total

sind, bilden die Funktionen der Menge $\mathcal{F}$ jedes mögliche Tupel über UD^* auf ein Element der Menge UD ab. Für eine bessere Übersichtlichkeit haben wir diesen Sachverhalt in Abbildung 4.1 mittels doppelliniger Pfeile zwischen den Mengen UD, UD^2, UD^3 und $\mathcal{F}$ lediglich angedeutet.

Ein weiterer Bestandteil des CL-Universums ist die Abbildung *seq*, die jeden CL-Sequenznamen auf ein Element aus UD^* abbildet. So wird in unserem Beispiel der CL-Sequenzname seq_1 auf das Tupel $\langle u_1, u_2 \rangle$ abgebildet.

Neben dem CL-Vokabular werden in einem CL-Universum auch die irregulären CL-Formeln interpretiert. Jede irreguläre CL-Formel wird als elementare Einheit angesehen und unabhängig von der inneren Struktur mithilfe der Funktion *irr* direkt in die Menge der Wahrheitswerte abgebildet. In Abbildung 4.1 haben wir dies anhand der irregulären CL-Formel $\diamond\varphi$ dargestellt. Für CL-Formeln, die keine irregulären CL-Formeln sind, ist der innere Aufbau einer CL-Formel für die Interpretation von großer Bedeutung. Dadurch unterscheiden sich irreguläre CL-Formeln semantisch sehr stark von anderen CL-Formeln.

Am Ende dieses Abschnitts wollen wir auf die Variablen in Common Logic näher eingehen. Da es in Common Logic keine Unterteilung in Relations-, Funktions- und Individuenbezeichner gibt, ist es in Common Logic möglich, auch über Relationsnamen und Funktionsnamen zu quantifizieren. Im Gegensatz dazu ist in der Prädikatenlogik erster Stufe lediglich die Quantifizierung von Individuenbezeichnern erlaubt. In Common Logic treten deshalb nicht nur Individuenvariablen, sondern auch Relations- und Funktionsvariablen auf. Allerdings existieren in Common Logic keine freien Variablen. Jeder CL-Name, welcher nicht durch einen Quantor gebunden ist, wird als CL-Name behandelt und damit als Relations-, Funktions- oder Individuenkonstante interpretiert.

4.4 Unsegregierte und segregierte CL-Dialekte

Jedes CL-Universum beinhaltet einen Referenzbereich UR und einen Gegenstandsbereich UD. Je nachdem wie diese beiden Mengen zueinander in Beziehung stehen, wird jeder Common Logic Dialekt in [48] entweder als *unsegregierter CL-Dialekt* oder als *segregierter CL-Dialekt* bezeichnet. Ziel dieses Unterkapitels ist es die Gemeinsamkeiten und Unterschiede der beiden Sprachfamilien zu beschreiben.

Definition 4.37 (unsegregierter CL-Dialekt)
Ein *unsegregierter CL-Dialekt* ist ein Common Logic Dialekt in dem jeder CL-Name ein Element des Gegenstandsbereichs bezeichnet.

□

CL-Namen, die Elemente des Gegenstandsbereichs referenzieren, werden als *CL-Objektnamen* bezeichnet. Die Menge aller CL-Objektnamen eines CL-Dialekts beschriften wir mit ON. Jeder CL-Name eines unsegregierten CL-Dialekts ist nach

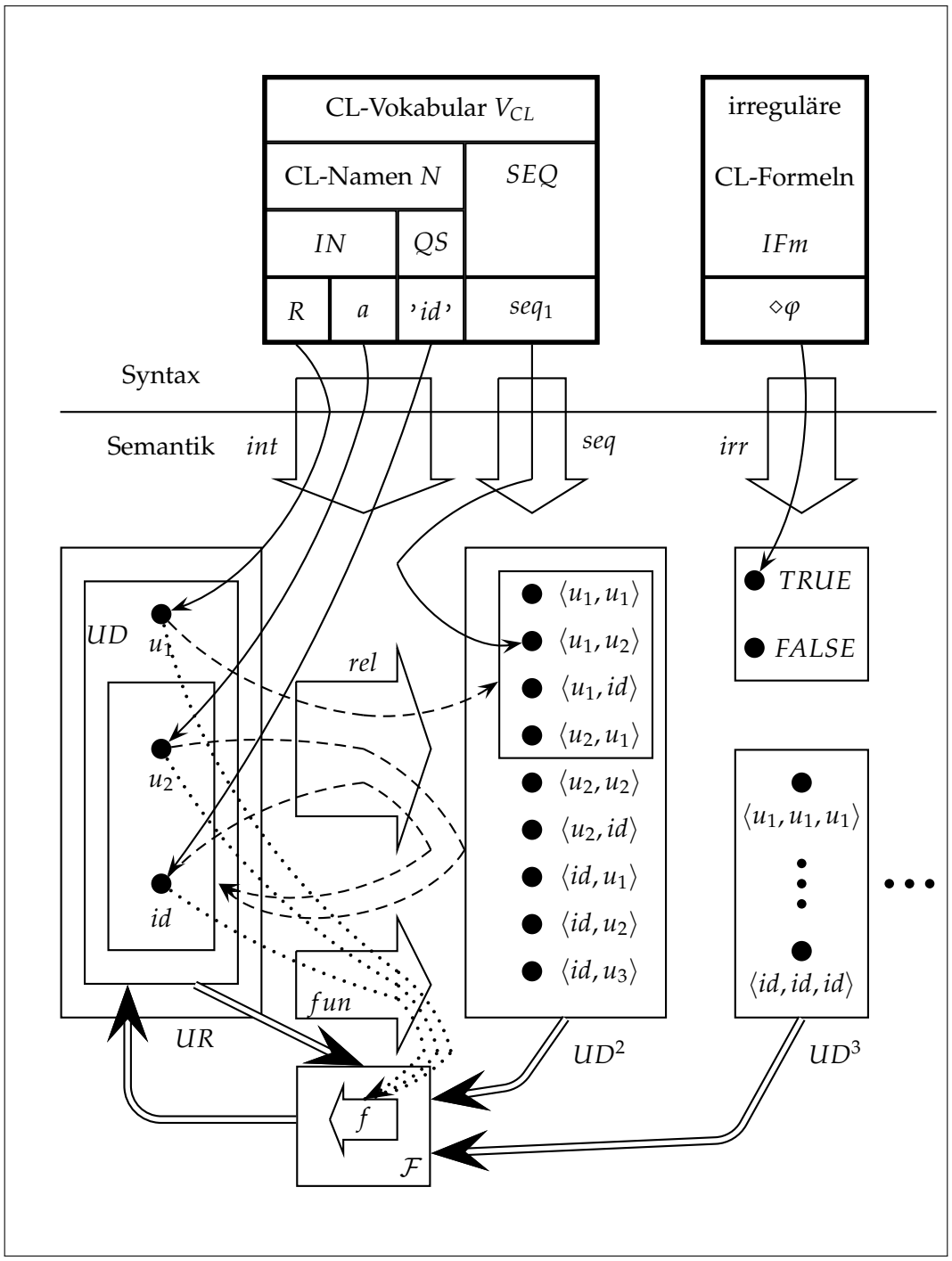

Abbildung 4.1: Schematische Darstellung des CL-Universums von Beispiel 4.36

obiger Definition ein CL-Objektname, das heißt es gilt $N = ON$. Ein Beispiel für einen unsegregierten CL-Dialekt ist das Common Logic Interchange Format (CLIF). Für unsegregierte CL-Dialekte gilt die nachfolgende Beobachtung.

Beobachtung 4.38
Sei $D = \langle \mathcal{FS}_{CL}(V_{CL}, IFm), \mathcal{IB}_{CL}(V_{CL}, IFm), \models \rangle$ *ein unsegregierter CL-Dialekt. Dann existiert für jede CL-Interpretation* $I \in \mathcal{IB}_{CL}(V_{CL}, IFm)$ *eine flache CL-Interpretation* $K \in \mathcal{IB}_{CL}(V_{CL}, IFm)$ *die zu I äquivalent ist.*

Beweis:
Wir wollen hier nur die grundlegende Beweisidee skizzieren.
Es sei $D = \langle \mathcal{FS}_{CL}(V_{CL}, IFm), \mathcal{IB}_{CL}(V_{CL}, IFm), \models \rangle$ ein unsegregierter CL-Dialekt. Weiterhin sei $I \in \mathcal{IB}_{CL}(V_{CL}, IFm)$ eine CL-Interpretation über V_{CL} und IFm des CL-Dialekts D mit $I = \langle \mathcal{S}_I, int_I, seq_I, irr_I \rangle$ und $\mathcal{S}_I = \langle UR_I, UD_I, rel_I, fun_I \rangle$.
Die flache CL-Interpretation $K \in \mathcal{IB}_{CL}(V_{CL}, IFm)$ über V_{CL} und IFm sei definiert als $K = \langle \mathcal{S}_K, int_I, seq_I, irr_I \rangle$ mit $\mathcal{S}_K = \langle UD_I, UD_I, rel_I|_{UD_I}, fun_I|_{UD_I} \rangle$. Da D ein unsegregierter CL-Dialekt ist, gilt $int_I : N \to UD_I$. Also ist K wohldefiniert.
Es ist leicht nachprüfbar, dass I und K äquivalent sind.

■

Definition 4.39 (segregierter CL-Dialekt)
Ein *segregierter CL-Dialekt* ist ein Common Logic Dialekt, welcher CL-Namen beinhaltet, die Elemente außerhalb des Gegenstandsbereichs bezeichnen.

□

CL-Namen, die Elemente außerhalb des Gegenstandsbereichs referenzieren, werden als *CL-Referenznamen* bezeichnet. Für die Menge aller CL-Referenznamen eines CL-Dialekts führen wir die Bezeichnung RN ein. Die CL-Namen eines segregierten CL-Dialekts lassen sich anhand der Elemente, die sie bezeichnen, in zwei disjunkte Mengen unterteilen, so dass $N = ON \cup RN$ gilt.

Der Common Logic Standard definiert für segregierte CL-Dialekte die nachfolgenden syntaktischen Bedingungen.

1. Jeder CL-Name eines segregierten CL-Dialekts ist entweder ein CL-Referenzname oder ein CL-Objektname. Das heißt für alle $l \in N$ gilt $l \in ON$ oder $l \in RN$ und $l \notin ON \cap RN$.

2. Kein CL-Referenzname darf als Argument eines CL-Funktionsterms, einer CL-Gleichung oder einer CL-Atomformel auftreten.

3. CL-Referenznamen dürfen nicht durch Quantoren gebunden werden.

Aufgrund dieser zusätzlichen syntaktischen Vorschriften erlaubt ein unsegregierter CL-Dialekt mehr syntaktische Freiheiten als der vergleichbare segregierte CL-Dialekt.

Segregierte CL-Dialekte und unsegregierte CL-Dialekte unterscheiden sich nicht nur syntaktisch, sondern auch semantisch. Wir betrachten die folgende CL-Formel, um die semantischen Unterschiede zu verdeutlichen.

$$\varphi = (\mathsf{and}\ (\mathsf{forall}\ (X)\ (X\ a))\ (\mathsf{not}\ (Y\ a))) \tag{4.20}$$

Lemma 4.40
Die CL-Formel φ von (4.20) ist in jedem unsegregierten CL-Dialekt unerfüllbar.

Beweis:
Im Folgenden bezeichne V_φ das Vokabular der CL-Formel φ von (4.20).
Es sei $D = \langle \mathcal{FS}_{CL}(V_{CL}, IFm), \mathcal{IB}_{CL}(V_{CL}, IFm), \models \rangle$ ein unsegregierter CL-Dialekt mit $V_\varphi \subseteq V_{CL}$. Weiter sei $I \in \mathcal{IB}_{CL}(V_{CL}, IFm)$ eine CL-Interpretation über V_{CL} und IFm mit $I = \langle \mathcal{S}_I, int_I, seq_I, irr_I \rangle$ und $\mathcal{S}_I = \langle UR_I, UD_I, rel_I, fun_I \rangle$.
Es gilt $I \models \varphi$ genau dann, wenn
$I \models (\mathsf{forall}\ (X)\ (X\ a))$ und $I \models (\mathsf{not}\ (Y\ a))$ nach Definition 4.32 (4.)
gdw. für alle CL-Modifikationen $K \in Modi^V_{CL}(I)$ von I bezüglich $V = \{X\}$ gilt $K \models (X\ a)$ und nicht $I \models (Y\ a)$ nach Definition 4.32 (10.) und (3.)
gdw. für alle CL-Modifikationen $K \in Modi^V_{CL}(I)$ von I bezüglich $V = \{X\}$ gilt $int^*_K(a) \in rel_K(int^*_K(X))$ und $int^*_I(a) \notin rel_I(int^*_I(Y))$ nach Definition 4.32 (2.)
gdw. für alle CL-Modifikationen $K \in Modi^V_{CL}(I)$ von I bezüglich $V = \{X\}$ gilt $int_K(a) \in rel_K(int_K(X))$ und $int_I(a) \notin rel_I(int_I(Y))$ nach Definition 4.30 (1.).
Da D ein unsegregierter CL-Dialekt ist, gilt $int_I(Y) \in UD_I$. Also existiert eine CL-Modifikation $K \in Modi^V_{CL}(I)$ von I bezüglich $V = \{X\}$ mit $int_K(X) = int_K(Y) = int_I(Y)$ und $int_K(a) = int_I(a)$ sowie $rel_K(u) = rel_I(u)$ für alle $u \in UR_K = UR_I$ nach Definition 4.23. Für diese CL-Modifikation K gilt: $K \models \varphi$ genau dann, wenn $int_I(a) \in rel_I(int_I(Y))$ und $int_I(a) \notin rel_I(int_I(Y))$. Diese Bedingung ist nicht erfüllbar. Also ist φ in D unerfüllbar.

■

Für segregierte CL-Dialekte gilt jedoch das folgende Lemma.

Lemma 4.41
Die CL-Formel φ von (4.20) ist in einem segregierten CL-Dialekt erfüllbar, falls Y ein CL-Referenzname ist.

Beweis:
Im Folgenden bezeichne V_φ das Vokabular der CL-Formel φ von (4.20).
Es sei $D = \langle \mathcal{FS}_{CL}(V_{CL}, IFm), \mathcal{IB}_{CL}(V_{CL}, IFm), \models \rangle$ ein segregierter CL-Dialekt mit $V_\varphi \subseteq V_{CL}$ und $Y \in RN$ ein CL-Referenzname. Weiter sei $I \in \mathcal{IB}_{CL}(V_{CL}, IFm)$ eine CL-Interpretation über V_{CL} und IFm mit $I = \langle \mathcal{S}_I, int_I, seq_I, irr_I \rangle$ und $\mathcal{S}_I = \langle UR_I, UD_I, rel_I, fun_I \rangle$ für die gilt:

1. $UR_I = \left\{ \widetilde{Y}, \widetilde{X}, \widetilde{a} \right\}$,

2. $UD_I = \left\{\widetilde{X}, \widetilde{a}\right\}$,

3. $rel_I\left(\widetilde{X}\right) = rel_I\left(\widetilde{a}\right) = \left\{\widetilde{a}\right\}$, $rel_I\left(\widetilde{Y}\right) = \emptyset$ und

4. $int_I(X) = \widetilde{X}$, $int_I(a) = \widetilde{a}$, $int_I(Y) = \widetilde{Y}$.

Für jede CL-Modifikation $K \in Modi_{CL}^{V}(I)$ von I bezüglich $V = \{X\}$ gilt $int_K(a) \in rel_K(int_K(X))$ und damit $K \models (X\ a)$. Außerdem gilt $I \models (\mathsf{not}(Y\ a))$. Also $I \models \varphi$. ■

Die unterschiedliche Semantik der unsegregierten und segregierten CL-Dialekte wird durch die CL-Referenznamen verursacht. Während in unsegregierten CL-Dialekten nur CL-Objektnamen auftreten, existieren in segregierten CL-Dialekten zusätzlich CL-Referenznamen. Weil CL-Referenznamen Elemente außerhalb des Gegenstandsbereichs referenzieren, liegen diese außerhalb des Wirkungsbereichs der Quantoren. In unsegregierten CL-Dialekten bezeichnen alle CL-Namen Elemente des Gegenstandsbereichs. Dadurch wird stets über sämtliche CL-Namen quantifiziert.

Um CL-Formeln innerhalb der Common Logic Sprachfamilie zwischen segregierten und unsegregierten CL-Dialekten übertragen und austauschen zu können, spezifiziert der Common Logic Standard [48] in Kapitel 6.6.1 eine Übersetzung zwischen diesen beiden Dialektarten.

4.5 CL-Sequenznamen

In Kapitel 4.1 haben wir die CL-Sequenznamen als mögliche Argumente von CL-Funktionstermen oder CL-Atomformeln kennengelernt. Ein CL-Sequenzname, der in einer Argumentsequenz enthalten ist, symbolisiert eine endliche Sequenz von Argumenten beliebiger Länge. Aus diesem Grund werden in Common Logic die CL-Sequenznamen verwendet, um Aussagen über Funktionen und Relationen formulieren zu können, die eine variable Anzahl von Argumenten berücksichtigen. In der Prädikatenlogik erster Stufe sind derartige Ausdrücke nicht möglich, da prädikatenlogische Funktionen und Relationen eine feste Stelligkeit besitzen.

4.5.1 Allquantifizierte CL-Sequenznamen

CL-Sequenznamen treten nicht nur als Argumente von CL-Funktionstermen oder CL-Atomformeln auf, sondern sie können in Common Logic auch durch einen Quantor gebunden werden. Wir wollen in diesem Abschnitt die Semantik von allquantifizierten CL-Sequenznamen näher untersuchen. Dazu betrachten wir zunächst ein Beispiel.

Beispiel 4.42 (allquantifizierter CL-Sequenzname)
Seien $\mathtt{R}_1, \mathtt{R}_2 \in N$ zwei CL-Namen. Wir wollen ausdrücken, dass die relationalen Extensionen der beiden Objekte, welche $\mathtt{R}_1$ und $\mathtt{R}_2$ referenzieren, identisch sind. Dieser Sachverhalt wird in Common Logic unter Verwendung des CL-Sequenznamens $s \in SEQ$ wie folgt formuliert:

$$\varphi = (\mathsf{forall}\ (s)\ (\mathsf{iff}\ (\mathtt{R}_1\ s)\ (\mathtt{R}_2\ s))) \tag{4.21}$$

□

Nun analysieren wir die Semantik der CL-Formel φ aus (4.21).
Sei V_φ das Vokabular der CL-Formel φ. Weiter sei $I \in \mathcal{IB}_{CL}(V_{CL}, IFm)$ eine CL-Interpretation über V_{CL} und IFm mit $I = \langle \mathcal{S}_I, int_I, seq_I, irr_I \rangle$, der semantischen CL-Struktur $\mathcal{S}_I = \langle UR_I, UD_I, rel_I, fun_I \rangle$ und $V_\varphi \subseteq V_{CL}$.
Es gilt $I \models \varphi$ genau dann, wenn
für alle CL-Modifikationen $K \in Modi^V_{CL}(I)$ von I bezüglich $V = \{s\}$ gilt $K \models (\mathsf{iff}\ (\mathtt{R}_1\ s)\ (\mathtt{R}_2\ s))$, nach Definition 4.32
gdw. für alle CL-Modifikationen $K \in Modi^V_{CL}(I)$ von I bezüglich $V = \{s\}$ gilt $K \models (\mathsf{if}\ (\mathtt{R}_1\ s)\ (\mathtt{R}_2\ s))$ und $K \models (\mathsf{if}\ (\mathtt{R}_2\ s)\ (\mathtt{R}_1\ s))$, nach Definition 4.32
gdw. für alle CL-Modifikationen $K \in Modi^V_{CL}(I)$ von I bezüglich $V = \{s\}$ gilt $K \models (\mathsf{or}\ (\mathsf{not}\ (\mathtt{R}_1\ s))\ (\mathtt{R}_2\ s))$ und $K \models (\mathsf{or}\ (\mathsf{not}\ (\mathtt{R}_2\ s))\ (\mathtt{R}_1\ s))$, nach Definition 4.32
gdw. für alle CL-Modifikationen $K \in Modi^V_{CL}(I)$ von I bezüglich $V = \{s\}$ gilt
(1) $K \models (\mathsf{not}\ (\mathtt{R}_1\ s))$ oder $K \models (\mathtt{R}_2\ s)$ und
(2) $K \models (\mathsf{not}\ (\mathtt{R}_2\ s))$ oder $K \models (\mathtt{R}_1\ s)$, nach Definition 4.32
gdw. für alle CL-Modifikationen $K \in Modi^V_{CL}(I)$ von I bezüglich $V = \{s\}$ gilt
(1) $int^*_K(s) \notin rel_K(int^*_K(\mathtt{R}_1))$ oder $int^*_K(s) \in rel_K(int^*_K(\mathtt{R}_2))$ und
(2) $int^*_K(s) \notin rel_K(int^*_K(\mathtt{R}_2))$ oder $int^*_K(s) \in rel_K(int^*_K(\mathtt{R}_1))$, nach Definition 4.32
gdw. für alle CL-Modifikationen $K \in Modi^V_{CL}(I)$ von I bezüglich $V = \{s\}$ gilt
(1) $seq_K(s) \notin rel_I(int_I(\mathtt{R}_1))$ und $seq_K(s) \notin rel_I(int_I(\mathtt{R}_2))$ oder
(2) $seq_K(s) \in rel_I(int_I(\mathtt{R}_2))$ und $seq_K(s) \in rel_I(int_I(\mathtt{R}_1))$.
Da $seq_K(s) \in UD^*_I$ und die obere Bedingung für alle CL-Modifikationen $K \in Modi^V_{CL}(I)$ von I bezüglich $V = \{s\}$ erfüllt sein muss, ist die CL-Formel φ semantisch äquivalent zu der CL-Formel in Abbildung 4.2 auf der nächsten Seite. Mithilfe von CL-Sequenznamen, die durch einen Allquantor gebunden werden, lassen sich in Common Logic unendliche Konjunktionen ausdrücken.
Dieses Resultat halten wir in der nachfolgenden Beobachtung fest.

Beobachtung 4.43
Eine quantifizierte CL-Formel φ vom Typ Allquantor, welche einen CL-Sequenznamen bindet, ist semantisch äquivalent zu der unendlichen Konjunktion über alle diejenigen CL-Formeln die man erhält, wenn der CL-Sequenzname in φ durch eine beliebige endliche Sequenz von CL-Namen, welche bisher nicht in φ enthalten sind, ersetzt wird.

Beweisidee:
Es sei $\varphi = (\mathsf{forall}\ (s)\ \psi)$ eine quantifizierte CL-Formel vom Typ Allquantor mit $s \in$

```
(and
                        (iff
                            (R1 )
                            (R2 )
                          )
    (forall (x1)        (iff
                            (R1 x1)
                            (R2 x1)
                        ))
    (forall (x1 x2)     (iff
                            (R1 x1 x2)
                            (R2 x1 x2)
                        ))
    (forall (x1 x2 x3)  (iff
                            (R1 x1 x2 x3)
                            (R2 x1 x2 x3)
                        ))
    ...
    (forall (x1 x2 ... xn) (iff
                            (R1 x1 x2 ... xn)
                            (R2 x1 x2 ... xn)
                        ))
  )
```

Abbildung 4.2: Gleichheit der relationalen Extensionen von $\mathtt{R}_1$ und $\mathtt{R}_2$

SEQ und $\psi \in Fm(V_{CL})$. Weiterhin bezeichne V_φ das Vokabular von φ und $tr[x]\,(\psi)$ die CL-Formel, welche man erhält, wenn alle Vorkommen von s in ψ durch die endliche Sequenz x von CL-Namen ersetzt werden. Sei $I \in \mathcal{IB}_{CL}(V_{CL}, IFm)$ eine CL-Interpretation über V_{CL} und IFm mit $V_\varphi \subseteq V_{CL}$, $I = \langle \mathcal{S}_I, int_I, seq_I, irr_I \rangle$ und $\mathcal{S}_I = \langle UR_I, UD_I, rel_I, fun_I \rangle$. Weiter sei $V_x = \{x_1, x_2, \ldots, x_n\} \subseteq N$ eine endliche Menge von CL-Namen mit $V_x \subseteq V_{CL}$ und $V_x \cap V_\varphi = \emptyset$.
Es gilt $I \models \varphi$ genau dann, wenn
für alle CL-Modifikationen $K \in Modi^V_{CL}\,(I)$ von I bezüglich $V = \{s\}$ gilt $K \models \psi$, nach Definition 4.32
Da $seq_K\,(s) \in UD^*_I$ folgt: $I \models \varphi$ genau dann, wenn
(1) für alle CL-Modifikationen $K \in Modi^V_{CL}\,(I)$ von I bezüglich $V = \{s\}$ mit $seq_K\,(s) \in UD^0_I$ gilt $K \models \psi$ und
(2) für alle CL-Modifikationen $K \in Modi^V_{CL}\,(I)$ von I bezüglich $V = \{s\}$ mit $seq_K\,(s) \in UD_I$ gilt $K \models \psi$ und
(3) für alle CL-Modifikationen $K \in Modi^V_{CL}\,(I)$ von I bezüglich $V = \{s\}$ mit $seq_K\,(s) \in UD^2_I$ gilt $K \models \psi$ usw.
Per Induktion über den Formelaufbau von ψ lässt sich zeigen, dass $I \models \varphi$ genau dann, wenn
(1) für alle CL-Modifikationen $K \in Modi^V_{CL}\,(I)$ von I bezüglich $V = \emptyset$ gilt $K \models tr[\varepsilon]\,(\psi)$ und
(2) für alle CL-Modifikationen $K \in Modi^V_{CL}\,(I)$ von I bezüglich $V = \{x_1\}$ gilt $K \models tr[x_1]\,(\psi)$ und
(3) für alle CL-Modifikationen $K \in Modi^V_{CL}\,(I)$ von I bezüglich $V = \{x_1, x_2\}$ gilt $K \models tr[x_1\ x_2]\,(\psi)$ usw.
Das heißt $I \models \varphi$ genau dann, wenn für alle CL-Modifikationen $K \in Modi^V_{CL}\,(I)$ von I bezüglich $V = \{x_1, x_2, \ldots, x_n\}$ mit $n \in \mathbb{N}$ gilt $K \models tr[x_1\ x_2\ \ldots\ x_n]\,(\psi)$.
Also $I \models \varphi$ genau dann, wenn
(1) $I \models tr[\varepsilon]\,(\psi)$ und
(2) $I \models (\mathsf{forall}\,(x_1)\,tr[x_1]\,(\psi))$ und
(3) $I \models (\mathsf{forall}\,(x_1\ x_2)\,tr[x_1\ x_2]\,(\psi))$ usw.
Damit gilt $I \models \varphi$ genau dann, wenn
$I \models (\mathsf{forall}\,(x_1\ x_2\ \ldots\ x_n)\,tr[x_1\ x_2\ \ldots\ x_n]\,(\psi))$ für alle $n \in \mathbb{N}$.

■

Dieser Sachverhalt ist der Common Logic Gemeinschaft bekannt und wird auch im Common Logic Standard [48, Seite 18] diskutiert. Da in der Prädikatenlogik der ersten Stufe keine unendlichen Konjunktionen formuliert werden können, ist die Ausdrucksstärke von Common Logic größer als die der Prädikatenlogik erster Stufe.

4.5.2 Existenzquantifizierte CL-Sequenznamen

Da CL-Sequenznamen in Common Logic nicht nur durch Allquantoren, sondern auch durch Existenzquantoren gebunden werden können, wollen wir nun die Semantik von existenzquantifizierten CL-Sequenznamen analysieren. Analog zu dem vorangegangenen Abschnitt betrachten wir auch hier zunächst ein Beispiel.

Beispiel 4.44 (existenzquantifizierter CL-Sequenzname)
Seien $\mathtt{R}_1, \mathtt{R}_2 \in N$ zwei CL-Namen. Um auszudrücken, dass die Relationen, welche die CL-Namen $\mathtt{R}_1$ und $\mathtt{R}_2$ bezeichnen, nicht disjunkt sind, verwenden wir einen CL-Sequenznamen $s \in SEQ$. Der Sachverhalt lässt sich in Common Logic auf folgende Weise ausdrücken:

$$\varphi = (\mathsf{exists}\ (s)\ (\mathsf{and}\ (\mathtt{R}_1\ s)\ (\mathtt{R}_2\ s))) \tag{4.22}$$

□

Die Semantik der CL-Formel φ aus (4.22) ist folgenderweise definiert:
Sei V_φ das Vokabular der CL-Formel φ. Weiter sei $I \in \mathcal{IB}_{CL}(V_{CL}, IFm)$ eine CL-Interpretation über V_{CL} und IFm mit $I = \langle \mathcal{S}_I, int_I, seq_I, irr_I \rangle$, der semantischen CL-Struktur $\mathcal{S}_I = \langle UR_I, UD_I, rel_I, fun_I \rangle$ und $V_\varphi \subseteq V_{CL}$.
Es gilt $I \models \varphi$ genau dann, wenn
eine CL-Modifikation $K \in Modi^V_{CL}(I)$ von I bezüglich $V = \{s\}$ existiert mit
$K \models (\mathsf{and}\ (\mathtt{R}_1\ s)\ (\mathtt{R}_2\ s))$, nach Definition (4.32)
gdw. eine CL-Modifikation $K \in Modi^V_{CL}(I)$ von I bezüglich $V = \{s\}$ existiert mit $K \models (\mathtt{R}_1\ s)$ und $K \models (\mathtt{R}_2\ s)$, nach Definition (4.32)
gdw. eine CL-Modifikation $K \in Modi^V_{CL}(I)$ von I bezüglich $V = \{s\}$ existiert mit $int^*_K(s) \in rel_K(int^*_K(\mathtt{R}_1))$ und $int^*_K(s) \in rel_K(int^*_K(\mathtt{R}_2))$, nach Definition (4.32)
gdw. eine CL-Modifikation $K \in Modi^V_{CL}(I)$ von I bezüglich $V = \{s\}$ existiert mit $seq_K(s) \in rel_I(int_I(\mathtt{R}_1))$ und $seq_K(s) \in rel_I(int_I(\mathtt{R}_2))$.
Aufgrund der Definition von seq_K mit $seq_K(s) \in UD^*_I$ und der geforderten Existenz einer CL-Modifikation $K \in Modi^V_{CL}(I)$ von I bezüglich $V = \{s\}$ mit $seq_K(s) \in rel_I(int_I(\mathtt{R}_1))$ und $seq_K(s) \in rel_I(int_I(\mathtt{R}_2))$ ist die CL-Formel φ semantisch äquivalent zu der CL-Formel in Abbildung 4.3 auf der nächsten Seite. Unter Verwendung von existenzquantifizierten CL-Sequenznamen lassen sich in Common Logic unendliche Disjunktionen formulieren.
Allgemein gilt die folgende Beobachtung:

Beobachtung 4.45
Eine quantifizierte CL-Formel φ vom Typ Existenzquantor, die einen CL-Sequenznamen bindet, ist semantisch äquivalent zu der unendlichen Disjunktion über alle diejenigen CL-Formeln die man erhält, wenn der CL-Sequenzname in φ durch eine beliebige endliche Sequenz von CL-Namen, welche bisher nicht in φ enthalten sind, ersetzt wird.

Beweisidee:
Es sei $\varphi = (\mathsf{exists}\ (s)\ \psi)$ eine quantifizierte CL-Formel vom Typ Existenzquantor

```
(or
                        (and
                            (R1 )
                            (R2 )
                          )
    (exists (x1)        (and
                            (R1 x1)
                            (R2 x1)
                          ))
    (exists (x1 x2)     (and
                            (R1 x1 x2)
                            (R2 x1 x2)
                          ))
    (exists (x1 x2 x3)  (and
                            (R1 x1 x2 x3)
                            (R2 x1 x2 x3)
                          ))
    ...
    (exists (x1 x2 ... xn) (and
                            (R1 x1 x2 ... xn)
                            (R2 x1 x2 ... xn)
                          ))
  )
```

Abbildung 4.3: Die relationalen Extensionen von $\mathtt{R}_1$ und $\mathtt{R}_2$ sind nicht disjunkt.

mit $s \in SEQ$ und $\psi \in Fm(V_{CL})$. Weiterhin bezeichne V_φ das Vokabular von φ und $tr[x]\,(\psi)$ die CL-Formel, welche man erhält, wenn alle Vorkommen von s in ψ durch die endliche Sequenz x von CL-Namen ersetzt werden. Sei $I \in \mathcal{IB}_{CL}(V_{CL}, IFm)$ eine CL-Interpretation über V_{CL} und IFm mit $V_\varphi \subseteq V_{CL}$, $I = \langle \mathcal{S}_I, int_I, seq_I, irr_I \rangle$ und $\mathcal{S}_I = \langle UR_I, UD_I, rel_I, fun_I \rangle$. Weiter sei $V_x = \{x_1, x_2, \ldots, x_n\} \subseteq N$ eine endliche Menge von CL-Namen mit $V_x \subseteq V_{CL}$ und $V_x \cap V_\varphi = \emptyset$.
Es gilt $I \models \varphi$ genau dann, wenn
eine CL-Modifikation $K \in Modi^V_{CL}\,(I)$ von I bezüglich $V = \{s\}$ existiert mit $K \models \psi$, nach Definition 4.32
Da $seq_K\,(s) \in UD^*_I$ folgt: $I \models \varphi$ genau dann, wenn
(1) eine CL-Modifikation $K \in Modi^V_{CL}\,(I)$ von I bezüglich $V = \{s\}$ mit
$seq_K\,(s) \in UD^0_I$ existiert, so dass $K \models \psi$ oder
(2) eine CL-Modifikation $K \in Modi^V_{CL}\,(I)$ von I bezüglich $V = \{s\}$ mit
$seq_K\,(s) \in UD_I$ existiert, so dass $K \models \psi$ oder
(3) eine CL-Modifikation $K \in Modi^V_{CL}\,(I)$ von I bezüglich $V = \{s\}$ mit
$seq_K\,(s) \in UD^2_I$ existiert, so dass $K \models \psi$ usw.
Per Induktion über den Formelaufbau von ψ lässt sich zeigen, dass $I \models \varphi$ genau dann, wenn
(1) eine CL-Modifikation $K \in Modi^V_{CL}\,(I)$ von I bezüglich $V = \emptyset$ existiert, so dass $K \models tr[\varepsilon]\,(\psi)$ oder
(2) eine CL-Modifikation $K \in Modi^V_{CL}\,(I)$ von I bezüglich $V = \{x_1\}$ existiert, so dass $K \models tr[x_1]\,(\psi)$ oder
(3) eine CL-Modifikation $K \in Modi^V_{CL}\,(I)$ von I bezüglich $V = \{x_1, x_2\}$ existiert, so dass $K \models tr[x_1 x_2]\,(\psi)$ usw.
Das heißt $I \models \varphi$ genau dann, wenn eine CL-Modifikation $K \in Modi^V_{CL}\,(I)$ von I bezüglich $V = \{x_1, x_2, \ldots, x_n\}$ mit $n \in \mathbb{N}$ existiert, so dass $K \models tr[x_1\ x_2\ \ldots\ x_n]\,(\psi)$.
Also $I \models \varphi$ genau dann, wenn
(1) $I \models tr[\varepsilon]\,(\psi)$ oder
(2) $I \models (\mathsf{exists}\,(x_1)\,tr[x_1]\,(\psi))$ oder
(3) $I \models (\mathsf{exists}\,(x_1\ x_2)\,tr[x_1\ x_2]\,(\psi))$ usw.
Damit gilt $I \models \varphi$ genau dann, wenn
$I \models (\mathsf{exists}\,(x_1\ x_2\ \ldots\ x_n)\,tr[x_1\ x_2\ \ldots\ x_n]\,(\psi))$ für ein $n \in \mathbb{N}$.

■

4.6 Endlichkeit und Kompaktheit

Nachdem wir in Kapitel 4.5 die Semantik der CL-Sequenznamen analysiert haben, werden wir in diesem Abschnitt unter Verwendung der CL-Sequenznamen zeigen, dass in Common Logic die Endlichkeit des Gegenstandsbereichs einer CL-Interpretation gefordert werden kann. Mithilfe dieses Resultats werden wir anschließend beweisen, dass Common Logic nicht kompakt ist.

Zunächst betrachten wir einige CL-Formeln und erläutern deren Semantik.

Seien $x, y \in N$ zwei CL-Objektnamen und $s \in SEQ$ ein CL-Sequenzname. Dann definieren wir die CL-Formel φ_1 wie folgt:

$$\varphi_1 = (\mathsf{exists}\ (s\ y)\ (\mathsf{forall}\ (x)\ (\mathtt{Id}\ x\ s\ y))) \tag{4.23}$$

Lemma 4.46
Sei $I \in \mathcal{IB}_{CL}(V_{CL}, IFm)$ eine CL-Interpretation mit $I = \langle \mathcal{S}_I, int_I, seq_I, irr_I \rangle$ und der semantischen CL-Struktur $\mathcal{S}_I = \langle UR_I, UD_I, rel_I, fun_I \rangle$. Dann gilt $I \models \varphi_1$ genau dann, wenn ein Tupel $\langle u_1, u_2, \ldots, u_n \rangle \in UD_I^*$ mit $|\langle u_1, u_2, \ldots, u_n \rangle| \geq 1$ existiert, so dass jedes Tupel $\langle v, u_1, u_2, \ldots, u_n \rangle$ mit $v \in UD_I$ zu der relationalen Extension des von `Id` referenzierten Objekts gehört.

Beweis:
Im Folgenden bezeichne $V_{\varphi_1} = \{x, y, s, \mathtt{Id}\}$ das Vokabular der CL-Formel φ_1. Weiter sei $I \in \mathcal{IB}_{CL}(V_{CL}, IFm)$ eine CL-Interpretation über V_{CL} und IFm mit $V_{\varphi_1} \subseteq V_{CL}$, $I = \langle \mathcal{S}_I, int_I, seq_I, irr_I \rangle$ und $\mathcal{S}_I = \langle UR_I, UD_I, rel_I, fun_I \rangle$.
Es gilt $I \models \varphi_1$ genau dann, wenn
eine CL-Modifikation $K \in Modi_{CL}^V(I)$ von I bezüglich $V = \{s, y\}$ existiert mit $K \models (\mathsf{forall}\ (x)\ (\mathtt{Id}\ x\ s\ y))$, nach Definition 4.32
gdw. für alle CL-Modifikationen $L \in Modi_{CL}^V(K)$ von K bezüglich $V = \{x\}$ gilt $L \models (\mathtt{Id}\ x\ s\ y)$, nach Definition 4.32
gdw. für alle CL-Modifikationen $L \in Modi_{CL}^V(K)$ von K bezüglich $V = \{x\}$ gilt $int_L^*(x\ s\ y) \in rel_L(int_L^*(\mathtt{Id}))$, nach Definition 4.32
gdw. für alle CL-Modifikationen $L \in Modi_{CL}^V(K)$ von K bezüglich $V = \{x\}$ gilt $\langle int_L^*(x) \rangle \bullet int_L^*(s) \bullet \langle int_L^*(y) \rangle \in rel_L(int_L^*(\mathtt{Id}))$, nach Definition 4.30
gdw. für alle CL-Modifikationen $L \in Modi_{CL}^V(K)$ von K bezüglich $V = \{x\}$ gilt $\langle int_L(x) \rangle \bullet seq_K(s) \bullet \langle int_K(y) \rangle \in rel_I(int_I(\mathtt{Id}))$,
nach Definition 4.30 und Definition 4.23
gdw. für alle $v \in UD_L = UD_I$ gilt: $\langle v \rangle \bullet seq_K(s) \bullet \langle int_K(y) \rangle \in rel_I(int_I(\mathtt{Id}))$
gdw. ein Tupel $\langle u_1, u_2, \ldots, u_n \rangle \in UD_K^* = UD_I^*$ mit $seq_K(s) = \langle u_1, u_2, \ldots, u_{n-1} \rangle$ und $int_K(y) = u_n$ existiert, so dass für alle $v \in UD_I$ gilt $\langle v, u_1, u_2, \ldots, u_n \rangle \in rel_I(int_I(\mathtt{Id}))$.

■

Weiter seien $x, y \in N$ zwei CL-Objektnamen und $s \in SEQ$ ein CL-Sequenzname. Die CL-Formel φ_2 sei folgendermaßen definiert:

$$\begin{aligned}\varphi_2 = (\mathsf{forall}\ (x\ y\ s)\ (\mathsf{iff}\\ &(\texttt{Id}\ x\ s\ y)\\ &(\mathsf{or}\\ &\quad(\mathsf{equal}\ x\ y)\\ &\quad(\texttt{Id}\ x\ s)\\ &\ )\\ ))\end{aligned} \tag{4.24}$$

Lemma 4.47
Sei $I \in \mathcal{IB}_{CL}(V_{CL}, IFm)$ eine CL-Interpretation mit $I = \langle \mathcal{S}_I, int_I, seq_I, irr_I \rangle$ und der semantischen CL-Struktur $\mathcal{S}_I = \langle UR_I, UD_I, rel_I, fun_I \rangle$.
Dann gilt $I \models \varphi_2$ genau dann, wenn für jedes Tupel $\langle u_1, u_2, \ldots, u_n \rangle \in UD_I^*$ mit $|\langle u_1, u_2, \ldots, u_n \rangle| \geq 2$ gilt $\langle u_1, u_2, \ldots, u_n \rangle \in rel_I\,(int_I\,(\texttt{Id}))$ genau dann, wenn $u_1 \in rel_I\,(int_I\,(\texttt{Id}))$ oder es existiert ein $u_i \in UD_I$ mit $u_1 = u_i$ für $2 \leq i \leq n$.

Beweis:
Im Folgenden bezeichne $V_{\varphi_2} = \{x, y, s, \texttt{Id}\}$ das Vokabular der CL-Formel φ_2. Weiter sei $I \in \mathcal{IB}_{CL}(V_{CL}, IFm)$ eine CL-Interpretation über V_{CL} und IFm mit $V_{\varphi_2} \subseteq V_{CL}$, $I = \langle \mathcal{S}_I, int_I, seq_I, irr_I \rangle$ und $\mathcal{S}_I = \langle UR_I, UD_I, rel_I, fun_I \rangle$.
Es gilt $I \models \varphi_2$ genau dann, wenn
für alle CL-Modifikationen $K \in Modi_{CL}^V\,(I)$ von I bezüglich $V = \{x, y, s\}$ gilt $K \models$ (iff (Id x s y) (or (equal x y) (Id x s))), nach Definition 4.32
gdw. für alle CL-Modifikationen $K \in Modi_{CL}^V\,(I)$ von I bezüglich $V = \{x, y, s\}$ gilt $K \models (\texttt{Id}\ x\ s\ y)$ genau dann, wenn $K \models$ (or (equal x y) (Id x s)),
nach Definition 4.32
gdw. für alle CL-Modifikationen $K \in Modi_{CL}^V\,(I)$ von I bezüglich $V = \{x, y, s\}$ gilt $int_K^*\,(x\ s\ y) \in rel_K\,(int_K^*\,(\texttt{Id}))$ genau dann, wenn $K \models$ (equal x y) oder $K \models (\texttt{Id}\ x\ s)$, nach Definition 4.32
gdw. für alle CL-Modifikationen $K \in Modi_{CL}^V\,(I)$ von I bezüglich $V = \{x, y, s\}$ gilt $\langle int_K\,(x) \rangle \bullet seq_K\,(s) \bullet \langle int_K\,(y) \rangle \in rel_I\,(int_I\,(\texttt{Id}))$ genau dann, wenn $int_K\,(x) = int_K\,(y)$ oder $\langle int_K\,(x) \rangle \bullet seq_K\,(s) \in rel_I\,(int_I\,(\texttt{Id}))$. $(*)$
Wir zeigen die Behauptung per Induktion über die Tupellänge n.
Induktionsanfang für $n = 2$:
Sei $\langle u_1, u_2 \rangle \in rel_I\,(int_I\,(\texttt{Id}))$ mit $\langle u_1, u_2 \rangle \in UD_I^2$.
Nach $(*)$ folgt $I \models \varphi_2$ genau dann, wenn für alle CL-Modifikationen $K \in Modi_{CL}^V\,(I)$ von I bezüglich $V = \{x, y, s\}$ gilt:
$\langle int_K\,(x) \rangle \bullet seq_K\,(s) \bullet \langle int_K\,(y) \rangle = \langle u_1, u_2 \rangle \in rel_I\,(int_I\,(\texttt{Id}))$ genau dann, wenn $u_1 = int_K\,(x) = int_K\,(y) = u_2$ oder $u_1 = \langle int_K\,(x) \rangle \bullet seq_K\,(s) \in rel_I\,(int_I\,(\texttt{Id}))$
gdw. für alle CL-Modifikationen $K \in Modi_{CL}^V\,(I)$ von I bezüglich $V = \{x, y, s\}$ gilt

$\langle u_1, u_2 \rangle \in rel_I\,(int_I\,(\texttt{Id}))$ genau dann, wenn $u_1 = u_2$ oder $u_1 \in rel_I\,(int_I\,(\texttt{Id}))$.
Induktionsvoraussetzung:
Die Behauptung gelte für alle Tupel $\langle\, u_1,\, u_2,\, \ldots,\, u_n \,\rangle \in UD_I^*$ mit $2 \leq |\langle u_1, u_2, \ldots u_n \rangle| \leq n$.
Induktionsschritt:
Wir zeigen nun, dass die Behauptung auch für alle Tupel der Länge $n+1$ gültig ist.
Sei $\langle u_1, u_2, \ldots, u_n, u_{n+1} \rangle \in rel_I\,(int_I\,(\texttt{Id}))$ mit $|\langle u_1, u_2, \ldots, u_n, u_{n+1} \rangle| > 2$.
Nach $(*)$ folgt $I \models \varphi_2$ genau dann, wenn für alle CL-Modifikationen $K \in Modi_{CL}^V\,(I)$ von I bezüglich $V = \{x, y, s\}$ gilt:
$\langle int_K\,(x) \rangle \bullet seq_K\,(s) \bullet \langle int_K\,(y) \rangle = \langle u_1, u_2, \ldots, u_{n+1} \rangle \in rel_I\,(int_I\,(\texttt{Id}))$ genau dann, wenn $u_1 = int_K\,(x) = int_K\,(y) = u_{n+1}$ oder $\langle u_1, u_2, \ldots, u_n \rangle = \langle int_K\,(x) \rangle \bullet seq_K\,(s) \in rel_I\,(int_I\,(\texttt{Id}))$
gdw. für alle CL-Modifikationen $K \in Modi_{CL}^V\,(I)$ von I bezüglich $V = \{x, y, s\}$ gilt $\langle u_1, u_2, \ldots, u_n, u_{n+1} \rangle \in rel_I\,(int_I\,(\texttt{Id}))$ genau dann, wenn $u_1 = u_{n+1}$ oder $u_1 \in rel_I\,(int_I\,(\texttt{Id}))$ oder es existiert ein $u_i \in UD_I$ für $2 \leq i \leq n$ mit $u_1 = u_i$ nach Induktionsvoraussetzung.

■

Unter der Annahme, dass $x \in N$ ein CL-Objektname ist, formulieren wir die CL-Formel φ_3 wie folgt:

$$\varphi_3 = (\mathsf{forall}\,(x)\,(\mathsf{not}\,(\texttt{Id}\ x))) \tag{4.25}$$

Lemma 4.48
Sei $I \in \mathcal{IB}_{CL}(V_{CL}, IFm)$ eine CL-Interpretation mit $I = \langle \mathcal{S}_I, int_I, seq_I, irr_I \rangle$ und der semantischen CL-Struktur $\mathcal{S}_I = \langle UR_I, UD_I, rel_I, fun_I \rangle$. Dann gilt $I \models \varphi_3$ genau dann, wenn für alle $u \in UD_I$ gilt $u \notin rel_I\,(int_I\,(\texttt{Id}))$.

Beweis:
Im Folgenden bezeichne $V_{\varphi_3} = \{x, \texttt{Id}\}$ das Vokabular der CL-Formel φ_3. Weiter sei $I \in \mathcal{IB}_{CL}(V_{CL}, IFm)$ eine CL-Interpretation über V_{CL} und IFm mit $V_{\varphi_3} \subseteq V_{CL}$, $I = \langle \mathcal{S}_I, int_I, seq_I, irr_I \rangle$ und $\mathcal{S}_I = \langle UR_I, UD_I, rel_I, fun_I \rangle$.
Es gilt $I \models \varphi_3$ genau dann, wenn
für alle CL-Modifikationen $K \in Modi_{CL}^V\,(I)$ von I bezüglich $V = \{x\}$ gilt
$K \models (\mathsf{not}\,(\texttt{Id}\ x))$, nach Definition 4.32
gdw. für alle CL-Modifikationen $K \in Modi_{CL}^V\,(I)$ von I bezüglich $V = \{x\}$ gilt $int_K\,(x) \notin rel_I\,(int_I\,(\texttt{Id}))$.
Also $I \models \varphi_3$ genau dann, wenn für alle $u \in UD_I$ gilt $u \notin rel_I\,(int_I\,(\texttt{Id}))$.

■

Mithilfe der CL-Formeln φ_1, φ_2 und φ_3 werden wir nun das erste Hauptresultat dieses Unterkapitels beweisen.

Satz 4.49
Es existiert eine CL-Formel φ, so dass für jede CL-Interpretation $I \in \mathcal{IB}_{CL}(V_{CL}, IFm)$ gilt $I \models \varphi$ genau dann, wenn der Gegenstandsbereich UD_I von I endlich ist.

Beweis:
Um Satz 4.49 zu beweisen, konstruieren wir zunächst die CL-Formel φ und zeigen anschließend, dass φ die geforderte Eigenschaft erfüllt.
Konstruktion der CL-Formel φ:
Die CL-Formel φ ist die Konjunktion der CL-Formeln φ_1 aus (4.23) , φ_2 aus (4.24) und φ_3 aus (4.25). Also $\varphi = (\mathsf{and}\ \ \varphi_1\ \ \varphi_2\ \ \varphi_3)$.
Nun zeigen wir für $I \in \mathcal{IB}_{CL}(V_{CL}, IFm)$ die Behauptung $I \models \varphi$ genau dann, wenn UD_I endlich ist.
Im Folgenden bezeichne V_φ das Vokabular der CL-Formel φ. Weiterhin sei $I \in \mathcal{IB}_{CL}(V_{CL}, IFm)$ eine CL-Interpretation über V_{CL} und IFm mit $V_\varphi \subseteq V_{CL}$, $I = \langle \mathcal{S}_I, int_I, seq_I, irr_I \rangle$ und $\mathcal{S}_I = \langle UR_I, UD_I, rel_I, fun_I \rangle$.
Es gilt $I \models \varphi$ genau dann, wenn
$I \models (\mathsf{and}\ \varphi_2\ \varphi_3)$ und nach Lemma 4.46 existiert ein Tupel $\langle u_1, u_2, \ldots, u_n \rangle \in UD_I^*$ mit $|\langle u_1, u_2, \ldots, u_n \rangle| \geq 1$, so dass für alle $v \in UD_I$ gilt
$\langle v, u_1, u_2, \ldots, u_n \rangle \in rel_I\,(int_I\,(\mathtt{Id}))$
gdw. $I \models \varphi_3$ und es existiert ein Tupel $\langle u_1, u_2, \ldots, u_n \rangle \in UD_I^*$ mit $|\langle u_1, u_2, \ldots, u_n \rangle| \geq 1$, so dass nach Lemma 4.47 für alle $v \in UD_I$ gilt
(1) $v \in rel_I\,(int_I\,(\mathtt{Id}))$ oder (2) es existiert ein u_i mit $v = u_i$ für $1 \leq i \leq n$
gdw. ein Tupel $\langle u_1, u_2, \ldots, u_n \rangle \in UD_I^*$ mit $|\langle u_1, u_2, \ldots, u_n \rangle| \geq 1$ existiert, so dass nach Lemma 4.48 für alle $v \in UD_I$ gilt $v = u_i$ für $1 \leq i \leq n$.
Das heißt $I \models \varphi$ genau dann, wenn eine natürliche Zahl n existiert, so dass der Gegenstandsbereich UD_I höchstens n verschiedene Elemente umfasst.

■

Es sei angemerkt, dass Satz 4.49 sowohl für unsegregierte CL-Dialekte als auch für segregierte CL-Dialekte gültig ist. In Common Logic ist es also generell möglich, unter Verwendung der CL-Sequenznamen, die Endlichkeit des Gegenstandsbereichs einer CL-Interpretation zu fordern. Allerdings lassen sich für die beiden Dialektarten von Common Logic unterschiedliche Folgerungen aus der Endlichkeit des Gegenstandsbereichs ableiten. Für jeden unsegregierten CL-Dialekt folgt aus Satz 4.49 auch die Endlichkeit des Referenzbereichs. Für segregierte CL-Dialekte ist diese Folgerung nicht möglich.
In der Prädikatenlogik erster Stufe ist im Gegensatz zu Common Logic die Endlichkeit nicht ausdrückbar.

Für die Formulierung des zweiten Hauptresultats betrachten wir zunächst eine Reihe von CL-Formeln.

$$\psi\,(2) = (\mathsf{exists}\,(x_1\ x_2)\,(\mathsf{not}\,(\mathsf{equal}\ x_1\ x_2))) \tag{4.26}$$

Für eine CL-Interpretation $I \in \mathcal{IB}_{CL}(V_{CL}, IFm)$ gilt $I \models \psi\,(2)$ genau dann, wenn

UD_I mindestens zwei verschiedene Elemente enthält.

$$\begin{aligned}\psi(3) = (\mathsf{exists}\,(x_1\ x_2\ x_3)\,(\mathsf{and}\\ &(\mathsf{not}\,(\mathsf{equal}\ x_1\ x_2))\\ &(\mathsf{not}\,(\mathsf{equal}\ x_1\ x_3))\\ &(\mathsf{not}\,(\mathsf{equal}\ x_2\ x_3))\\ ))\end{aligned} \tag{4.27}$$

Also gilt $I \models \psi(3)$ genau dann, wenn UD_I mindestens drei verschiedene Elemente enthält.

$$\begin{aligned}\psi(n) = (\mathsf{exists}\,(x_1\ x_2\ \ldots\ x_n)\,(\mathsf{and}\\ &(\mathsf{not}\,(\mathsf{equal}\ x_1\ x_2))\\ &(\mathsf{not}\,(\mathsf{equal}\ x_1\ x_3))\\ &\ldots\\ &(\mathsf{not}\,(\mathsf{equal}\ x_1\ x_n))\\ &(\mathsf{not}\,(\mathsf{equal}\ x_2\ x_3))\\ &\ldots\\ &(\mathsf{not}\,(\mathsf{equal}\ x_{n-1}\ x_n))\\ ))\end{aligned} \tag{4.28}$$

Für den allgemeinen Fall definieren wir die CL-Formel $\psi(n)$. Dann gilt $I \models \psi(n)$ genau dann, wenn UD_I mindestens n verschiedene Elemente enthält.

Satz 4.50
In Common Logic ist der Endlichkeitssatz (Satz 2.14) nicht gültig.

Beweis:
Wir konstruieren ein Gegenbeispiel, um den Endlichkeitssatz zu widerlegen. Es sei $F \subseteq Fm(V_{CL})$ eine Menge von CL-Formeln mit $F = \{\varphi, \psi(2), \psi(3), \psi(4), \ldots\}$, wobei φ die CL-Formel aus dem Beweis von Satz 4.49 bezeichnet und die Endlichkeit des Gegenstandsbereichs fordert. Es gilt nun, dass jede endliche Teilmenge von F ein Modell hat, aber die Formelmenge F selbst besitzt kein Modell. Dies widerspricht dem Endlichkeitssatz.

■

Eine wichtige Folgerung von Satz 4.50 ist, dass die CL-Folgerungsrelation $\models_{CL}$ nicht kompakt ist.
Die Tatsache, dass der Endlichkeitssatz in Common Logic nicht gültig ist, ist der Common Logic Gemeinschaft bereits bekannt. Ein weiteres Gegenbeispiel, welches den Endlichkeitssatz ebenfalls widerlegt, ist in [52] zu finden. Dieses Beispiel werden wir nun zum Abschluss des Kapitels angeben.

Beispiel 4.51 (weiteres Gegenbeispiel zum Endlichkeitssatz, aus [52])
Seien $x_1, x_2, \ldots, x_n \in N$ CL-Objektnamen, $\mathtt{R} \in N$ ein CL-Name und $seq \in SEQ$ ein CL-Sequenzname. Wir definieren die CL-Formeln φ, $\varphi(0)$, $\varphi(1)$ und $\varphi(n)$ mit $n \in \mathbb{N}$ wie folgt:

$$\varphi = (\mathsf{exists}\,(seq)\,(\mathtt{R}\; seq)) \tag{4.29}$$
$$\varphi(0) = (\mathsf{not}\,(\mathtt{R})) \tag{4.30}$$
$$\varphi(1) = (\mathsf{not}\,(\mathsf{exists}\,(x_1)\,(\mathtt{R}\; x_1))) \tag{4.31}$$
$$\varphi(n) = (\mathsf{not}\,(\mathsf{exists}\,(x_1\; x_2\; \ldots\; x_n)\,(\mathtt{R}\; x_1\; x_2\; \ldots\; x_n))) \tag{4.32}$$

Weiter sei $G \subseteq Fm(V_{CL})$ eine Menge von CL-Formeln, die wie folgt definiert ist: $G = \{\varphi,\ \varphi(0),\ \varphi(1),\ \varphi(2),\ \ldots\}$.
Dann besitzt jede endliche Teilmenge von G ein Modell, aber die Formelmenge G selbst hat kein Modell.

□

4.7 Die syntaktischen Freiheiten von Common Logic

Common Logic und die Prädikatenlogik der ersten Stufe unterscheiden sich auch ohne die Verwendung der CL-Sequenznamen in ihren Modellierungsfähigkeiten. Aufgrund der einheitlichen Behandlung der CL-Namen sind in Common Logic syntaktische Konstruktionen möglich, die in der Prädikatenlogik erster Stufe nicht erlaubt sind. Als Beispiel sei die Quantifizierung über Funktions- und Relationssymbole genannt. In diesem Abschnitt wollen wir die syntaktischen Freiheiten von Common Logic anhand einiger Beispiele demonstrieren.
Wir beginnen mit nachfolgender CL-Formel, wobei $X \in N$ ein CL-Objektname ist.

$$\varphi = (\mathsf{exists}\,(X)\,(X\,(X\; X))) \tag{4.33}$$

Die formale Semantik der CL-Formel φ ist wie folgt definiert.
Im Folgenden bezeichne V_φ das Vokabular der CL-Formel φ. Weiterhin sei $I \in \mathcal{IB}_{CL}(V_{CL}, IFm)$ eine CL-Interpretation über V_{CL} und IFm mit $V_\varphi \subseteq V_{CL}$, $I = \langle \mathcal{S}_I, int_I, seq_I, irr_I \rangle$ und $\mathcal{S}_I = \langle UR_I, UD_I, rel_I, fun_I \rangle$.
Es gilt $I \models \varphi$ genau dann, wenn
eine CL-Modifikation $K \in Modi^V_{CL}(I)$ von I bezüglich $V = \{X\}$ existiert mit $K \models (X\,(X\; X))$, nach Definition 4.32
gdw. eine CL-Modifikation $K \in Modi^V_{CL}(I)$ von I bezüglich $V = \{X\}$ existiert mit $int^*_K((X\; X)) \in rel_K(int^*_K(X))$, nach Definition 4.32
gdw. eine CL-Modifikation $K \in Modi^V_{CL}(I)$ von I bezüglich $V = \{X\}$ existiert mit $fun_K(int^*_K(X))(int^*_K(X)) \in rel_K(int^*_K(X))$, nach Definition 4.30
gdw. eine CL-Modifikation $K \in Modi^V_{CL}(I)$ von I bezüglich $V = \{X\}$ existiert mit $fun_I(int_K(X))(int_K(X)) \in rel_I(int_K(X))$.

Die CL-Formel φ aus 4.33 fordert also, dass die CL-Interpretation I mindestens ein $u \in UD_I$ enthält, so dass die funktionale Extension von u das Element u auf ein Element $v \in UD_I$ abbildet, welches ein Element der relationalen Extension von u ist.
Diese Bedingung ist durchaus erfüllbar wie das folgende Beispiel zeigt.

Beispiel 4.52
Es sei $I \in \mathcal{IB}_{CL}(V_{CL}, IFm)$ eine CL-Interpretation, die wie folgt definiert ist:

- $UD_I = UR_I = \{u\}$,
- $rel_I(u) = \{u\}$,
- $fun_I(u) = f$ mit $f(x) = u$ für alle $x \in UD_I^*$ und
- $int_I(X) = u$.

Dann gilt $I \models \varphi$.

□

Aufgrund der syntaktischen Freiheiten lassen sich in Common Logic Bedingungen formulieren, die in der Prädikatenlogik nicht oder nur sehr unzulänglich spezifiziert werden können. Zur Veranschaulichung betrachten wir ein Beispiel.

Beispiel 4.53 (Fixpunkteigenschaft aller Funktionen)
Unter der Annahme, dass alle Funktionsnamen CL-Objektnamen sind, kann in Common Logic gefordert werden, dass alle Funktionen einen Fixpunkt besitzen. Die folgende CL-Formel drückt diesen Sachverhalt aus.

$$(\mathsf{forall}\ (f)\ (\mathsf{exists}\ (x)\ (\mathsf{equal}\ (f\ x)\ x))) \tag{4.34}$$

□

In der Prädikatenlogik sind derartige Aussagen nicht möglich, da dort nicht über Funktionssymbole quantifiziert werden kann. Um eine vergleichbare Eigenschaft für die prädikatenlogischen Funktionen auszudrücken, muss die Fixpunkteigenschaft für alle einstelligen Funktionssymbole f_1, f_2, ..., f_n einzeln gefordert werden, wie in der prädikatenlogischen Formel (4.35) gezeigt.

$$((\exists x\,(f_1(x) \circledcirc x) \wedge \exists x\,(f_2(x) \circledcirc x)) \ \wedge\ \ldots) \tag{4.35}$$

Die Quantifizierung über Relationsnamen erlaubt ebenfalls die Formulierung von Bedingungen, die in der Prädikatenlogik der ersten Stufe nicht ausgedrückt werden können.

Beispiel 4.54 (Potenzmenge)
So lässt sich in Common Logic die Potenzmenge `PotX` der Menge `X` wie folgt charakterisieren:

$$
\begin{aligned}
&(\mathsf{forall}\ (\texttt{U})\ (\mathsf{iff} \\
&\qquad (\texttt{PotX}\ \texttt{U}) \\
&\qquad (\mathsf{forall}\ (y)\ (\mathsf{if} \\
&\qquad\qquad (\texttt{U}\ y) \\
&\qquad\qquad (\texttt{X}\ y) \\
&\qquad\qquad)) \\
&\qquad))
\end{aligned}
\tag{4.36}
$$

□

Eine weitere interessante Eigenschaft von Common Logic ist die Tatsache, dass eine Klasse sich selbst als Element enthalten kann. Diesen Sachverhalt wollen wir anhand des folgenden Beispiels näher erläutern.

Beispiel 4.55
Sei ψ eine CL-Formel, welche genau die Bedingungen beschreibt, die erfüllt sein müssen, um als ein Mensch charakterisiert zu werden. Weiter bezeichne `Mensch` die Klasse aller Menschen, also diejenige Klasse, welche genau die Objekte enthält, die die CL-Formel ψ erfüllen. Die CL-Formel (4.37) formuliert diesen Sachverhalt in Common Logic, wobei der Ausdruck $\psi\,(x)$ bedeutet, dass x die CL-Formel ψ erfüllt.

$$(\mathsf{forall}\ (x)\ (\mathsf{iff}\ (\texttt{Mensch}\ x)\ (\psi\ (x)))) \tag{4.37}$$

Des Weiteren führen wir die Klasse `NichtMensch` ein, welche alle Objekte beinhaltet, die keine Menschen sind. Dies wird in Common Logic wie folgt formuliert:

$$(\mathsf{forall}\ (x)\ (\mathsf{iff}\ (\texttt{NichtMensch}\ x)\ (\mathsf{not}\ (\texttt{Mensch}\ x)))) \tag{4.38}$$

□

Unter der Annahme, dass `Mensch` ein CL-Objektname ist, ist das von `Mensch` referenzierte Objekt des Gegenstandsbereichs im Allgemeinen kein Mensch, sondern eine Klasse. Deshalb ist in jeder CL-Interpretation, welche die CL-Formeln (4.37) und (4.38) erfüllt, auch die folgende CL-Formel wahr:

$$(\mathsf{not}\ (\texttt{Mensch}\ \texttt{Mensch})) \tag{4.39}$$

Demnach ist auch die nachfolgende Aussage gültig.

$$(\texttt{NichtMensch}\ \texttt{Mensch}) \tag{4.40}$$

Der CL-Name `NichtMensch` bezeichnet auch keinen Menschen, sondern ebenfalls eine Klasse und zwar die Klasse aller derjenigen Objekte die keine Menschen sind. Das heißt, falls `NichtMensch` ein CL-Objektname ist, dann beinhaltet die relationale Extension des von `NichtMensch` benannten Objekts das referenzierte Objekt selbst. Dies bedeutet, dass auch die folgende CL-Formel gültig ist.

$$(\texttt{NichtMensch}\ \texttt{NichtMensch}) \tag{4.41}$$

Dieses Phänomen wird in der Literatur als *Selbstreferenz* (self-reference) oder *Selbstanwendung* (self-application) bezeichnet [15].

4.8 Die Russellsche Klasse

Im vorangegangenen Abschnitt haben wir die syntaktischen Freiheiten, welche Common Logic erlaubt, ausführlich erläutert. Aufgrund dieser Freiheiten ist es möglich, in Common Logic sogenannte *Antinomien* zu formulieren. Eine sehr bekannte Antinomie, die sich in Common Logic als wohlgeformte CL-Formel beschreiben lässt, ist die *Russellsche Antinomie* [47], deren Semantik in diesem Teil der Diplomarbeit näher untersucht wird.

Definition 4.56 (Russellsche Klasse)
Die *Russellsche Klasse* `Russell` ist die Klasse aller derjenigen Klassen, die sich nicht selbst als Element enthalten.

□

In Common Logic lässt sich die Russellsche Klasse unter der Annahme, dass K und `Russell` zwei CL-Objektnamen sind, wie folgt definieren:

$$\begin{aligned}\varphi = (\mathsf{exists}\ (\texttt{Russell})\ (\mathsf{forall}\ (K)\ (\mathsf{iff}\\ &(\texttt{Russell}\ K)\\ &(\mathsf{not}\ (K\ K))\\ &)))\end{aligned} \tag{4.42}$$

Für diese CL-Formel zeigen wir nun das nachfolgende Lemma.

Lemma 4.57
Die CL-Formel φ aus (4.42) ist in allen CL-Dialekten unerfüllbar.

Beweis:
Im Folgenden bezeichne V_φ das Vokabular der CL-Formel φ. Weiterhin sei $I \in \mathcal{IB}_{CL}(V_{CL}, IFm)$ eine CL-Interpretation über V_{CL} und IFm mit $V_\varphi \subseteq V_{CL}$, $I = \langle \mathcal{S}_I, int_I, seq_I, irr_I \rangle$ und $\mathcal{S}_I = \langle UR_I, UD_I, rel_I, fun_I \rangle$.
Es gilt $I \models \varphi$ genau dann, wenn
eine CL-Modifikation $P \in Modi^V_{CL}(I)$ von I bezüglich $V = \{\texttt{Russell}\}$ existiert mit

$P \models (\mathsf{forall}\,(K)\,(\mathsf{iff}\,(\texttt{Russell}\;K)\,(\mathsf{not}\,(K\;K))))$, nach Definition 4.32
gdw. für alle CL-Modifikationen $Q \in Modi^{V}_{CL}(P)$ von P bezüglich $V = \{K\}$ gilt $Q \models (\mathsf{iff}\,(\texttt{Russell}\;K)\,(\mathsf{not}\,(K\;K)))$, nach Definition 4.32
gdw. für alle CL-Modifikationen $Q \in Modi^{V}_{CL}(P)$ von P bezüglich $V = \{K\}$ gilt $Q \models (\mathsf{if}\,(\texttt{Russell}\;K)\,(\mathsf{not}\,(K\;K)))$ und $Q \models (\mathsf{if}\,(\mathsf{not}\,(K\;K))\,(\texttt{Russell}\;K))$,
nach Definition 4.32
gdw. für alle CL-Modifikationen $Q \in Modi^{V}_{CL}(P)$ von P bezüglich $V = \{K\}$ gilt $Q \models (\mathsf{or}\,(\mathsf{not}\,(\texttt{Russell}\;K))(\mathsf{not}\,(K\;K)))$ und $Q \models (\mathsf{or}\,(\mathsf{not}\,(\mathsf{not}\,(K\;K)))(\texttt{Russell}\;K))$,
nach Definition 4.32
gdw. für alle CL-Modifikationen $Q \in Modi^{V}_{CL}(P)$ von P bezüglich $V = \{K\}$ gilt
(1) $Q \models (\mathsf{not}\,(\texttt{Russell}\;K))$ oder $Q \models (\mathsf{not}\,(K\;K))$ und
(2) $Q \models (\mathsf{not}\,(\mathsf{not}\,(K\;K)))$ oder $Q \models (\texttt{Russell}\;K)$, nach Definition 4.32
gdw. für alle CL-Modifikationen $Q \in Modi^{V}_{CL}(P)$ von P bezüglich $V = \{K\}$ gilt
(1) $int^{*}_{Q}(K) \notin rel_{Q}\left(int^{*}_{Q}(\texttt{Russell})\right)$ oder $int^{*}_{Q}(K) \notin rel_{Q}\left(int^{*}_{Q}(K)\right)$ und
(2) $int^{*}_{Q}(K) \in rel_{Q}\left(int^{*}_{Q}(K)\right)$ oder $int^{*}_{Q}(K) \in rel_{Q}\left(int^{*}_{Q}(\texttt{Russell})\right)$,
nach Definition 4.32
gdw. für alle CL-Modifikationen $Q \in Modi^{V}_{CL}(P)$ von P bezüglich $V = \{K\}$ gilt
(1) $int_{Q}(K) \notin rel_{I}(int_{P}(\texttt{Russell}))$ und $int_{Q}(K) \in rel_{I}(int_{Q}(K))$ oder
(2) $int_{Q}(K) \notin rel_{I}(int_{Q}(K))$ und $int_{Q}(K) \in rel_{I}(int_{P}(\texttt{Russell}))$.
Die Bedingungen (1) und (2) müssen für alle CL-Modifikationen $Q \in Modi^{V}_{CL}(P)$ von P bezüglich $V = \{K\}$ erfüllt sein.
Also auch für die CL-Modifikation $\widehat{Q}$ mit $int_{\widehat{Q}}(K) = int_{P}(\texttt{Russell})$.
Dann gilt $\widehat{Q} \models \varphi$ genau dann, wenn
(1) $int_{P}(\texttt{Russell}) \notin rel_{I}(int_{P}(\texttt{Russell}))$ und
$int_{P}(\texttt{Russell}) \in rel_{I}(int_{P}(\texttt{Russell}))$ oder
(2) $int_{P}(\texttt{Russell}) \notin rel_{I}(int_{P}(\texttt{Russell}))$ und
$int_{P}(\texttt{Russell}) \in rel_{I}(int_{P}(\texttt{Russell}))$.
Da sowohl die Bedingung (1) als auch die Bedingung (2) unerfüllbar ist, gilt also $\widehat{Q} \not\models \varphi$ und damit $I \not\models \varphi$.

■

Nun werden wir die CL-Formel φ aus (4.42) geringfügig abändern.

$$
\begin{array}{l}
(\mathsf{forall}\,(K)\,(\mathsf{iff} \\
\qquad (\texttt{Russell}\;K) \\
\qquad (\mathsf{not}\,(K\;K)) \\
\quad))
\end{array}
\tag{4.43}
$$

Die beiden CL-Formeln aus (4.42) und (4.43) unterscheiden sich darin, dass der CL-Name `Russell` in (4.43) nicht durch einen Quantor gebunden ist und deshalb in einem segregierten CL-Dialekt auch als CL-Referenzname interpretiert werden kann.

Zunächst zeigen wir die Gültigkeit des folgenden Lemmas.

Lemma 4.58
Die CL-Formel φ aus (4.43) ist in allen unsegregierten CL-Dialekten unerfüllbar.

Beweis:
Im Folgenden bezeichne V_φ das Vokabular der CL-Formel φ. Weiterhin sei $I \in \mathcal{IB}_{CL}(V_{CL}, IFm)$ eine CL-Interpretation über V_{CL} und IFm mit $V_\varphi \subseteq V_{CL}$, $I = \langle \mathcal{S}_I, int_I, seq_I, irr_I \rangle$ und $\mathcal{S}_I = \langle UR_I, UD_I, rel_I, fun_I \rangle$.
Es gilt $I \models \varphi$ genau dann, wenn
für alle CL-Modifikationen $Q \in Modi^V_{CL}(I)$ von I bezüglich $V = \{K\}$ gilt
$Q \models (\mathsf{iff}\,(\texttt{Russell}\ K)\,(\mathsf{not}\,(K\ K)))$, nach Definition 4.32.
Analog zu dem Beweis von Lemma 4.57 erhalten wir $I \models \varphi$ genau dann, wenn für alle CL-Modifikationen $Q \in Modi^V_{CL}(I)$ von I bezüglich $V = \{K\}$ gilt
(1) $int_Q(K) \notin rel_I(int_I(\texttt{Russell}))$ und $int_Q(K) \in rel_I(int_Q(K))$ oder
(2) $int_Q(K) \notin rel_I(int_Q(K))$ und $int_Q(K) \in rel_I(int_I(\texttt{Russell}))$.
Da die Bedingungen (1) und (2) für alle CL-Modifikationen Q von I bezüglich $V = \{K\}$ erfüllt sein müssen, existiert eine CL-Modifikation $\widehat{Q}$ mit $int_{\widehat{Q}}(K) = int_I(\texttt{Russell})$. Dann gilt $\widehat{Q} \models \varphi$ genau dann, wenn
(1) $int_I(\texttt{Russell}) \notin rel_I(int_I(\texttt{Russell}))$ und
$int_I(\texttt{Russell}) \in rel_I(int_I(\texttt{Russell}))$ oder
(2) $int_I(\texttt{Russell}) \notin rel_I(int_I(\texttt{Russell}))$ und
$int_I(\texttt{Russell}) \in rel_I(int_I(\texttt{Russell}))$.
Da sowohl die Bedingung (1) als auch die Bedingung (2) unerfüllbar ist, gilt also $\widehat{Q} \not\models \varphi$ und damit $I \not\models \varphi$.

■

Falls die CL-Formel φ aus (4.43) in einem segregierten CL-Dialekt interpretiert wird, dann erhalten wir das folgende Resultat:

Lemma 4.59
Sei φ die CL-Formel aus (4.43). Falls der CL-Name `Russell` *ein CL-Referenzname ist, dann ist φ in segregierten CL-Dialekten erfüllbar.*

Beweis:
Um die Erfüllbarkeit der CL-Formel φ aus (4.43) zu zeigen, konstruieren wir eine CL-Interpretation I, die φ erfüllt. Seien $I \in \mathcal{IB}_{CL}(V_{CL}, IFm)$ eine CL-Interpretation über V_{CL} und IFm mit $I = \langle \mathcal{S}_I, int_I, seq_I, irr_I \rangle$ und $\mathcal{S}_I = \langle UR_I, UD_I, rel_I, fun_I \rangle$ die dazugehörige semantische CL-Struktur. Wir definieren I und $\mathcal{S}_I$ wie folgt:

1. $UR_I = \{a, b, r\}$,
2. $UD_I = \{a, b\}$,
3. $rel_I(a) = \{b\}$, $rel_I(b) = \{a\}$ sowie $rel_I(r) = \{a, b\}$ und

4. $int_I(K) = a, int_I(\texttt{Russell}) = r$

Dann gilt $I \models \varphi$.

■

Die Ursache dafür, dass die CL-Formel φ aus (4.43) erfüllbar ist, falls der CL-Name `Russell` als CL-Referenzname interpretiert wird, liegt darin begründet, dass die CL-Modifikation $Q \in Modi^V_{CL}(I)$ von I bezüglich $V = \{K\}$ mit $int_Q(K) = int_I(\texttt{Russell}) \notin UD_I$ in diesem Fall nicht existiert, da K ein CL-Objektname ist. Falls `Russell` als CL-Referenzname interpretiert wird, dann ist die eigentliche problematische Fragestellung, nämlich ob die Russellsche Klasse selbst ein Element der Russellschen Klasse ist, nicht in Common Logic ausdrückbar, da Formulierungen der Art

$$(\texttt{Russell Russell}) \tag{4.44}$$

in Common Logic nur für CL-Objektnamen zulässig sind[13].

4.9 Weitere Eigenschaften von Common Logic

Wir werden unsere Untersuchungen zu Common Logic mit einem Abschnitt beenden, der weitere Eigenschaften von Common Logic überblicksartig zusammenfasst. Darüber hinaus enthält dieser Teil der Diplomarbeit eine Reihe von offenen Fragen, auf die wir bei unseren Recherchen gestoßen sind.

Aufgrund der Trennung zwischen einem Relationsnamen und seiner relationalen Extension hat das Extensionalitätsaxiom in Common Logic keine Gültigkeit. Dies bedeutet, dass aus der Gleichheit zweier relationaler Extensionen nicht die Identität der zugehörigen Relationsnamen geschlussfolgert werden kann. Allerdings kann das Extensionalitätsaxiom analog zu *SKIF* [38] wie folgt formuliert werden:

$$\begin{array}{l}(\textsf{forall}\ (\texttt{R}_1\ \texttt{R}_2)\ (\textsf{if} \\ \qquad (\textsf{forall}\ (seq)\ (\textsf{iff}\ (\texttt{R}_1\ seq)\ (\texttt{R}_2\ seq))) \\ \qquad (\textsf{equal}\ \texttt{R}_1\ \texttt{R}_2) \\ \quad))\end{array} \tag{4.45}$$

Die Bezeichner $\texttt{R}_1$ und $\texttt{R}_2$ werden dabei als CL-Objektnamen und $seq \in SEQ$ als CL-Sequenzname interpretiert. Somit besteht in Common Logic die Möglichkeit, das Extensionalitätsaxiom für CL-Objektnamen manuell zu fordern.

Da Common Logic im Vergleich zur Prädikatenlogik erster Stufe eine größere syntaktische Freiheit erlaubt, lassen sich in Common Logic zahlreiche Tautologien formulieren, welche in der Prädikatenlogik der ersten Stufe nicht ausdrückbar sind. Die Abbildung 4.4 auf der nächsten Seite zeigt einige dieser Tautologien.

[13] Der Grund dafür sind die zusätzlichen syntaktischen Bedingungen für segregierte CL-Dialekte, welche nach Definition 4.39 auf Seite 68 nachgelesen werden können.

(and)

$(\mathsf{not}\,(\mathsf{or}))$

$(\mathsf{iff}\,(\mathsf{and}\,\psi)\,\psi)$,
wobei $\psi \in Fm(V_{CL})$ eine beliebige CL-Formel bezeichnet

$(\mathsf{iff}\,(\mathsf{and}\,\psi)\,(\mathsf{or}\,\psi))$,
wobei $\psi \in Fm(V_{CL})$ eine beliebige CL-Formel bezeichnet

$(\mathsf{forall}\,(\mathtt{R})\,(\mathsf{if}\,(\mathtt{R}\,a)\,(a\,a)))$,
wobei $\mathtt{R}$ und a zwei CL-Objektnamen sind

$(\mathsf{forall}\,(\mathtt{f})\,(\mathsf{exists}\,(a)\,(\mathsf{equal}\,(\mathtt{f}\,\mathtt{f})\,a)))$
$(\mathsf{forall}\,(\mathtt{f})\,(\mathsf{exists}\,(a)\,(\mathsf{equal}\,(\mathtt{f}\,\mathtt{f}\,\mathtt{f})\,a)))$
$\ldots$,
wobei $\mathtt{f}$ und a zwei CL-Objektnamen sind

Abbildung 4.4: Common Logic Tautologien

Andererseits existieren durchaus logische Formeln, deren Interpretationen in der Prädikatenlogik und in Common Logic voneinander abweichen. So ist beispielsweise die folgende CL-Formel,

$$\begin{array}{l}(\mathsf{if}\\ \quad(\mathsf{and}\,(\mathtt{R}_1\,a)\,(\mathsf{not}\,(\mathtt{R}_2\,a)))\\ \quad(\mathsf{not}\,(\mathsf{forall}\,(x\,y)\,(\mathsf{equal}\,x\,y)))\\ \;)\end{array} \tag{4.46}$$

welche wir aus [69] entnommen haben, eine Tautologie, falls die Bezeichner a, x, y, $\mathtt{R}_1$ und $\mathtt{R}_2$ allesamt CL-Objektnamen sind. Die dazu korrespondierende prädikatenlogische Formel

$$((\mathtt{R}_1\,(a) \wedge \neg \mathtt{R}_2\,(a)) \rightarrow \neg\,(\forall x\,\forall y\,(x \circledcirc y))) \tag{4.47}$$

ist jedoch keine prädikatenlogische Tautologie, sondern lediglich erfüllbar.

Die Semantik von Common Logic unterscheidet sich auch von der Standardsemantik der *Prädikatenlogik höherer Stufe (HOL)* [64]. So fordert die logische Formel

$$\begin{aligned}\neg\exists \mathtt{R}\,(&\\ &\forall x\,\forall y\,\forall z\,((\mathtt{R}\,(x,y)\wedge\mathtt{R}\,(y,z))\rightarrow\mathtt{R}\,(x,z))\wedge\\ &\forall x\,(\neg\mathtt{R}\,(x,x)\wedge\exists y\,\mathtt{R}\,(x,y))\\ &)\end{aligned}\tag{4.48}$$

entnommen aus [23], die Endlichkeit des Universums einer HOL-Interpretation. Die entsprechende CL-Formel

$$\begin{aligned}(\mathsf{not}\,(\mathsf{exists}\,(\mathtt{R})\,(\mathsf{and}&\\ &(\mathsf{forall}\,(x\;y\;z)\,(\mathsf{if}\,(\mathsf{and}\,(\mathtt{R}\;x\;y)\,(\mathtt{R}\;y\;z))\,(\mathtt{R}\;x\;z)))\\ &(\mathsf{forall}\,(x)\,(\mathsf{not}\,(\mathtt{R}\;x\;x)))\\ &(\mathsf{forall}\,(x)\,(\mathsf{exists}\,(y)\,(\mathtt{R}\;x\;y)))\\ &)))\end{aligned}\tag{4.49}$$

verbietet weder die Existenz von CL-Objektnamen mit einer unendlichen relationalen Extension noch die Unendlichkeit des Gegenstandsbereichs.
Des Weiteren ist die CL-Formel

$$(\mathsf{forall}\,(\mathtt{R})\,(\mathtt{R}\;a))\tag{4.50}$$

erfüllbar, falls $\mathtt{R}$ und a zwei CL-Objektnamen sind. Im Gegensatz dazu ist die korrespondierende logische Formel

$$\forall \mathtt{R}\,\mathtt{R}\,(a)\tag{4.51}$$

in der Prädikatenlogik höherer Stufe unter Standardsemantik unerfüllbar.

In Abschnitt 4.5 haben wir gezeigt, dass in Common Logic mithilfe von quantifizierten CL-Sequenznamen unendliche Konjunktionen und unendliche Disjunktionen ausgedrückt werden können. Aufgrund dieser Tatsache ist Common Logic durchaus vergleichbar mit der logischen Sprache $L\,(\omega_1,\omega)$ [11]. Da $L\,(\omega_1,\omega)$ unendliche Konjunktionen und Disjunktionen beliebiger Komplexität erlaubt, ist bisher nicht klar, ob sich in Common Logic lediglich ein Teil der in $L\,(\omega_1,\omega)$ formulierbaren Sachverhalte ausdrücken lässt oder Common Logic die gesamte Ausdrucksstärke von $L\,(\omega_1,\omega)$ umfasst.

Eine weitere offene Frage existiert bezüglich der Kompaktheit von Common Logic. Im Allgemeinen ist Common Logic, wie in Abschnitt 4.6 gezeigt, nicht kompakt. Verzichtet man auf die CL-Sequenznamen, so wird behauptet, dass die dadurch entstehende Teilsprache von Common Logic kompakt sei. Ein formaler Beweis dafür wurde bisher jedoch noch nicht erbracht.

5 RDF und RDF Schema

Dieses Kapitel ist dem *Resource Description Framework (RDF)* und dem darauf basierenden *RDF Schema (RDFS)* vorbehalten. Beide logischen Sprachen sind sogenannte *W3C-Empfehlungen*, die im Rahmen des Semantic Web entwickelt wurden.

Wir werden zunächst die Syntax von RDF und RDFS einführen und danach die formale Semantik der beiden Sprachen definieren. Im dritten Teil des Kapitels wird das RDFS-Universum anhand eines Beispiels graphisch dargestellt und charakterisiert. Anschließend folgt eine detaillierte Beschreibung der sogenannten Reifikation, die eine Besonderheit von RDF und RDFS darstellt. Wir zeigen dabei unter Verwendung eines Beispiels, dass mittels Reifikation logische Aussagen über syntaktische Ausdrücke von RDF/RDFS möglich sind. Zum Abschluss dieses Kapitels übertragen wir RDF und RDFS nach Common Logic, indem wir zeigen, wie die einzelnen Sprachen schrittweise in Common Logic eingebettet werden können.

5.1 Die abstrakte Syntax von RDF und RDFS

Analog zu Common Logic existiert auch für RDF/RDFS eine abstrakte Syntax, für die zahlreiche konkrete Umsetzungen erarbeitet wurden. Als Beispiele seien die graphbasierte Syntax [51], die XML-basierte Syntax *RDF/XML* [9] und die *Terse RDF Triple Language (Turtle)* [10] genannt.

Wir werden in diesem Abschnitt die abstrakten Syntaxkategorien von RDF und RDFS basierend auf [49] einführen und beginnen mit den elementaren Syntaxkategorien.

Definition 5.1 (RDF URI-Referenz)
Eine *RDF URI-Referenz* ist eine Zeichenfolge von Unicode-Schriftzeichen [70], welche eine URI [12] repräsentiert.
Die Menge aller RDF URI-Referenzen bezeichnen wir mit URI_{RDF}.

□

Eine RDF URI-Referenz wie beispielsweise

```
<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
```

werden wir mit den spitzen Klammern < und > umschließen.
Um eine bessere Lesbarkeit zu erreichen, werden wir die RDF URI-Referenzen

durch sogenannte *qualified names (QNamen)* darstellen. Ein QName ist ein Bezeichner der Form `Präfix:Name` bestehend aus einem Namensraumpräfix und einem Namen. Mithilfe dieser Schreibweise lassen sich zum Beispiel die beiden RDF URI-Referenzen

`<http://www.example.org/a>` und `<http://www.example.org/b>`

abkürzend als `ex:a` und `ex:b` schreiben, falls `ex` der Namensraumpräfix der URI-Referenz `<http://www.example.org/>` ist.

Definition 5.2 (ungetyptes RDF-Literal)
Ein *ungetyptes RDF-Literal* ist entweder eine Folge von Unicode-Schriftzeichen [70] oder ein Paar bestehend aus einer Folge von Unicode-Schriftzeichen und einer Sprachkennzeichnung. Für die Menge aller ungetypten RDF-Literale verwenden wir die Bezeichnung $uLit$.

□

Sei `s` eine Sequenz von Unicode-Schriftzeichen und `t` eine Sprachkennzeichnung. Ein ungetyptes RDF-Literal ohne Sprachkennzeichnung `"s"` umklammern wir mit den doppelten Hochkommata `"`. Ein ungetyptes RDF-Literal mit Sprachkennzeichnung `"s"@t` wird dargestellt, indem das `@`-Zeichen und die Sprachkennzeichnung an das ungetypte RDF-Literal angefügt werden.

Neben den ungetypten RDF-Literalen existieren auch getypte RDF-Literale.

Definition 5.3 (getyptes RDF-Literal)
Ein *getyptes RDF-Literal* ist ein Paar dessen erste Komponente eine Sequenz von Unicode-Schriftzeichen und die zweite Komponente eine RDF URI-Referenz ist. Die zweite Komponente wird auch als *Datentyp-URI* bezeichnet.
Die Menge aller getypten RDF-Literale kennzeichnen wir mit $tLit$.

□

Es sei `<http://www.w3.org/2001/XMLSchema#integer>` eine Datentyp-URI und 27 eine Sequenz von Unicode-Schriftzeichen. Das getypte RDF-Literal

`"27"^^http://www.w3.org/2001/XMLSchema#integer`

schreiben wir als eine von doppelten Hochkommata `"` umrandete Sequenz von Unicode-Schriftzeichen, gefolgt von der Zeichenfolge `^^` und einer Datentyp-URI. Die Datentyp-URI kann mithilfe eines QNamens abgekürzt dargestellt werden.

Definition 5.4 (RDF-Literal)
Ein *RDF-Literal* ist entweder ein ungetyptes RDF-Literal oder ein getyptes RDF-Literal. Für die Menge aller RDF-Literale vereinbaren wir die Bezeichnung Lit.

□

Die beiden Mengen URI_{RDF} der RDF URI-Referenzen und Lit der RDF-Literale sind disjunkt zueinander. Die Vereinigung dieser beiden Mengen heißt *R-Vokabular* und wird mit $V_R = URI_{RDF} \cup Lit$ bezeichnet.

Definition 5.5 (leere RDF-Knoten)
Die Menge der *leeren RDF-Knoten* ist eine weitere elementare Syntaxkategorie, die jedoch nicht näher spezifiziert wird.
Wir bezeichnen die Menge der leeren RDF-Knoten mit Var_{RDF}.
□

Die Menge der leeren RDF-Knoten ist disjunkt zu dem R-Vokabular. Da leere RDF-Knoten vergleichbar mit prädikatenlogischen Variablen sind, werden wir leere RDF-Knoten in der Regel mit x, y, x_1, x_2 usw. bezeichnen.

Unter Verwendung der elementaren Syntaxkategorien lassen sich nun komplexere Konstruktionen definieren.

Definition 5.6 (RDF-Tripel)
Sei $V_R = URI_{RDF} \cup Lit$ ein R-Vokabular und Var_{RDF} eine Menge von leeren RDF-Knoten. Ein *RDF-Tripel* $p(s, o)$ besteht aus den folgenden drei Komponenten:

1. dem *Prädikat* $p \in URI_{RDF}$, welches eine RDF URI-Referenz ist,
2. dem *Subjekt* $s \in URI_{RDF} \cup Var_{RDF}$, welches entweder eine RDF URI-Referenz oder ein leerer RDF-Knoten ist und
3. dem *Objekt* $o \in URI_{RDF} \cup Lit \cup Var_{RDF}$, welches entweder eine RDF URI-Referenz, ein RDF-Literal oder ein leerer RDF-Knoten ist.

Für die Menge aller RDF-Tripel über V_R und Var_{RDF} führen wir die Bezeichnung $Tri_{RDF}(V_R, Var_{RDF})$ ein.
□

Des Weiteren werden Mengen von RDF-Tripeln zu sogenannten RDF-Graphen zusammengefasst.

Definition 5.7 (RDF-Graph)
Sei V_R ein R-Vokabular und Var_{RDF} eine Menge von leeren RDF-Knoten. Ein *RDF-Graph* $G \subseteq Tri_{RDF}(V_R, Var_{RDF})$ ist eine Menge von RDF-Tripeln.
□

Die Menge der RDF URI-Referenzen und RDF-Literale, welche in dem RDF-Graph G vorkommen, nennen wir $V_R(G)$. Für die Menge der leeren RDF-Knoten von G führen wir die Bezeichnung $Var_{RDF}(G)$ ein. Weiterhin bezeichnen wir die Menge aller RDF-Graphen über V_R und Var_{RDF} mit $Gr_{RDF}(V_R, Var_{RDF})$.

5.2 Formale Semantik

Nachdem wir die abstrakten Syntaxkategorien von RDF und RDFS eingeführt haben, werden wir nun die formale Semantik dieser Syntaxkategorien nach dem Vorbild von [35] festlegen. Die Definition der Semantik erfolgt schrittweise durch eine

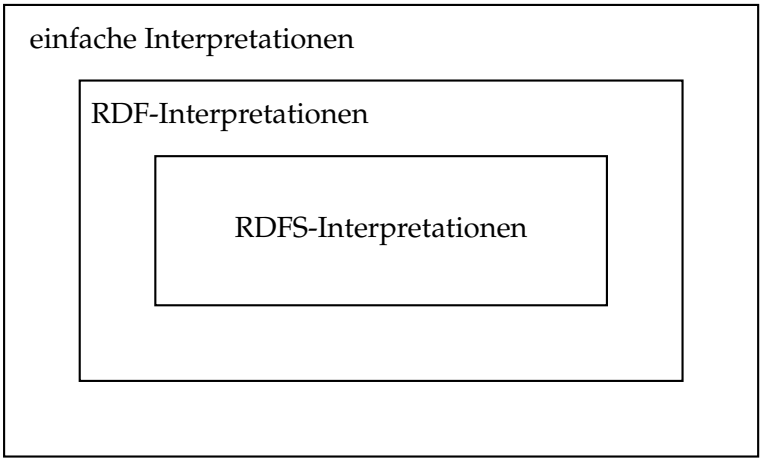

Abbildung 5.1: Beziehung der verschiedenen Interpretationen nach [44]

stetige Verfeinerung der jeweiligen Interpretationen mithilfe von zusätzlichen Bedingungen. Den Ausgangspunkt bilden die einfachen Interpretationen. Die nächste Stufe sind die RDF-Interpretationen, welche auf den einfachen Interpretationen basieren. Durch die Angabe von weiteren Kriterien, welche die RDF-Interpretationen erfüllen müssen, werden schließlich die RDFS-Interpretationen definiert. Eine Folge dieser sukzessiven Definition ist die Tatsache, dass jede RDFS-Interpretation auch eine RDF-Interpretation ist. Des Weiteren lässt sich jede RDF-Interpretation auch als einfache Interpretation charakterisieren. Die Abbildung 5.1 stellt diese Beziehungen zwischen den einzelnen Interpretationen graphisch dar. Es sei angemerkt, dass RDFS-Interpretationen nicht den Abschluss dieser Stufendefinition bilden. In [35] werden RDFS-Interpretationen zu sogenannten D-Interpretationen erweitert, indem zusätzliche Bedingungen spezifiziert werden, die ein Datentyp D erfüllen muss. Auf diese Erweiterung der RDFS-Interpretationen werden wir in dieser Arbeit nicht näher eingehen.

5.2.1 Einfache Interpretationen

Wir führen als erstes die einfachen Interpretationen ein. Die syntaktischen Konstruktionen werden, analog zur Prädikatenlogik und zu Common Logic, mithilfe von mengentheoretischen Strukturen gedeutet. Für einfache Interpretationen und RDF-Interpretationen bezeichnen wir diese Strukturen als semantische RDF-Strukturen.

Definition 5.8 (semantische RDF-Struktur)
Sei $V_R = URI_{RDF} \dot{\cup} Lit$ ein R-Vokabular.
Eine *semantische RDF-Struktur* $\mathcal{S} = \langle U, P, LV, rel \rangle$ über V_R besteht aus:

- Einer nichtleeren Menge U, die *Universum* von $\mathcal{S}$ genannt wird.

- Einer Menge P, die als *Menge der Propertys* von $\mathcal{S}$ bezeichnet wird.
- Einer Menge $LV \subseteq U$, die *Menge der Literalwerte* genannt wird und alle ungetypten RDF-Literale aus V_R enthält, also $uLit \subseteq LV$.
- Einer Abbildung $rel : P \to \mathcal{P}(U \times U)$.

□

Für semantische RDF-Strukturen führen wir nun die Identitätsrelation ein.

Definition 5.9 (Identität zweier semantischer RDF-Strukturen)
Es sei V_R ein R-Vokabular. Des Weiteren seien $\mathcal{S}_1 = \langle U_1, P_1, LV_1, rel_1 \rangle$ und $\mathcal{S}_2 = \langle U_2, P_2, LV_2, rel_2 \rangle$ zwei semantische RDF-Strukturen über V_R.
Die beiden semantischen RDF-Strukturen $\mathcal{S}_1$ und $\mathcal{S}_2$ sind *identisch* genau dann, wenn die nachfolgenden Bedingungen erfüllt sind.

1. $U_1 = U_2$,
2. $P_1 = P_2$,
3. $LV_1 = LV_2$ und
4. $rel_1(p) = rel_2(p)$ für alle $p \in P_1$.

□

Falls die beiden semantischen RDF-Strukturen $\mathcal{S}_1$ und $\mathcal{S}_2$ identisch sind, dann verwenden wir dafür die Schreibweise $\mathcal{S}_1 = \mathcal{S}_2$.

Als nächstes erweitern wir die semantischen RDF-Strukturen zu einfachen Interpretationen. Dies erfolgt durch die Einführung von Funktionen, welche die Elemente eines R-Vokabulars und die leeren RDF-Knoten auf eine semantische RDF-Struktur abbilden.

Definition 5.10 (einfache Interpretation)
Es sei V_R ein R-Vokabular und Var_{RDF} eine Menge von leeren RDF-Knoten.
Eine *einfache Interpretation* $I = \langle \mathcal{S}, int^U, int^L, int^V \rangle$ über V_R und Var_{RDF} besteht aus:

- einer semantischen RDF-Struktur $\mathcal{S} = \langle U, P, LV, rel \rangle$,
- einer Abbildung $int^U : URI_{RDF} \to U \cup P$,
- einer Abbildung $int^L : tLit \to U$ und
- einer Abbildung $int^V : Var_{RDF} \to U$.

□

Die Menge der einfachen Interpretationen über dem R-Vokabular V_R und der Menge der leeren RDF-Knoten Var_{RDF} heißt *einfacher Interpretationsbereich* über V_R und Var_{RDF} und wird mit $\mathcal{IB}_e(V_R, Var_{RDF})$ bezeichnet.

Um die Erfüllbarkeitsrelation für einfache Interpretationen formal definieren zu können, führen wir zunächst die nachfolgenden Begriffe ein.

Definition 5.11 (RDF-Modifikation)
Seien $I, K \in \mathcal{IB}_e(V_R, Var_{RDF})$ zwei einfache Interpretationen über V_R und Var_{RDF} mit $I = \langle \mathcal{S}_I, int_I^U, int_I^L, int_I^V \rangle$ und $K = \langle \mathcal{S}_K, int_K^U, int_K^L, int_K^V \rangle$.
Die Interpretation K ist eine *RDF-Modifikation* der Interpretation I, falls die folgenden Bedingungen erfüllt sind:

1. $\mathcal{S}_K = \mathcal{S}_I$, d.h. $\mathcal{S}_K$ und $\mathcal{S}_I$ sind identisch,
2. $int_K^U(u) = int_I^U(u)$ für alle $u \in URI_{RDF}$ und
3. $int_K^L(l) = int_I^L(l)$ für alle $l \in tLit$.

□

Falls K eine RDF-Modifikation von I ist, dann unterscheiden sich die beiden einfachen Interpretationen lediglich darin, wie die leeren RDF-Knoten interpretiert werden. Die Menge aller RDF-Modifikationen der einfachen Interpretation I bezeichnen wir mit $Modi_{RDF}(I)$.

Definition 5.12 (RDF-Interpretationsabbildung)
Sei $V_R = URI_{RDF} \dot\cup Lit$ ein R-Vokabular und Var_{RDF} eine Menge von leeren RDF-Knoten. Weiterhin sei $I \in \mathcal{IB}_e(V_R, Var_{RDF})$ eine einfache Interpretation über V_R und Var_{RDF} mit $I = \langle \mathcal{S}_I, int_I^U, int_I^L, int_I^V \rangle$ und der dazugehörigen semantischen RDF-Struktur $\mathcal{S}_I = \langle U_I, P_I, LV_I, rel_I \rangle$.
Die *RDF-Interpretationsabbildung* $int_I^* : V_R \cup Var_{RDF} \to U_I \cup P_I$ ist eine Abbildung, die wie folgt definiert ist:

1. Sei $\texttt{"s"} \in uLit$ ein ungetyptes RDF-Literal, dann gilt $int_I^*(\texttt{"s"}) = \texttt{s}$.
2. Sei $\texttt{"s"@t} \in uLit$ ein ungetyptes RDF-Literal mit Sprachkennzeichnung, dann gilt $int_I^*(\texttt{"s"@t}) = \langle \texttt{s}, \texttt{t} \rangle$.
3. Sei $\texttt{l} \in tLit$ ein getyptes RDF-Literal, dann gilt $int_I^*(\texttt{l}) = int_I^L(\texttt{l})$.
4. Sei $u \in URI_{RDF}$ eine RDF URI-Referenz, dann gilt $int_I^*(u) = int_I^U(u)$.
5. Sei $x \in Var_{RDF}$ ein leerer RDF-Knoten, dann gilt $int_I^*(x) = int_I^V(x)$.

□

Die RDF-Interpretationsabbildung int_I^* bildet also jedes Element der elementaren Syntaxkategorien auf ein Element des Universums oder auf eine Property ab. Eine RDF-Interpretationsabbildung wird eindeutig durch die Abbildungen int_I^L, int_I^U und int_I^V festgelegt.

Die Interpretation der komplexen Syntaxkategorien (die RDF-Tripel und RDF-Graphen) wird mithilfe einer Erfüllbarkeitsrelation definiert.

Definition 5.13 (RDF-Erfüllbarkeitsrelation)
Sei $I \in \mathcal{IB}_e(V_R, Var_{RDF})$ eine einfache Interpretation über V_R und Var_{RDF} mit $I = \langle \mathcal{S}_I, int_I^U, int_I^L, int_I^V \rangle$ und $\mathcal{S}_I = \langle U_I, P_I, LV_I, rel_I \rangle$ die zu I gehörende semantische RDF-Struktur. Weiterhin bezeichne $int_I^* : V_R \cup Var_{RDF} \to U_I \cup P_I$ die durch I eindeutig festgelegte RDF-Interpretationsabbildung.
Die *RDF-Erfüllbarkeitsrelation* $\models$ ist wie folgt definiert:

1. Sei $\varphi = p(s,o)$ ein RDF-Tripel. Dann gilt $I \models \varphi$ genau dann, wenn $p \in URI_{RDF}, s \in URI_{RDF} \cup Var_{RDF}, o \in URI_{RDF} \cup Var_{RDF} \cup Lit, int_I^*(p) \in P_I$ und $\langle int_I^*(s), int_I^*(o) \rangle \in rel_I(int_I^*(p))$.

2. Sei $G = \{t_1, t_2, \ldots, t_n\}$ ein RDF-Graph. Dann gilt $I \models G$ genau dann, wenn eine RDF-Modifikation $K \in Modi_{RDF}(I)$ von I existiert mit $K \models t_i$ für alle $1 \leq i \leq n$.

□

5.2.2 RDF-Interpretationen

Nachdem wir im vorangegangenen Abschnitt die einfachen Interpretationen formal eingeführt haben, werden wir diese nun zu RDF-Interpretationen erweitern. Einfache Interpretationen und RDF-Interpretationen unterscheiden sich in der Interpretation einiger RDF URI-Referenzen. Während einfache Interpretationen alle RDF URI-Referenzen gleich behandeln, werden in RDF-Interpretationen einige RDF URI-Referenzen auf besondere Weise interpretiert.
Diese RDF URI-Referenzen werden als RDF-Vokabular bezeichnet.

Definition 5.14 (RDF-Vokabular)
Es sei `rdf` der Namensraumpräfix der URI-Referenz
`<http://www.w3.org/1999/02/22-rdf-syntax-ns#>`.
Das *RDF-Vokabular* V_{RDF} ist eine unendliche Menge von RDF URI-Referenzen, die wie folgt definiert ist:

$V_{RDF} = \{$ `rdf:type, rdf:Property, rdf:XMLLiteral, rdf:nil, rdf:List, rdf:Statement, rdf:subject, rdf:predicate, rdf:object, rdf:first, rdf:rest, rdf:Seq, rdf:Bag, rdf:Alt, rdf:value` $\} \cup \{$`rdf:_i`$|$ für alle $i \in \{1,2,\ldots\}\}$.

□

Bevor wir die RDF-Interpretationen formal definieren können, führen wir den Begriff Datentyp nach Sichtweise von RDF ein.

Definition 5.15 (Datentyp)
Ein *Datentyp* $D = \langle L, W, L2W \rangle$ ist ein Tripel, welches aus den folgenden Komponenten besteht:

- einer nichtleeren Menge L, die *lexikalischer Bereich* genannt wird,

- einer nichtleeren Menge W, dem *Wertebereich* und
- einer Abbildung $L2W : L \to W$.

□

Der lexikalische Bereich eines Datentyps D enthält alle Zeichenfolgen, die als syntaktische Konstruktionen die Werte von D repräsentieren. Falls eine Zeichenkette s zu dem lexikalischen Bereich von D gehört, dann wird diese als *wohlgeformtes* Element von D bezeichnet. Andernfalls ist s ein *nicht wohlgeformtes* Element von D. Die Abbildung $L2W$ ordnet jedem wohlgeformten Element von D ein Element aus dem Wertebereich zu. Ein Datentyp in RDF und RDFS wird unter Verwendung einer RDF URI-Referenz identifiziert.

Die RDF-Interpretationen besitzen genau einen vordefinierten Datentypen, welcher mit der RDF URI-Referenz `rdf:XMLLiteral` angesprochen wird. Dieser Datentyp ist in [49] formal definiert und erlaubt die Verwendung von XML-Fragmenten als Wertebezeichner. Die Elemente des Wertebereichs von `rdf:XMLLiteral` werden als *XML-Werte* bezeichnet.

Nun sind wir in der Lage die einfachen Interpretationen zu RDF-Interpretationen zu erweitern.

Definition 5.16 (RDF-Interpretation)
Es sei V_R ein R-Vokabular und Var_{RDF} eine Menge von leeren RDF-Knoten.
Eine *RDF-Interpretation* $I = \langle \mathcal{S}, int^U, int^L, int^V \rangle$ über V_R und Var_{RDF} mit der semantischen RDF-Struktur $\mathcal{S} = \langle U, P, LV, rel \rangle$ ist eine einfache Interpretation über $V_R \cup V_{RDF}$ und Var_{RDF}, welche zusätzlich die nachfolgenden Bedingungen erfüllt:

1. $x \in P$ genau dann, wenn $\langle x, int^*(\texttt{rdf:Property}) \rangle \in rel\,(int^*(\texttt{rdf:type}))$
2. falls `"s"^^rdf:XMLLiteral` $\in V_R$ und `s` ein wohlgeformtes XML-Literal ist, dann gilt:
   - int^L (`"s"^^rdf:XMLLiteral`) ist der XML-Wert von `s`,
   - int^L (`"s"^^rdf:XMLLiteral`) $\in LV$ und
   - $\langle int^L(\texttt{"s"\^\^rdf:XMLLiteral}), int^*(\texttt{rdf:XMLLiteral}) \rangle \in rel(int^*(\texttt{rdf:type}))$
3. falls `"s"^^rdf:XMLLiteral` $\in V_R$ und `s` ein nicht wohlgeformtes XML-Literal ist, dann gilt:
   - int^L (`"s"^^rdf:XMLLiteral`) $\notin LV$ und
   - $\langle int^L(\texttt{"s"\^\^rdf:XMLLiteral}), int^*(\texttt{rdf:XMLLiteral}) \rangle \notin rel(int^*(\texttt{rdf:type}))$
4. I erfüllt die *axiomatischen RDF-Tripel* aus Abbildung 5.2 auf der nächsten Seite.

□

Die Menge der RDF-Interpretationen über dem R-Vokabular V_R und der Menge der leeren RDF-Knoten Var_{RDF} heißt *RDF-Interpretationsbereich* über V_R und Var_{RDF} und wird mit $\mathcal{IB}_{RDF}(V_R, Var_{RDF})$ bezeichnet.

rdf:type(rdf:type, rdf:Property)
rdf:type(rdf:subject, rdf:Property)
rdf:type(rdf:predicate, rdf:Property)
rdf:type(rdf:object, rdf:Property)
rdf:type(rdf:first, rdf:Property)
rdf:type(rdf:rest, rdf:Property)
rdf:type(rdf:value, rdf:Property)
rdf:type(rdf:_i, rdf:Property) für alle $i \in \{1, 2, \ldots\}$
rdf:type(rdf:nil, rdf:List)

Abbildung 5.2: Die axiomatischen RDF-Tripel

5.2.3 RDFS-Interpretationen

Die Semantik von RDF werden wir in diesem Abschnitt durch die Einführung der RDFS-Interpretationen verfeinern. RDFS-Interpretationen erweitern das RDF-Vokabular um zusätzliche RDF URI-Referenzen, die eine spezielle Bedeutung in RDFS-Interpretationen besitzen.
Diese RDF URI-Referenzen nennt man RDFS-Vokabular.

Definition 5.17 (RDFS-Vokabular)
Es sei rdfs der Namensraumpräfix der URI-Referenz
<http://www.w3.org/2000/01/rdf-schema#>.
Das *RDFS-Vokabular* V_{RDFS} ist eine Menge von RDF URI-Referenzen, die folgendermaßen definiert ist:

$V_{RDFS} = \{$ rdfs:domain, rdfs:range, rdfs:Resource,rdfs:Literal,
rdfs:Datatype, rdfs:Class, rdfs:subClassOf,
rdfs:subPropertyOf, rdfs:member, rdfs:Container,
rdfs:ContainerMembershipProperty, rdfs:comment,
rdfs:seeAlso, rdfs:isDefinedBy, rdfs:label
$\}$.

□

Ein grundlegender Unterschied zwischen RDF und RDFS besteht darin, dass in RDFS nicht nur zweistellige Relationen (die sogenannten Propertys) existieren, sondern zusätzlich die Modellierung von einstelligen Relationen möglich ist, welche als *Klassen* bezeichnet werden. Die Einführung der Klassen erfordert eine Erweiterung der semantischen RDF-Strukturen.
Diese erweiterten Strukturen werden semantische RDFS-Strukturen genannt.

Definition 5.18 (semantische RDFS-Struktur)
Sei $V_R = URI_{RDF} \dot{\cup} Lit$ ein R-Vokabular.
Eine *semantische RDFS-Struktur* $\mathcal{S} = \langle \overline{\mathcal{S}}, K, Kext \rangle$ besteht aus:

- Einer semantischen RDF-Struktur $\overline{\mathcal{S}} = \langle U, P, LV, rel \rangle$.
- Einer Menge K, die *Menge der Klassen* von $\mathcal{S}$ genannt wird.
- Einer Abbildung $Kext : K \to \mathcal{P}(U)$.

□

Die Abbildung *Kext* ordnet jeder Klasse diejenigen Elemente des Universums zu, die zu der jeweiligen Klasse gehören.
Unter Verwendung der semantischen RDFS-Strukturen sind wir nun in der Lage, die RDFS-Interpretationen zu definieren.

Definition 5.19 (RDFS-Interpretation)
Es sei V_R ein R-Vokabular und Var_{RDF} eine Menge von leeren RDF-Knoten.
Eine *RDFS-Interpretation* $I = \langle \mathcal{S}, int^U, int^L, int^V \rangle$ über V_R und Var_{RDF} erfüllt die folgenden Bedingungen:

1. $\mathcal{S} = \langle \overline{\mathcal{S}}, K, Kext \rangle$ mit $\overline{\mathcal{S}} = \langle U, P, LV, rel \rangle$ ist eine semantische RDFS-Struktur.
2. $\langle \overline{\mathcal{S}}, int^U, int^L, int^V \rangle$ ist eine RDF-Interpretation über $V_R \cup V_{RDFS}$ und Var_{RDF}.
3. $x \in Kext(y)$ genau dann, wenn $\langle x, y \rangle \in rel(int^*(\texttt{rdf:type}))$
4. $K = Kext(int^*(\texttt{rdfs:Class}))$
5. $U = Kext(int^*(\texttt{rdfs:Resource}))$
6. $LV = Kext(int^*(\texttt{rdfs:Literal}))$
7. Falls $\langle x, y \rangle \in rel(int^*(\texttt{rdfs:domain}))$ und $\langle u, v \rangle \in rel(x)$, dann $u \in Kext(y)$.
8. Falls $\langle x, y \rangle \in rel(int^*(\texttt{rdfs:range}))$ und $\langle u, v \rangle \in rel(x)$, dann $v \in Kext(y)$.
9. $rel(int^*(\texttt{rdfs:subPropertyOf}))$ ist transitiv und reflexiv auf der Menge P.
10. Falls $\langle x, y \rangle \in rel(int^*(\texttt{rdfs:subPropertyOf}))$, dann $x, y \in P$ und $rel(x) \subseteq rel(y)$.
11. Falls $x \in K$, dann $\langle x, int^*(\texttt{rdfs:Resource}) \rangle \in rel(int^*(\texttt{rdfs:subClassOf}))$.
12. Falls $\langle x, y \rangle \in rel(int^*(\texttt{rdfs:subClassOf}))$, dann $x, y \in K$ und $Kext(x) \subseteq Kext(y)$.
13. $rel(int^*(\texttt{rdfs:subClassOf}))$ ist transitiv und reflexiv auf der Menge K.
14. Falls $x \in Kext(int^*(\texttt{rdfs:ContainerMembershipProperty}))$, dann $\langle x, int^*(\texttt{rdfs:member}) \rangle \in rel(int^*(\texttt{rdfs:subPropertyOf}))$.
15. Falls $x \in Kext(int^*(\texttt{rdfs:Datatype}))$, dann $\langle x, int^*(\texttt{rdfs:Literal}) \rangle \in rel(int^*(\texttt{rdfs:subClassOf}))$.

16. I erfüllt die *axiomatischen RDFS-Tripel* aus Abbildung 5.3 auf Seite 103 und Abbildung 5.4 auf Seite 104.

□

Die Menge der RDFS-Interpretationen über dem R-Vokabular V_R und der Menge der leeren RDF-Knoten Var_{RDF} heißt *RDFS-Interpretationsbereich* über V_R und Var_{RDF} und wird mit $\mathcal{IB}_{RDFS}(V_R, Var_{RDF})$ bezeichnet.

Es sei angemerkt, dass sowohl die Menge K als auch die Abbildung $Kext$ einer RDFS-Interpretation I eindeutig durch I und die zu I gehörende semantische RDF-Struktur festgelegt sind.
Insbesondere gilt das nachfolgende Lemma.

Lemma 5.20
Seien $I, M \in \mathcal{IB}_{RDFS}(V_R, Var_{RDF})$ zwei RDFS-Interpretationen über V_R und der Menge der leeren RDF-Knoten Var_{RDF} mit $I = \langle\langle \mathcal{S}_I, K_I, Kext_I\rangle, int_I^U, int_I^L, int_I^V\rangle$ und $M = \langle\langle \mathcal{S}_M, K_M, Kext_M\rangle, int_M^U, int_M^L, int_M^V\rangle$.
Falls $\mathcal{S}_I = \mathcal{S}_M$ sowie $int_I^U(u) = int_M^U(u)$ für alle $u \in URI_{RDF}$ und $int_I^L(l) = int_M^L(l)$ für alle $l \in tLit$, dann gilt auch $K_I = K_M$ und $Kext_I(k) = Kext_M(k)$ für alle $k \in K_I$.

Beweis:
Die zu I und M gehörenden semantischen RDF-Strukturen $\mathcal{S}_I$ und $\mathcal{S}_M$ seien definiert als $\mathcal{S}_I = \langle U_I, P_I, LV_I, rel_I\rangle$ und $\mathcal{S}_M = \langle U_M, P_M, LV_M, rel_M\rangle$.
Wir zeigen nun: $Kext_I(k) = Kext_M(k)$ für alle $k \in U_I$.
Seien $x, k \in U_I$. Dann gilt: $x \in Kext_I(k)$
gdw. $\langle x, k\rangle \in rel_I(int_I^*(\texttt{rdf:type}))$, nach 3. Bedingung von Definition 5.19
gdw. $\langle x, k\rangle \in rel_I(int_I^U(\texttt{rdf:type}))$, nach Definition 5.12
gdw. $\langle x, k\rangle \in rel_M(int_M^U(\texttt{rdf:type}))$, da $int_I^U(\texttt{rdf:type}) = int_M^U(\texttt{rdf:type})$ und $rel_I(p) = rel_M(p)$ für alle $p \in P_I = P_M$
gdw. $\langle x, k\rangle \in rel_M(int_M^*(\texttt{rdf:type}))$, nach Definition 5.12
gdw. $x \in Kext_M(k)$, nach 3. Bedingung von Definition 5.19
Also $Kext_I(k) = Kext_M(k)$ für alle $k \in U_I$.
Nun beweisen wir: $K_I = K_M$
Es sei $x \in K_I$. Dann gilt: $x \in K_I$
gdw. $x \in Kext_I(int_I^*(\texttt{rdfs:Class}))$, nach 4. Bedingung von Definition 5.19
gdw. $x \in Kext_I(int_I^U(\texttt{rdfs:Class}))$, nach Definition 5.12
gdw. $x \in Kext_M(int_M^U(\texttt{rdfs:Class}))$, da $int_I^U(\texttt{rdfs:Class}) = int_M^U(\texttt{rdfs:Class})$ und $Kext_I(k) = Kext_M(k)$ für alle $k \in U_I = U_M$
gdw. $x \in Kext_M(int_M^*(\texttt{rdfs:Class}))$, nach Definition 5.12
gdw. $x \in K_M$, nach 4. Bedingung von Definition 5.19
Also $K_I = K_M$.

■

Aufgrund von Lemma 5.20 ist die Definition 5.13 (RDF-Erfüllbarkeitsrelation) auch für RDFS-Interpretationen wohldefiniert.

Abschließend legen wir die Folgerungsrelation für RDFS-Interpretationen wie folgt fest:

Definition 5.21 (Modellmenge, Folgerungsrelation)
Sei V_R ein R-Vokabular und Var_{RDF} eine Menge von leeren RDF-Knoten. Weiterhin seien $G_1, G_2 \in Gr_{RDF}(V_R, Var_{RDF})$ zwei RDF-Graphen über V_R und Var_{RDF}.

(a) Die *RDFS-Modellmenge* eines RDF-Graphen G_1 definieren wir wie folgt:
$Mod_{RDFS}(G_1) = \{I \in \mathcal{IB}_{RDFS}(V_R, Var_{RDF}) \mid I \models G_1\}$.

(b) Die *RDFS-Folgerungsrelation* $\models_{RDFS}$ ist definiert als:
$G_1 \models_{RDFS} G_2$ genau dann, wenn $Mod_{RDFS}(G_1) \subseteq Mod_{RDFS}(G_2)$.

□

5.3 Charakterisierung des RDFS-Universums

Der Aufbau des RDFS-Universums steht im Mittelpunkt dieses Unterkapitels. Bevor wir das RDFS-Universum anhand eines Beispiels graphisch darstellen, werden wir zunächst einige allgemeine Eigenschaften des RDFS-Universums zeigen.
Für RDFS-Interpretationen gilt das folgende Lemma:

Lemma 5.22
Sei $I \in \mathcal{IB}_{RDFS}(V_R, Var_{RDF})$ eine RDFS-Interpretation über V_R und Var_{RDF} mit $I = \langle\langle \mathcal{S}, K, Kext\rangle, int^U, int^L, int^V\rangle$ und der semantischen RDF-Struktur $\mathcal{S} = \langle U, P, LV, rel\rangle$. Dann gelten die folgenden Eigenschaften:

(a) $P = Kext(int^*(\texttt{rdf:Property}))$,

(b) $P \subseteq U$ *und*

(c) $K \subseteq U$

Beweis:
(a) Behauptung: $P = Kext(int^*(\texttt{rdf:Property}))$
Es gilt: $x \in Kext(int^*(\texttt{rdf:Property}))$
gdw. $\langle x, int^*(\texttt{rdf:Property})\rangle \in rel(int^*(\texttt{rdf:type}))$,
nach 3. Bedingung von Definition 5.19
gdw. $x \in P$, nach 1. Bedingung von Definition 5.16.
(b) Behauptung: $P \subseteq U$
Es gilt: $x \in P$
gdw. $\langle x, int^*(\texttt{rdf:Property})\rangle \in rel(int^*(\texttt{rdf:type}))$,
nach 1. Bedingung von Definition 5.16.
Da die Abbildung *rel* definiert ist als $rel : P \to \mathcal{P}(U \times U)$ folgt $x \in U$.
(c) Behauptung: $K \subseteq U$

```
rdfs:domain(rdf:type, rdfs:Resource)
rdfs:domain(rdfs:domain, rdf:Property)
rdfs:domain(rdfs:range, rdf:Property)
rdfs:domain(rdfs:subPropertyOf, rdf:Property)
rdfs:domain(rdfs:subClassOf, rdfs:Class)
rdfs:domain(rdf:subject, rdf:Statement)
rdfs:domain(rdf:predicate, rdf:Statement)
rdfs:domain(rdf:object, rdf:Statement)
rdfs:domain(rdfs:member, rdfs:Resource)
rdfs:domain(rdf:first, rdf:List)
rdfs:domain(rdf:rest, rdf:List)
rdfs:domain(rdfs:seeAlso, rdfs:Resource)
rdfs:domain(rdfs:isDefinedBy, rdfs:Resource)
rdfs:domain(rdfs:comment, rdfs:Resource)
rdfs:domain(rdfs:label, rdfs:Resource)
rdfs:domain(rdf:value, rdfs:Resource)

rdfs:range(rdf:type, rdfs:Class)
rdfs:range(rdfs:domain, rdfs:Class)
rdfs:range(rdfs:range, rdfs:Class)
rdfs:range(rdfs:subPropertyOf, rdf:Property)
rdfs:range(rdfs:subClassOf, rdfs:Class)
rdfs:range(rdf:subject, rdfs:Resource)
rdfs:range(rdf:predicate, rdfs:Resource)
rdfs:range(rdf:object, rdfs:Resource)
rdfs:range(rdfs:member, rdfs:Resource)
rdfs:range(rdf:first, rdfs:Resource)
rdfs:range(rdf:rest, rdf:List)
rdfs:range(rdfs:seeAlso, rdfs:Resource)
rdfs:range(rdfs:isDefinedBy, rdfs:Resource)
rdfs:range(rdfs:comment, rdfs:Literal)
rdfs:range(rdfs:label, rdfs:Literal)
rdfs:range(rdf:value, rdfs:Resource)
```

Abbildung 5.3: Die axiomatischen RDFS-Tripel – Teil 1

rdfs:subClassOf(rdf:Alt, rdfs:Container)
rdfs:subClassOf(rdf:Bag, rdfs:Container)
rdfs:subClassOf(rdf:Seq, rdfs:Container)

rdfs:subClassOf (rdfs:ContainerMembershipProperty,
rdf:Property)

rdfs:subPropertyOf(rdfs:isDefinedBy, rdfs:seeAlso)

rdf:type(rdf:XMLLiteral, rdfs:Datatype)
rdfs:subClassOf(rdf:XMLLiteral, rdfs:Literal)
rdfs:subClassOf(rdfs:Datatype, rdfs:Class)

rdf:type(rdf:_i, rdfs:ContainerMembershipProperty)
für alle $i \in \{1, 2, \ldots\}$

rdfs:domain(rdf:_i, rdfs:Resource)
für alle $i \in \{1, 2, \ldots\}$

rdfs:range(rdf:_i, rdfs:Resource)
für alle $i \in \{1, 2, \ldots\}$

Abbildung 5.4: Die axiomatischen RDFS-Tripel – Teil 2

Es gilt: $x \in K$
gdw. $x \in Kext\,(int^*\,(\texttt{rdfs:Class}))$, nach 4. Bedingung von Definition 5.19.
Da die Abbildung $Kext$ definiert ist als $Kext : K \to \mathcal{P}\,(U)$ folgt $x \in U$.

■

Für den Beweis von (b) wird eine Bedingung der RDF-Interpretationen verwendet, daher ist die Behauptung $P \subseteq U$ für einfache Interpretationen im Allgemeinen nicht gültig.

Betrachten wir nun ein Beispiel um den Aufbau des RDFS-Universums mit dessen Hilfe zu veranschaulichen.

Beispiel 5.23
Es sei $V_R = URI_{RDF} \cup Lit$ ein R-Vokabular bestehend aus $URI_{RDF} = \{\texttt{ex:a}\}$, $uLit = \{\texttt{"s"}\}$ und $tLit = \{\texttt{1}\}$. Die Menge der leeren RDF-Knoten sei definiert als $Var_{RDF} = \{x\}$.
Die RDFS-Interpretation $I \in \mathcal{IB}_{RDFS}(V_R, Var_{RDF})$ über V_R und Var_{RDF} mit $I = \langle \mathcal{S}, int^U, int^L, int^V \rangle$ und die zu I gehörende semantische RDFS-Struktur $\mathcal{S} = \langle \langle U, P, LV, rel \rangle, K, Kext \rangle$ seien wie folgt definiert:

1. $U = \{\texttt{s}, u_1, u_2, u_3, u_4, u_5, u_6\}$
2. $P = \{u_2\}$
3. $LV = \{\texttt{s}, u_1\}$
4. $K = \{u_3, u_4, u_5, u_6\}$
5. die Abbildung $rel : P \to \mathcal{P}\,(U \times U)$ ist definiert als:
   - $rel\,(u_2) = \{\langle u_1, u_2 \rangle, \langle u_1, u_3 \rangle\}$
6. die Abbildung $Kext : K \to \mathcal{P}\,(U)$ ist definiert als:
   - $Kext\,(u_3) = \{u_2\}$
   - $Kext\,(u_4) = \{\texttt{s}, u_1, u_2, u_3, u_4, u_5, u_6\}$
   - $Kext\,(u_5) = \{\texttt{s}, u_1\}$
   - $Kext\,(u_6) = \{u_3, u_4, u_5, u_6\}$
7. die Abbildung $int^U : URI_{RDF} \to U \cup P$ ist definiert als:
   - $int^U\,(\texttt{ex:a}) = u_2$
8. die Abbildung $int^L : tLit \to U$ ist definiert als:
   - $int^L\,(\texttt{1}) = u_1$
9. die Abbildung $int^V : Var_{RDF} \to U$ ist definiert als:
   - $int^V\,(x) = u_2$

□

Das RDFS-Universum der RDFS-Interpretation aus Beispiel 5.23 wird in Abbildung 5.5 auf Seite 108 dargestellt. In RDF und RDFS existiert in Übereinstimmung mit Common Logic keine syntaktische Trennung zwischen Funktions-, Relations- und Konstantenbezeichnern. Stattdessen wird zwischen leeren RDF-Knoten, RDF-Literalen und RDF URI-Referenzen unterschieden. All diese Syntaxkategorien können sowohl Individuen als auch Relationen bezeichnen. Im Gegensatz zur Prädikatenlogik der ersten Stufe und Common Logic existieren in RDF und RDFS keine Funktionen.

Die Interpretation der einzelnen Bezeichner erfolgt in zwei Etappen. Zunächst werden sowohl die leeren RDF-Knoten als auch die RDF URI-Referenzen und die RDF-Literale in das Universum U abgebildet, so dass U sowohl Individuen als auch Relationen enthält. Darüber hinaus ist es in RDFS möglich, dass das Universum auch RDF-Tripel enthält[14].

Im zweiten Interpretationsschritt werden den Relationen die jeweiligen Extensionen zugewiesen. Im Gegensatz zu Common Logic wird dabei für jede Stelligkeit eine eigene Abbildung verwendet. Die relationale Extension der Klassen wird durch die Funktion $Kext$ festgelegt. In Abbildung 5.5 ist dies mithilfe von Strichlinien dargestellt. In gleicher Weise ordnet die Funktion rel jedem Property eine Extension zu. In der graphischen Darstellung des RDFS-Universums haben wir die Extension der Propertys durch Punktlinien skizziert. Da die Menge der Klassen und die Menge der Propertys nicht disjunkt sein müssen, existieren in RDFS durchaus Relationen mit einer variablen Stelligkeit.

Neben der eigentlichen Menge der RDF URI-Referenzen URI_{RDF} sind die beiden Mengen V_{RDF} und V_{RDFS}, die ebenfalls RDF URI-Referenzen beinhalten, Bestandteile einer RDFS-Interpretation. Die Elemente dieser beiden Mengen werden ebenfalls unter Verwendung der Funktion int^U in die Menge U abgebildet. Die Semantik dieser RDF URI-Referenzen ist jedoch durch zusätzliche Bedingungen der RDF-Interpretationen und RDFS-Interpretationen eindeutig festgelegt. So ist die relationale Extension der Klasse `rdf:Property` stets identisch mit der Menge P. In Abbildung 5.5 haben wir die Interpretation der Elemente von V_{RDF} und V_{RDFS} anhand einiger sehr wichtiger Klassenbezeichner angedeutet.

Abschließend werden wir die leeren RDF-Knoten genauer betrachten. Ein leerer RDF-Knoten ist semantisch vergleichbar mit einer Variablen, die durch einen Existenzquantor gebunden ist. Im Unterschied zur Prädikatenlogik und zu Common Logic erfolgt diese Quantifizierung implizit, da keine Quantorensymbole in RDFS auftreten. Der Wirkungsbereich eines impliziten Existenzquantors umfasst in RDFS das gesamte Universum. Da jeder leere RDF-Knoten in RDFS existenzquantifiziert ist, treten in RDFS keine freien Variablen auf. Weiterhin ist es zulässig, dass leere RDF-Knoten nicht nur Individuen, sondern auch Relationen bezeichnen. Damit

[14] Diese Eigenschaft von RDFS wird durch die sogenannte *Reifikation* verursacht, die wir im nachfolgenden Kapitel näher untersuchen werden.

sind auch Quantifizierungen über Relationen in RDF und RDFS möglich.

5.4 Reifikation

Die Interpretationen der abstrakten Syntaxkategorien von RDF/RDFS können unter Verwendung von zusätzlichen Bedingungen noch weiter eingeschränkt werden. Die *Reifikation* ist eine solche semantische Erweiterung, welche zusätzliche Forderungen für RDF-Interpretationen und somit auch für RDFS-Interpretationen definiert. Wir bezeichnen derartige Interpretationen als Reif-Interpretationen.

Definition 5.24 (Reif-Interpretation)
Es sei V_R ein R-Vokabular und Var_{RDF} eine Menge von leeren RDF-Knoten.
Eine *Reif-Interpretation* $I = \langle \mathcal{S}, int^U, int^L, int^V \rangle$ über V_R und Var_{RDF} ist eine RDF-Interpretation über V_R und Var_{RDF}, welche die folgenden Bedingungen erfüllt:

1. Falls $x \in Kext\left(int^*\left(\texttt{rdf:Statement}\right)\right)$, dann $x \in Tri_{RDF}\left(V_R, Var_{RDF}\right)$.
2. Falls $\langle x, y \rangle \in rel\left(int^*\left(\texttt{rdf:subject}\right)\right)$, dann $y = int^*\left(s\right)$ und $x = p\left(s, o\right)$ mit $p\left(s, o\right) \in Tri_{RDF}\left(V_R, Var_{RDF}\right)$.
3. Falls $\langle x, y \rangle \in rel\left(int^*\left(\texttt{rdf:predicate}\right)\right)$, dann $y = int^*\left(p\right)$ und $x = p\left(s, o\right)$ mit $p\left(s, o\right) \in Tri_{RDF}\left(V_R, Var_{RDF}\right)$.
4. Falls $\langle x, y \rangle \in rel\left(int^*\left(\texttt{rdf:object}\right)\right)$, dann $y = int^*\left(o\right)$ und $x = p\left(s, o\right)$ mit $p\left(s, o\right) \in Tri_{RDF}\left(V_R, Var_{RDF}\right)$.

□

Unter Verwendung der Reifikation lassen sich Aussagen über RDF-Tripel formulieren. Dadurch kann man in RDF/RDFS auch Annahmen oder Wünsche von Personen ausdrücken, wie wir anhand des nachfolgenden Beispiels zeigen werden.

Beispiel 5.25
Erinnern wir uns an das Beispiel 2.16 auf Seite 20, welches eine kleine Ontologie über die familiären Beziehungen beschreibt. Angenommen die Person `Thomas` wünscht sich einen Bruder, dann kann dieser Sachverhalt mittels Reifikation wie folgt ausgedrückt werden:

$$
\begin{aligned}
G = \{ & \\
& \texttt{rdf:type}\left(\texttt{ex:Geschwisterwunsch}, \texttt{rdf:Statement}\right), \\
& \texttt{rdf:subject}\left(\texttt{ex:Geschwisterwunsch}, \texttt{ex:Lukas}\right), \\
& \texttt{rdf:predicate}\left(\texttt{ex:Geschwisterwunsch}, \texttt{ex:istBruderVon}\right), \\
& \texttt{rdf:object}\left(\texttt{ex:Geschwisterwunsch}, \texttt{ex:Thomas}\right), \\
& \texttt{ex:wuenschtSich}\left(\texttt{ex:Thomas}, \texttt{ex:Geschwisterwunsch}\right) \\
\} &
\end{aligned}
\tag{5.1}
$$

□

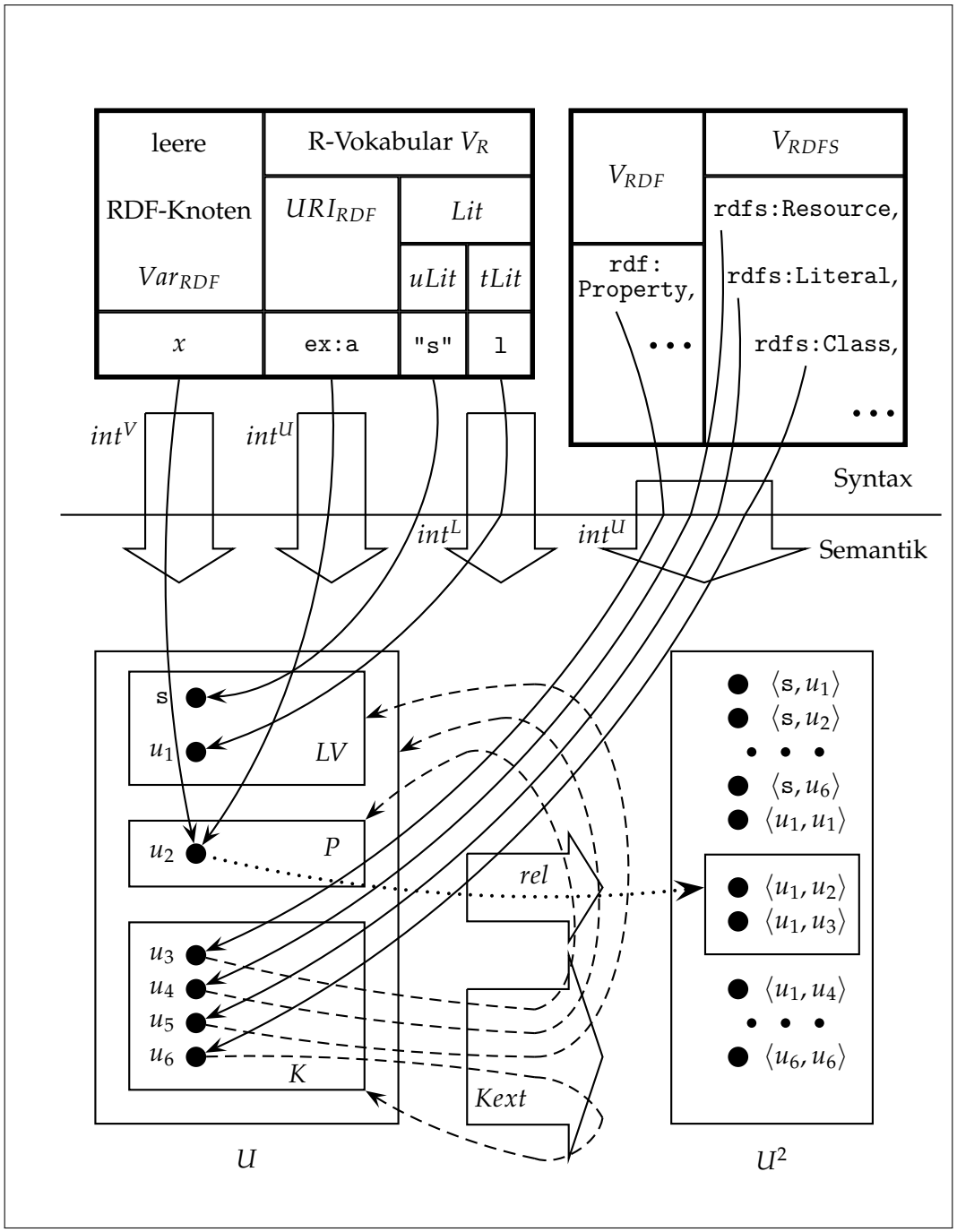

Abbildung 5.5: Schematische Darstellung des RDFS-Universums von Beispiel 5.23

Sehen wir uns nun die formale Semantik dieses Beispiels an.
Sei I eine Reif-Interpretation über $V_R = V_R(G) \cup V_{RDF}$ und $Var_{RDF} = Var_{RDF}(G)$ mit $I = \langle \mathcal{S}, int^U, int^L, int^V \rangle$.
Es gilt $I \models G$ genau dann, wenn die folgenden Bedingungen erfüllt sind:
(1) $int^*(\texttt{ex:Geschwisterwunsch}) = t$, wobei $t = p(s,o)$ ein RDF-Tripel ist,
(2) $int^*(\texttt{ex:Lukas}) = int^*(s)$,
(3) $int^*(\texttt{ex:istBruderVon}) = int^*(p)$,
(4) $int^*(\texttt{ex:Thomas}) = int^*(o)$ und
(5) $\langle int^*(\texttt{ex:Thomas}), t \rangle \in rel(int^*(\texttt{ex:wuenschtSich}))$
Die Person `Thomas` wünscht sich also, dass die Aussage

$$\texttt{ex:istBruderVon}(\texttt{ex:Lukas}, \texttt{ex:Thomas}) \tag{5.2}$$

wahr wird.

Die zusätzlichen Bedingungen einer Reif-Interpretation fordern lediglich, dass ein RDF-Tripel im Universum von I existiert, welches genau so gedeutet wird wie das RDF-Tripel aus (5.2). Es werden jedoch keine Aussagen über die Gültigkeit des RDF-Tripels getroffen. Dadurch ist es nicht möglich aus dem RDF-Graphen G aus Beispiel 5.25 das RDF-Tripel aus (5.2) semantisch zu folgern. Das heißt es ist möglich, dass die Reif-Interpretation I den RDF-Graphen G erfüllt, aber das RDF-Tripel aus (5.2) nicht.

Die Reifikation erweitert nicht nur die RDF-Interpretationen, sondern wirkt sich auch auf den Aufbau des RDFS-Universums aus. Aufgrund der 1. Bedingung von Definition 5.24 besteht das Universum nicht nur aus Individuen und Relationen, sondern es enthält auch RDF-Tripel. Dies ist ein großer Unterschied zur Prädikatenlogik erster Stufe und zu Common Logic. Sowohl die Grundmenge eines prädikatenlogischen Universums als auch der Referenzbereich eines CL-Universums beinhalten keine prädikatenlogischen Formeln bzw. CL-Formeln.

5.5 Übertragung nach Common Logic

Wir werden das Kapitel über RDF und RDFS mit einem Abschnitt beenden, welcher die Beziehungen zwischen RDF/RDFS und Common Logic näher untersucht. Das Ziel dieses Unterkapitels besteht darin, RDF/RDFS-Dokumente unter Beibehaltung ihrer Semantik nach Common Logic abzubilden. Die Überlegungen dieses Abschnitts basieren dabei im Wesentlichen auf den Arbeiten von Patrick Hayes [36] und Christopher Menzel [53].

Da RDF und RDFS spezielle RDF URI-Referenzen für die Modellierung von Wissen bereitstellen, existieren prinzipiell zwei verschiedene Vorgehensweisen den Inhalt von RDF/RDFS-Dokumenten nach Common Logic zu überführen. Einerseits ist es möglich, die speziellen RDF URI-Referenzen in Common Logic zu erhalten

und deren logische Bedeutung mithilfe von Axiomen in Common Logic festzulegen. Andererseits können die RDF URI-Referenzen bei der Übertragung nach Common Logic direkt durch CL-Formeln ersetzt werden, welche die logische Bedeutung der RDF URI-Referenzen beschreiben. Wir entscheiden uns für die erstgenannte Methode, die in [36] als *Einbettung* bezeichnet wird. Bei Anwendung der zweiten Variante spricht man von einer *Übersetzung*.

Bei der Übertragung nach Common Logic werden wir schrittweise vorgehen wie in Abschnitt 5.2. Wir beginnen mit den einfachen Interpretationen.

5.5.1 Einbettung der einfachen Interpretationen

Die Hauptaufgabe bei der Einbettung der einfachen Interpretationen besteht darin, die abstrakten Syntaxkategorien von RDF/RDFS auf äquivalente Syntaxkategorien von Common Logic abzubilden. Dazu führen wir die Abbildung

$$RDFtoCL : Gr_{RDF}\,(V_R, Var_{RDF}) \rightarrow Fm(V_{CL}) \tag{5.3}$$

ein, welche jedem RDF-Graphen eine CL-Formel zuordnet.
Die Abbildung $RDFtoCL$ wird induktiv über den Aufbau der RDF/RDFS-Syntax definiert, so dass wir diese zunächst für die elementaren Syntaxkategorien festlegen.

Eine RDF URI-Referenz ist semantisch betrachtet ein Bezeichner. Deshalb wird jede RDF URI-Referenz in Common Logic als ein interpretierbarer CL-Name behandelt. Bei der Einbettung in einen konkreten CL-Dialekt muss beachtet werden, dass die URI-Referenzen die syntaktischen Bedingungen des jeweiligen CL-Dialekts erfüllen. Weiter ist zu überlegen ob nur vollständige RDF URI-Referenzen oder auch QNamen erlaubt sind. Um eine kompakte Darstellung der RDF URI-Referenzen zu ermöglichen, werden wir die QNamen auch in Common Logic verwenden.
Seien $<aaa>$ eine RDF URI-Referenz und $bb:aaa$ ein QName, dann gilt:

$$\begin{aligned} RDFtoCL\,(<aaa>) &= aaa \qquad \text{und} \\ RDFtoCL\,(bb:aaa) &= bb:aaa \end{aligned} \tag{5.4}$$

Die leeren RDF-Knoten werden ebenfalls als interpretierbare CL-Namen gedeutet.
Sei $xxx \in Var_{RDF}$, dann gilt:

$$RDFtoCL\,(xxx) = xxx \tag{5.5}$$

Als nächstes betrachten wir die ungetypten RDF-Literale. Jedes ungetypte RDF-Literal ohne Sprachkennzeichnung wird nach Definition 5.12 auf sich selbst abgebildet und ist daher vergleichbar mit einem quoted String.

Sei "s" $\in$ *uLit* ein ungetyptes RDF-Literal ohne Sprachkennzeichnung, dann gilt:

$$RDFtoCL\,(\texttt{"s"}) = \texttt{'s'} \tag{5.6}$$

Demgegenüber ordnen wir jedem ungetypten RDF-Literal mit Sprachkennzeichnung einen CL-Funktionsterm zu. Dieser besteht aus der Funktion `CLungeLit`, welche als Argumente das ungetypte RDF-Literal und die Sprachkennzeichnung enthält, die jeweils als quoted String behandelt werden. Die Funktion `CLungeLit` bildet jedes Argumentenpaar auf ein Element des Gegenstandsbereichs UD ab, welches eindeutig durch die beiden Argumente festgelegt ist.
Sei "s"@t $\in$ *uLit* ein ungetyptes RDF-Literal mit Sprachkennzeichnung, dann gilt:

$$RDFtoCL\,(\texttt{"s"@t}) = (\texttt{CLungeLit 's' 't'}) \tag{5.7}$$

Ein getyptes RDF-Literal wird ebenfalls auf einen CL-Funktionsterm abgebildet. Der Grund dafür ist, dass Common Logic keine vordefinierten Datentypen beinhaltet und diese manuell simuliert werden müssen. Die Funktion des betreffenden CL-Funktionsterms entspricht dabei der Abbildung *L2W* aus Definition 5.15 auf Seite 97 und ist mit der entsprechenden Datentyp-URI bezeichnet.
Sei "*aaa*"^^ddd ein getyptes RDF-Literal, dann gilt:

$$RDFtoCL\,(\texttt{"}aaa\texttt{"\^\^ddd}) = (\texttt{ddd '}aaa\texttt{'}) \tag{5.8}$$

Nachdem wir die Abbildung *RDFtoCL* für die elementaren Syntaxkategorien von RDF/RDFS festgelegt haben, werden wir nun die Einbettung von RDF-Tripeln und RDF-Graphen definieren. Um die RDF-Tripel in Common Logic zu kennzeichnen, führen wir die Relation `RDF_TRIPEL` ein.
Sei $p\,(s,o)$ ein RDF-Tripel, dann gilt:

$$\begin{aligned} RDFtoCL\,(p\,(s,o)) = (\texttt{RDF\_TRIPEL} \\ & RDFtoCL\,(p) \\ & RDFtoCL\,(s) \\ & RDFtoCL\,(o) \\ &) \end{aligned} \tag{5.9}$$

Die Semantik eines RDF-Tripels $p\,(s,o)$ ist vergleichbar mit der Semantik einer CL-Atomformel, welche das Property p des RDF-Tripels als Prädikat und das Subjekt s sowie das Objekt o des RDF-Tripels als Argumentsequenz enthält. Daher führen wir für RDF-Tripel das Axiom Ax_1 ein.

$$\begin{aligned} Ax_1 = (\textsf{forall}\,(P\ S\ O)\ (\textsf{iff} \\ & (\texttt{RDF\_TRIPEL}\ P\ S\ O) \\ & (P\ S\ O) \\ &)) \end{aligned} \tag{5.10}$$

Als letzten Schritt legen wir fest wie RDF-Graphen nach Common Logic abgebildet werden. Nach Definition 5.13 entspricht ein leerer RDF-Knoten einem existenzquantifizierten CL-Namen. Allerdings ist die Reichweite des Existenzquantors nicht auf das RDF-Tripel beschränkt in dem der leere RDF-Knoten auftritt, sondern der Wirkungsbereich des Existenzquantors umfasst den gesamten RDF-Graphen. Des Weiteren muss eine einfache Interpretation, die einen RDF-Graphen erfüllt, jedes RDF-Tripel des entsprechenden RDF-Graphen erfüllen, so dass ein RDF-Graph semantisch mit einer Konjunktion über alle RDF-Tripel vergleichbar ist.
Sei $G = \{t_1, t_2, \ldots, t_n\}$ ein RDF-Graph bestehend aus den RDF-Tripeln $t_1, t_2, \ldots, t_n$. Weiter sei $Var_{RDF}(G) = \{x_1, x_2, \ldots, x_m\}$ die Menge der leeren RDF-Knoten von G. Dann gilt also:
$RDFtoCL(G) =$

$$
\begin{aligned}
&(\text{exists } (RDFtoCL(x_1)\ \ldots\ RDFtoCL(x_m))\ (\text{and} \\
&\qquad RDFtoCL(t_1) \\
&\qquad RDFtoCL(t_2) \\
&\qquad \ldots \\
&\qquad RDFtoCL(t_n) \\
&))
\end{aligned}
\tag{5.11}
$$

Die Definition der Abbildung $RDFtoCL : Gr_{RDF}(V_R, Var_{RDF}) \rightarrow Fm(V_{CL})$ wird in Tabelle 5.1 auf der nächsten Seite noch einmal übersichtlich dargestellt.

Im zweiten Teil dieses Abschnitts werden wir die Eigenschaften der Einbettung von einfachen Interpretationen genauer untersuchen. Dazu führen wir die nachfolgende Definition ein.

Definition 5.26 (Verträglichkeit einer CL-Interpretation)
Sei $I = \langle \mathcal{S}_I, int_I^U, int_I^L, int_I^V \rangle$ eine einfache Interpretation über V_R und Var_{RDF} mit der dazugehörigen semantischen RDF-Struktur $\mathcal{S}_I = \langle U_I, P_I, LV_I, rel_I \rangle$. Weiterhin sei $K = \langle \mathcal{S}_K, int_K, seq_K, irr_K \rangle$ eine CL-Interpretation über $V_{CL} = N \dot{\cup} SEQ$ und IFm mit der semantischen CL-Struktur $\mathcal{S}_K = \langle UR_K, UD_K, rel_K, fun_K \rangle$.
Die CL-Interpretation K heißt *verträglich* bezüglich der einfachen Interpretation I, falls K die folgenden Bedingungen erfüllt:

1. $V \subseteq N$ mit $V = RDFtoCL(URI_{RDF}) \cup Var_{RDF} \cup \{\texttt{CLungeLit}, \texttt{RDF\_TRIPEL}\}$
2. $\{RDFtoCL(\texttt{"s"}) \mid \text{für } \texttt{"s"} \in uLit\} \subseteq N$
3. $\{\texttt{'}a\texttt{'} \mid \text{mit } a = \texttt{s} \text{ oder } a = \texttt{t} \text{ für } \texttt{"s"@t} \in uLit\} \subseteq N$
4. $\{\texttt{ddd} \mid \text{für getyptes RDF-Literal } \texttt{"}aaa\texttt{"\^\^ddd} \in tLit\} \subseteq N$
5. $\{\texttt{'}aaa\texttt{'} \mid \text{für getyptes RDF-Literal } \texttt{"}aaa\texttt{"\^\^ddd} \in tLit\} \subseteq N$

RDF-Syntaxkategorie *Aus*	**Definition von** $RDFtoCL\,(Aus)$	**Nr.**
RDF URI-Referenz $<aaa>$	interpretierbarer CL-Name *aaa*	5.4
QName *bb*:*aaa*	interpretierbarer CL-Name *bb*:*aaa*	5.4
leerer RDF-Knoten *xxx*	interpretierbarer CL-Name *xxx*	5.5
ungetyptes RDF-Literal ohne Sprachkennzeichnung `"s"`	quoted String `'s'`	5.6
ungetyptes RDF-Literal mit Sprachkennzeichnung `"s"@t`	CL-Funktionsterm `(CLungeLit 's' 't')`	5.7
getyptes RDF-Literal `"aaa"^^ddd`	CL-Funktionsterm `(ddd 'aaa')`	5.8
RDF-Tripel $p\,(s,o)$	CL-Atomformel $\big($`RDF_TRIPEL` $RDFtoCL\,(p)$ $RDFtoCL\,(s)$ $RDFtoCL\,(o)\big)$	5.9
RDF-Graph $G = \{t_1, t_2, \ldots, t_n\}$ mit $Var_{RDF}\,(G) = \{x_1, \ldots, x_m\}$	CL-Formel $\big($exists$($ $RDFtoCL\,(x_1)$ $\ldots$ $RDFtoCL\,(x_m)\big)$ $($and $RDFtoCL\,(t_1)$ $\ldots$ $RDFtoCL\,(t_n)\big)\big)$	5.11

Tabelle 5.1: Definition der Abbildung $RDFtoCL$

6. $U_I \cup P_I \subseteq UD_K$
7. $int_K(RDFtoCL(u)) = int_I^U(u)$ für alle $u \in URI_{RDF}$
8. $int_K(RDFtoCL(x)) = int_I^V(x)$ für alle $x \in Var_{RDF}$
9. $fun_K(int_K(\texttt{CLungeLit})) = f_1$, wobei $f_1(\texttt{s},\texttt{t}) = int_I^*(\texttt{"s"@t})$ für alle ungetypten RDF-Literale mit Sprachkennzeichnung $\texttt{"s"@t} \in uLit$ gilt.
10. $fun_K(int_K(\texttt{ddd})) = f_{\texttt{ddd}}$, wobei $f_{\texttt{ddd}}(aaa) = int_I^L(\texttt{"}aaa\texttt{"\^\^ddd})$ für alle getypten RDF-Literale $\texttt{"}aaa\texttt{"\^\^ddd} \in tLit$ gilt.
11. $rel_I(p) \subseteq rel_K(p)$ für alle $p \in P_I$

□

Als Abschluss dieser Passage beweisen wir das folgende Resultat:

Lemma 5.27
Seien $G \in Gr_{RDF}(V_R, Var_{RDF})$ ein RDF-Graph und $I \in \mathcal{IB}_e(V_R, Var_{RDF})$ eine einfache Interpretation über V_R und Var_{RDF} mit $I \models G$. Weiterhin sei K eine CL-Interpretation über V_{CL} und IFm, die bezüglich I verträglich ist.
Falls $K \models Ax_1$, dann gilt $K \models RDFtoCL(G)$.

Beweis:
Sei $G = \{t_1, t_2, \ldots, t_n\}$ ein RDF-Graph, welcher aus den RDF-Tripeln $t_1, t_2, \ldots, t_n$ besteht. Die Menge der leeren RDF-Knoten von G sei $Var_{RDF}(G) = \{x_1, \ldots, x_m\}$. Weiterhin sei $I = \langle \mathcal{S}_I, int_I^U, int_I^L, int_I^V \rangle$ eine einfache Interpretation mit der semantischen RDF-Struktur $\mathcal{S}_I = \langle U_I, P_I, LV_I, rel_I \rangle$ und $I \models G$.
Da $I \models G$ existiert eine RDF-Modifikation $\widehat{I} \in Modi_{RDF}(I)$ von I mit $\widehat{I} \models t_i$ für alle $1 \leq i \leq n$. Sei $\widehat{K} = \langle \mathcal{S}_{\widehat{K}}, int_{\widehat{K}}, seq_{\widehat{K}}, irr_{\widehat{K}} \rangle$ eine CL-Interpretation mit $\mathcal{S}_{\widehat{K}} = \langle UR_{\widehat{K}}, UD_{\widehat{K}}, rel_{\widehat{K}}, fun_{\widehat{K}} \rangle$ und $\widehat{K} \models Ax_1$, die bezüglich $\widehat{I}$ verträglich ist.
Wir zeigen nun $\widehat{K} \models (\mathsf{and}\ RDFtoCL(t_1)\ \ldots\ RDFtoCL(t_n))$.
Sei $t_i \in G$ ein RDF-Tripel von G mit $t_i = p(s,o)$. Zu zeigen ist: $\widehat{K} \models RDFtoCL(t_i)$.
Wir unterscheiden dabei die nachfolgenden Fälle.
1. Fall: Seien $p, s, o \in URI_{RDF}$.
Da $\widehat{I} \models t_i$ folgt $int_{\widehat{I}}^U(p) \in P_{\widehat{I}}$ und $\left\langle int_{\widehat{I}}^U(s), int_{\widehat{I}}^U(o) \right\rangle \in rel_{\widehat{I}}\left(int_{\widehat{I}}^U(p)\right)$.
Damit $\langle int_{\widehat{K}}(RDFtoCL(s)), int_{\widehat{K}}(RDFtoCL(o)) \rangle \in rel_{\widehat{K}}\left(int_{\widehat{K}}(RDFtoCL(p))\right)$, nach Definition 5.26.
Somit gilt $\widehat{K} \models (RDFtoCL(p)\ RDFtoCL(s)\ RDFtoCL(o))$.
Also $\widehat{K} \models RDFtoCL(p(s,o))$, da $\widehat{K} \models Ax_1$.
2. Fall: Seien $p, s \in URI_{RDF}$ und $o \in Var_{RDF}$.
Da $\widehat{I} \models t_i$ folgt $int_{\widehat{I}}^U(p) \in P_{\widehat{I}}$ und $\left\langle int_{\widehat{I}}^U(s), int_{\widehat{I}}^V(o) \right\rangle \in rel_{\widehat{I}}\left(int_{\widehat{I}}^U(p)\right)$.
Damit $\langle int_{\widehat{K}}(RDFtoCL(s)), int_{\widehat{K}}(RDFtoCL(o)) \rangle \in rel_{\widehat{K}}\left(int_{\widehat{K}}(RDFtoCL(p))\right)$, nach Definition 5.26.

Somit gilt $\widehat{K} \models \left(RDFtoCL\left(p\right)\ RDFtoCL\left(s\right)\ RDFtoCL\left(o\right)\right)$.
Also $\widehat{K} \models RDFtoCL\left(p\left(s,o\right)\right)$, da $\widehat{K} \models Ax_1$.
3. Fall: Seien $p, o \in URI_{RDF}$ und $s \in Var_{RDF}$.
analog zu 2. Fall
4. Fall: Seien $p \in URI_{RDF}$ und $s, o \in Var_{RDF}$.
analog zu 2. Fall
5. Fall: Seien $p, s \in URI_{RDF}$ und $o \in uLit$, wobei $o =$ `"z"` ein ungetyptes RDF-Literal ohne Sprachkennzeichnung sei.
Da $\widehat{I} \models t_i$ folgt $int^U_{\widehat{I}}\left(p\right) \in P_{\widehat{I}}$ und $\left\langle int^U_{\widehat{I}}\left(s\right), \texttt{z} \right\rangle \in rel_{\widehat{I}}\left(int^U_{\widehat{I}}\left(p\right)\right)$.
Damit $\left\langle int_{\widehat{K}}\left(RDFtoCL\left(s\right)\right), int_{\widehat{K}}\left(RDFtoCL\left(o\right)\right)\right\rangle \in rel_{\widehat{K}}\left(int_{\widehat{K}}\left(RDFtoCL\left(p\right)\right)\right)$, nach Definition 5.26.
Somit gilt $\widehat{K} \models \left(RDFtoCL\left(p\right)\ RDFtoCL\left(s\right)\ RDFtoCL\left(o\right)\right)$.
Also $\widehat{K} \models RDFtoCL\left(p\left(s,o\right)\right)$, da $\widehat{K} \models Ax_1$.
6. Fall: Seien $p, s \in URI_{RDF}$ und $o \in uLit$, wobei $o =$ `"z"@t` ein ungetyptes RDF-Literal mit Sprachkennzeichnung sei.
Da $\widehat{I} \models t_i$ folgt $int^U_{\widehat{I}}\left(p\right) \in P_{\widehat{I}}$ und $\left\langle int^U_{\widehat{I}}\left(s\right), \langle \texttt{z}, \texttt{t} \rangle \right\rangle \in rel_{\widehat{I}}\left(int^U_{\widehat{I}}\left(p\right)\right)$. Damit $\left\langle int_{\widehat{K}}\left(RDFtoCL\left(s\right)\right), int^*_{\widehat{K}}\left(\left(\texttt{CLungeLit 'z' 't'}\right)\right)\right\rangle \in rel_{\widehat{K}}\left(int_{\widehat{K}}\left(RDFtoCL\left(p\right)\right)\right)$, nach Definition 5.26.
Somit gilt $\widehat{K} \models \left(RDFtoCL\left(p\right)\ RDFtoCL\left(s\right)\ RDFtoCL\left(o\right)\right)$.
Also $\widehat{K} \models RDFtoCL\left(p\left(s,o\right)\right)$, da $\widehat{K} \models Ax_1$.
7. Fall: Seien $p, s \in URI_{RDF}$ und $o \in tLit$, wobei $o =$ `"aaa"^^ddd` ein getyptes RDF-Literal sei.
Da $\widehat{I} \models t_i$ folgt $int^U_{\widehat{I}}\left(p\right) \in P_{\widehat{I}}$ und $\left\langle int^U_{\widehat{I}}\left(s\right), int^L_{\widehat{I}}\left(o\right) \right\rangle \in rel_{\widehat{I}}\left(int^U_{\widehat{I}}\left(p\right)\right)$.
Damit $\left\langle int_{\widehat{K}}\left(RDFtoCL\left(s\right)\right), int^*_{\widehat{K}}\left(\left(\texttt{ddd 'aaa'}\right)\right)\right\rangle \in rel_{\widehat{K}}\left(int_{\widehat{K}}\left(RDFtoCL\left(p\right)\right)\right)$, nach Definition 5.26.
Somit gilt $\widehat{K} \models \left(RDFtoCL\left(p\right)\ RDFtoCL\left(s\right)\ RDFtoCL\left(o\right)\right)$.
Also $\widehat{K} \models RDFtoCL\left(p\left(s,o\right)\right)$, da $\widehat{K} \models Ax_1$.
8. Fall: Seien $p \in URI_{RDF}$, $s \in Var_{RDF}$ und $o \in uLit$, wobei $o =$ `"z"` ein ungetyptes RDF-Literal ohne Sprachkennzeichnung sei.
analog zu 5. Fall
9. Fall: Seien $p \in URI_{RDF}$, $s \in Var_{RDF}$ und $o \in uLit$, wobei $o =$ `"z"@t` ein ungetyptes RDF-Literal mit Sprachkennzeichnung sei.
analog zu 6. Fall
10. Fall: Seien $p \in URI_{RDF}$, $s \in Var_{RDF}$ und $o \in tLit$, wobei $o =$ `"aaa"^^ddd` ein getyptes RDF-Literal sei.
analog zu 7. Fall
Da t_i ein beliebiges RDF-Tripel ist und für jedes RDF-Tripel genau einer der obigen Fälle zutrifft, folgt also $\widehat{K} \models \left(\mathsf{and}\ RDFtoCL\left(t_1\right)\ \ldots\ RDFtoCL\left(t_n\right)\right)$.
Als nächsten Schritt konstruieren wir eine CL-Interpretation $K \in \mathcal{IB}_{CL}(V_{CL}, IFm)$ über V_{CL} und IFm mit $K = \langle \mathcal{S}_K, int_K, seq_K, irr_K \rangle$ wie folgt:

(1) $\mathcal{S}_K := \mathcal{S}_{\widehat{K}}$,
(2) $int_K(n) := int_{\widehat{K}}(n)$ für alle $n \in N$ mit $n \notin Var_{RDF}$,
(3) $int_K(n) := int_I^V(n)$ für alle $n \in N$ mit $n \in Var_{RDF}$,
(4) $seq_K(s) := seq_{\widehat{K}}(s)$ für alle $s \in SEQ$ und
(5) $irr_K(\varphi) := irr_{\widehat{K}}(\varphi)$ für alle $\varphi \in IFm$.
Es ist leicht nachprüfbar, dass für K die beiden Eigenschaften
(1) $K \models Ax_1$ und
(2) K ist verträglich bezüglich der einfachen Interpretation I
gelten.
Aufgrund der Konstruktion von K gilt außerdem, dass die CL-Interpretation $\widehat{K}$ eine CL-Modifikation von K bezüglich der Menge Var_{RDF} ist.
Da $\widehat{K} \models (\mathsf{and}\ RDFtoCL(t_1)\ \ldots\ RDFtoCL(t_n))$ folgt
$K \models \Big(\mathsf{exists}\ (x_1\ \ldots\ x_m)\ (\mathsf{and}\ RDFtoCL(t_1)\ \ldots\ RDFtoCL(t_n))\Big)$.
Also $K \models RDFtoCL(G)$.

■

5.5.2 Einbettung der RDF-Interpretationen

Nachdem wir im vorangegangenen Abschnitt die einfachen Interpretationen in Common Logic eingebettet haben, wenden wir uns nun den RDF-Interpretationen zu. Angesichts der Tatsache, dass RDF-Interpretationen auf den selben Syntaxkategorien basieren wie die einfachen Interpretationen, ist es möglich die Abbildung $RDFtoCL : Gr_{RDF}(V_R, Var_{RDF}) \to Fm(V_{CL})$ unverändert für die Einbettung der RDF-Interpretationen anzuwenden. Die RDF-Interpretationen erfüllen jedoch im Vergleich zu einfachen Interpretationen zusätzliche Bedingungen, die wir mithilfe weiterer Axiome für Common Logic sicherstellen werden.

Die erste Anforderung von Definition 5.16 auf Seite 98 besagt, dass für alle Propertys p das RDF-Tripel $\texttt{rdf:type}\,(p, \texttt{rdf:Property})$ gültig ist. In Common Logic gewährleisten wir diesen Sachverhalt durch das Axiom Ax_2, welches wie folgt definiert ist:

$$\begin{aligned} Ax_2 = (\mathsf{forall}\ (x)\ (\mathsf{iff} \\ &(\texttt{rdf:type}\ x\ \texttt{rdf:Property}) \\ &(\texttt{rdf:Property}\ x) \\ ))\end{aligned} \tag{5.12}$$

Nach Definition 5.13 gilt für jede RDF-Interpretation I mit $I \models p(s,o)$ auch $int_I^*(p) \in P_I$. Diese Eigenschaft wird in Common Logic durch die CL-Formel Ax_3

garantiert.

$$\begin{aligned} Ax_3 = (\textsf{forall}\,(P\ S\ O)\,(\textsf{if} \\ &(\texttt{RDF\_TRIPEL}\ P\ S\ O) \\ &(\texttt{rdf:type}\ P\ \texttt{rdf:Property}) \\ &)) \end{aligned} \tag{5.13}$$

Aufgrund der neu eingeführten RDF URI-Referenzen des RDF-Vokabulars kann das Axiom Ax_1 aus 5.10 für RDF-Interpretationen folgendermaßen verfeinert werden:

$$\begin{aligned} Ax_1^{'} = (\textsf{forall}\,(P\ S\ O)\,(\textsf{iff} \\ &(\texttt{RDF\_TRIPEL}\ P\ S\ O) \\ &(\textsf{and} \\ &\quad (P\ S\ O) \\ &\quad (\texttt{rdf:type}\ P\ \texttt{rdf:Property}) \\ &) \\ &)) \end{aligned} \tag{5.14}$$

Die Erweiterung des Axioms Ax_1 zu $Ax_1^{'}$ beinhaltet bereits das Axiom Ax_3, so dass bei Verwendung von $Ax_1^{'}$ das Axiom Ax_3 überflüssig wird.

Des Weiteren muss die Gültigkeit der axiomatischen RDF-Tripel in Common Logic sichergestellt werden. Da diese RDF-Tripel lediglich RDF URI-Referenzen beinhalten, können diese problemlos mit der Abbildung *RDFtoCL* nach Common Logic übertragen werden.

Abschließend sei angemerkt, dass die Definition 5.16 auch Bedingungen für den vordefinierten Datentyp der RDF-Interpretationen formuliert. Diese zusätzlichen Anforderungen lassen sich in Common Logic mithilfe des RDF-Vokabulars nur unzureichend axiomatisieren. Die nachfolgenden CL-Formeln formulieren daher nur einen Teil dieser Forderungen.
Sei s ein wohlgeformtes XML-Literal und $'s' \in N$, dann gilt:

$$Ax_4^s = (\texttt{RDF\_TRIPEL rdf:type}\ (\texttt{rdf:XMLLiteral}'s')\ \texttt{rdf:XMLLiteral}) \tag{5.15}$$

Sei s ein nicht wohlgeformtes XML-Literal und $'s' \in N$, dann gilt:

$$Ax_5^s = (\textsf{not}\,(\texttt{rdf:type}\ (\texttt{rdf:XMLLiteral}'s')\ \texttt{rdf:XMLLiteral})) \tag{5.16}$$

5.5.3 Einbettung der RDFS-Interpretationen

Nun zeigen wir, dass auch die RDFS-Interpretationen in Common Logic eingebettet werden können. Die Syntaxkategorien der RDFS-Interpretationen werden mithilfe

der Abbildung *RDFtoCL* ebenfalls vollständig nach Common Logic übertragen. Ähnlich der RDF-Interpretationen ist es jedoch notwendig, zusätzliche Axiome in Common Logic zu fordern, um die Bedingungen der RDFS-Interpretationen aus Definition 5.19 auf Seite 100 sicherzustellen.

Eine der wichtigsten Veränderungen in RDFS besteht darin, dass die RDF URI-Referenz `rdf:type` nicht nur dazu verwendet wird um Bezeichner als Propertys zu kennzeichnen, sondern damit lassen sich nun beliebige Klassenzugehörigkeiten formulieren. Daher ist es erforderlich das Axiom Ax_2 aus 5.12 zu dem Axiom Ax_2' zu verallgemeinern. Es gilt daher:

$$\begin{aligned} Ax_2' = (\text{forall}\ (X\ Y)\ (\text{iff} \\ &(\texttt{rdf:type}\ X\ Y) \\ &(\text{and} \\ &\quad (Y\ X) \\ &\quad (\texttt{rdf:type}\ Y\ \texttt{rdfs:Class}) \\ &) \\ &)) \end{aligned} \tag{5.17}$$

In Lemma 5.22 auf Seite 102 haben wir gezeigt, dass jede Property und auch jede Klasse eines RDFS-Universums ein Element der Menge U ist. Daher umfasst die Klasse `rdfs:Resource` alle Elemente eines RDFS-Universums. Das Axiom Ax_6 formuliert die entsprechende Bedingung für Common Logic.

$$Ax_6 = (\text{forall}\ (x)\ (\texttt{rdf:type}\ x\ \texttt{rdfs:Resource})) \tag{5.18}$$

Als nächstes axiomatisieren wir die Semantik der beiden RDF URI-Referenzen `rdfs:domain` und `rdfs:range` mit deren Hilfe der Definitionsbereich sowie der Wertebereich einer Property eingeschränkt werden kann. Für `rdfs:domain` ist das folgende Axiom in Common Logic gültig:

$$\begin{aligned} Ax_7 = (\text{forall}\ (X\ Y\ u\ v)\ (\text{if} \\ &(\text{and} \\ &\quad (\texttt{RDF\_TRIPEL}\ \texttt{rdfs:domain}\ X\ Y) \\ &\quad (\texttt{RDF\_TRIPEL}\ X\ u\ v) \\ &) \\ &(\texttt{rdf:type}\ u\ Y) \\ &)) \end{aligned} \tag{5.19}$$

Das dazu korrespondierende Axiom für `rdfs:range` lautet:

$$
\begin{aligned}
Ax_8 = (&\textsf{forall}\,(X\ Y\ u\ v)\ (\textsf{if}\\
&\quad(\textsf{and}\\
&\qquad(\texttt{RDF\_TRIPEL}\ \texttt{rdfs:range}\ X\ Y)\\
&\qquad(\texttt{RDF\_TRIPEL}\ X\ u\ v)\\
&\quad)\\
&\quad(\texttt{rdf:type}\ v\ Y)\\
&))
\end{aligned}
\tag{5.20}
$$

Die neunte Bedingung von Definition 5.19 fordert die Transitivität und Reflexivität der RDF URI-Referenz `rdfs:subPropertyOf`. Die Transitivität stellen wir unter Verwendung von Axiom Ax_9 sicher.

$$
\begin{aligned}
Ax_9 = (&\textsf{forall}\,(x\ y\ z)\ (\textsf{if}\\
&\quad(\textsf{and}\\
&\qquad(\texttt{RDF\_TRIPEL}\ \texttt{rdfs:subPropertyOf}\ x\ y)\\
&\qquad(\texttt{RDF\_TRIPEL}\ \texttt{rdfs:subPropertyOf}\ y\ z)\\
&\quad)\\
&\quad(\texttt{RDF\_TRIPEL}\ \texttt{rdfs:subPropertyOf}\ x\ z)\\
&))
\end{aligned}
\tag{5.21}
$$

Die Reflexivität der Property `rdfs:subPropertyOf` wird durch die CL-Formel Ax_{10} garantiert.

$$
\begin{aligned}
Ax_{10} = (&\textsf{forall}\,(x)\ (\textsf{if}\\
&\quad(\texttt{rdf:type}\ x\ \texttt{rdf:Property})\\
&\quad(\texttt{RDF\_TRIPEL}\ \texttt{rdfs:subPropertyOf}\ x\ x)\\
&))
\end{aligned}
\tag{5.22}
$$

Die eigentliche Semantik von `rdfs:subPropertyOf` gewährleisten wir durch die nachfolgende CL-Formel.

$$
\begin{aligned}
Ax_{11} = (&\textsf{forall}\,(X\ Y)\ (\textsf{if}\\
&\quad(\texttt{RDF\_TRIPEL}\ \texttt{rdfs:subPropertyOf}\ X\ Y)\\
&\quad(\textsf{forall}\,(a\ b)\ (\textsf{if}\\
&\qquad(\texttt{RDF\_TRIPEL}\ X\ a\ b)\\
&\qquad(\texttt{RDF\_TRIPEL}\ Y\ a\ b)\\
&\quad))\\
&))
\end{aligned}
\tag{5.23}
$$

Nach Definition 5.19 fordert eine RDFS-Interpretation zusätzlich, dass aus dem RDF-Tripel `rdfs:subPropertyOf` (x, y) auch $int^*(x) \in P$ und $int^*(y) \in P$ abgeleitet werden kann. In Common Logic folgen die dazu korrespondierenden CL-Formeln (`rdf:type` x `rdf:Property`) und (`rdf:type` y `rdf:Property`) bereits aus den axiomatischen RDFS-Tripeln für `rdfs:subPropertyOf` sowie den beiden Axiomen Ax_7 und Ax_8.

Die nachfolgend angegebene CL-Formel gewährleistet die elfte Bedingung der RDFS-Interpretationen.

$$\begin{aligned} Ax_{12} = (&\text{forall } (x) \text{ (if} \\ &\quad (\texttt{rdf:type } x \texttt{ rdfs:Class}) \\ &\quad (\texttt{RDF\_TRIPEL rdfs:subClassOf } x \texttt{ rdfs:Resource}) \\ &\text{))} \end{aligned} \tag{5.24}$$

Der nächste Schritt besteht darin, die Semantik von `rdfs:subClassOf` in Common Logic zu charakterisieren.

$$\begin{aligned} Ax_{13} = (&\text{forall } (X\ Y) \text{ (if} \\ &\quad (\texttt{RDF\_TRIPEL rdfs:subClassOf } X\ Y) \\ &\quad (\text{forall } (a) \text{ (if} \\ &\qquad (\texttt{rdf:type } a\ X) \\ &\qquad (\texttt{rdf:type } a\ Y) \\ &\quad \text{))} \\ &\text{))} \end{aligned} \tag{5.25}$$

Auch in diesem Fall ist es nicht notwendig die Axiome (`rdf:type` x `rdfs:Class`) und (`rdf:type` y `rdfs:Class`) explizit in Common Logic zu fordern, denn diese folgen ebenfalls aus den eingebetteten axiomatischen RDFS-Tripeln und den beiden Axiomen Ax_7 und Ax_8.

Weiterhin wird für RDFS-Interpretationen gefordert, dass die RDF URI-Referenz `rdfs:subClassOf` sowohl transitiv als auch reflexiv auf der Menge aller Klassen ist. Das Axiom Ax_{14} garantiert die Transitivität.

$$\begin{aligned} Ax_{14} = (&\text{forall } (x\ y\ z) \text{ (if} \\ &\quad (\text{and} \\ &\qquad (\texttt{RDF\_TRIPEL rdfs:subClassOf } x\ y) \\ &\qquad (\texttt{RDF\_TRIPEL rdfs:subClassOf } y\ z) \\ &\quad) \\ &\quad (\texttt{RDF\_TRIPEL rdfs:subClassOf } x\ z) \\ &\text{))} \end{aligned} \tag{5.26}$$

Die Reflexivität von `rdfs:subClassOf` wird durch die folgende CL-Formel gewährleistet:

$$\begin{aligned} Ax_{15} = (\textsf{forall}\,(x)\,(\textsf{if} \\ &(\texttt{rdf:type}\ x\ \texttt{rdfs:Class}) \\ &(\texttt{RDF\_TRIPEL}\ \texttt{rdfs:subClassOf}\ x\ x) \\ &)) \end{aligned} \tag{5.27}$$

Die vierzehnte Bedingung von Definition 5.19 wird unter Verwendung von Axiom Ax_{16} in Common Logic sichergestellt.

$$\begin{aligned} Ax_{16} = (\textsf{forall}\,(x)\,(\textsf{if} \\ &(\texttt{rdf:type}\ x\ \texttt{rdfs:ContainerMembershipProperty}) \\ &(\texttt{RDF\_TRIPEL}\ \texttt{rdfs:subPropertyOf}\ x\ \texttt{rdfs:member}) \\ &)) \end{aligned} \tag{5.28}$$

Die darauffolgende Forderung für RDFS-Interpretationen wird in Common Logic mittels der CL-Formel Ax_{17} axiomatisiert. Es gilt:

$$\begin{aligned} Ax_{17} = (\textsf{forall}\,(x)\,(\textsf{if} \\ &(\texttt{rdf:type}\ x\ \texttt{rdfs:Datatype}) \\ &(\texttt{RDF\_TRIPEL}\ \texttt{rdfs:subClassOf}\ x\ \texttt{rdfs:Literal}) \\ &)) \end{aligned} \tag{5.29}$$

Schließlich müssen wir auch die Gültigkeit der axiomatischen RDFS-Tripel in Common Logic garantieren. Dies erfolgt, analog der axiomatischen RDF-Tripel, durch die Einbettung mithilfe der Abbildung *RDFtoCL*.

Abschließend wenden wir uns dem vordefinierten Datentyp `rdf:XMLLiteral` zu. Aufgrund der neu eingeführten RDF URI-Referenz `rdfs:Literal` lassen sich die zweite und dritte Bedingung aus Definition 5.16 in Common Logic weiter verfeinern. Mithilfe der nachfolgenden CL-Formel,

$$(\texttt{RDF\_TRIPEL}\ \texttt{rdfs:subClassOf}\ \texttt{rdf:XMLLiteral}\ \texttt{rdfs:Literal}) \tag{5.30}$$

welche durch Einbettung der axiomatischen RDFS-Tripel in Common Logic gültig ist, folgt aufgrund von Ax_4^s auch die CL-Formel

$$(\texttt{RDF\_TRIPEL}\ \texttt{rdf:type}\ (\texttt{rdf:XMLLiteral'}s\texttt{'})\ \texttt{rdfs:Literal}) \tag{5.31}$$

für jedes wohlgeformte XML-Literal s mit $\texttt{'}s\texttt{'} \in N$. Für nicht wohlgeformte XML-Literale ist es jedoch notwendig das Axiom Ax_5^s zu erweitern. Sei s ein nicht wohlgeformtes XML-Literal und $\texttt{'}s\texttt{'} \in N$, dann fordern wir:

$$\widehat{Ax}_5^s = (\textsf{not}\,(\texttt{rdf:type}\ (\texttt{rdf:XMLLiteral'}s\texttt{'})\,\texttt{rdfs:Literal})) \tag{5.32}$$

Aufgrund der CL-Formel aus 5.30 stellt das Axiom $\widehat{Ax}_5^s$ auch die Gültigkeit von Ax_5^s sicher.

Zusammenfassend halten wir fest, dass sowohl einfache Interpretationen als auch RDF-Interpretationen und RDFS-Interpretationen problemlos nach Common Logic übertragen werden können. Für die in Definition 5.24 auf Seite 107 eingeführten Reif-Interpretationen ist dies jedoch nicht möglich, da Common Logic keine Aussagen über CL-Formeln erlaubt. Um in der Lage zu sein, auch die Reifikation modellieren zu können, ist es notwendig eine Erweiterung von Common Logic wie beispielsweise IKL [40] zu verwenden.

Prinzipiell ist es möglich sowohl RDF als auch RDFS in die Prädikatenlogik der ersten Stufe zu übertragen. Allerdings sind Axiome wie Ax_1 aus 5.10 in der Prädikatenlogik nicht möglich, da zum Beispiel aus dem eingebetteten axiomatischen RDF-Tripel

$$(\texttt{RDF\_TRIPEL}\ \texttt{rdf:type}\ \texttt{rdf:type}\ \texttt{rdf:Property}) \tag{5.33}$$

nicht der Ausdruck

$$\texttt{rdf:type}\,(\texttt{rdf:type}, \texttt{rdf:Property}) \tag{5.34}$$

ableitbar ist. Dadurch bleiben die syntaktischen Freiheiten der RDF/RDFS-Syntax bei der Übertragung in die traditionelle Prädikatenlogik der ersten Stufe nicht erhalten. Für weitere Informationen zu dieser Thematik verweisen wir auf die Arbeiten von Christopher Menzel und Patrick Hayes [53] sowie Richard Fikes und Deborah McGuinness [26].

6 Extended RDF

Mithilfe von RDF können nur sehr einfache Sachverhalte modelliert werden. Betrachten wir beispielsweise die Ontologie aus Beispiel 2.16 auf Seite 20. So sind wir mit RDF in der Lage, einzelne Bezeichner unter Verwendung der RDF URI-Referenzen `rdf:Property` und `rdf:type` als Propertys zu deklarieren. Das RDF-Tripel `rdf:type` (`ex:hatVater`, `rdf:Property`) formuliert die entsprechende Aussage für die binäre Relation `hatVater`, wobei `ex` den Namensraumpräfix der URI-Referenz `<http://www.example.org/>` bezeichne. Des Weiteren lassen sich mittels RDF einzelne Individuen zueinander in Beziehung setzen. Dies ist jedoch nicht ausreichend um Ontologien formal beschreiben zu können. Auch unsere Beispielontologie wird in RDF nur unzureichend spezifiziert, da selbst elementare Bestandteile von Ontologien wie die Definition eigener Klassen oder die Spezifikation von Beziehungen zwischen einzelnen Klassen mithilfe von RDF nicht realisiert werden können.

Unter Verwendung von RDFS sind derartige Einschränkungen aufgehoben. Mittels `rdfs:Class` sind wir beispielsweise in der Lage eigene Klassen zu definieren. Die Modellierung von hierarchischen Beziehungen zwischen Klassen und Propertys ist in RDFS ebenfalls möglich. Aber die Ontologie aus Beispiel 2.16 lässt sich auch in RDFS nicht vollständig beschreiben. Einerseits ist es aufgrund der fehlenden Negation nicht möglich die Disjunktion von Klassen zu formulieren. Dadurch kann die Aussage, dass jeder `Mensch` entweder ein `Mann` oder eine `Frau` ist, nicht in RDFS gefordert werden. Andererseits können Eigenschaften von Propertys nur sehr eingeschränkt modelliert werden. So ist es beispielsweise nicht möglich die Reflexivität einer Property einzufordern oder Kardinalitätsrestriktionen auszudrücken. Daher ist sowohl die Klasse `kinderreicheMutter` als auch das Axiom, dass die beiden Relationen `hatSohn` und `hatVater` invers zueinander sind, nicht formulierbar.

Für die Beschreibung von komplexen Ontologien wurde deshalb die *Web Ontology Language (OWL)* entwickelt, welche die Ausdruckskraft von RDFS um ein Vielfaches übersteigt. Die erweiterten Modellierungsmöglichkeiten von OWL werden hauptsächlich durch zusätzliche Konstruktoren für Klassen und Propertys realisiert. So können mittels der RDF URI-Referenz `owl:disjointWith`[15] die beiden Klassen `Mann` und `Frau` als disjunkt gekennzeichnet werden. Die Aussage, dass die

[15] Der Namensraumpräfix `owl` wird für die URI-Referenz `<http://www.w3.org/2002/07/owl#>` verwendet.

Propertys `hatSohn` und `hatVater` invers zueinander sind, ist in OWL ebenfalls artikulierbar. Es existieren jedoch Anwendungsfälle für die die Modellierungsmöglichkeiten von OWL nicht ausreichend sind [28]. So sind beispielsweise Kompositionen[16] von Propertys in OWL nicht realisierbar. Falls wir unsere Beispielontologie um die Familienbeziehung Onkel erweitern und diese definieren als: die Person p ist ein Onkel der Person s, falls p und der Vater von s Brüder sind. Dann ist dieser Sachverhalt nicht in OWL darstellbar.

Aufgrund der Tatsache, dass auch Inferenzregeln für die Umsetzung des Semantic Web in naher Zukunft eingesetzt werden sollen, wurden zahlreiche Erweiterungen von RDFS und OWL entwickelt, die unter Verwendung von Regelmechanismen die bisherigen Modellierungsmöglichkeiten erweitern. Eine Vielzahl dieser Erweiterungen ist in [7] übersichtlich zusammengefasst. Die Anwendung von Regelformalismen ist keinesfalls neu, sondern geht auf die sogenannten *logischen Programme* [8] zurück, welche zur Wissensrepräsentation eingesetzt werden.

Wir stellen in diesem Kapitel eine Arbeit von Anastasia Analyti und anderen [5] vor, die RDFS durch Hinzunahme von zwei verschiedenen Negationsarten sowie einen Regelmechanismus zu *Extended RDF (ERDF)* erweitert, um dadurch die Ausdrucksstärke von RDFS zu erhöhen. Zunächst führen wir die Syntax von ERDF ein, bevor wir im zweiten Abschnitt die formale Semantik definieren. Anschließend untersuchen wir den Aufbau des ERDF-Universums und analysieren die partiellen Propertys und Klassen. Der fünfte Teil thematisiert die Einbettung von ERDF in Common Logic. Am Ende des Kapitels knüpfen wir an die in Abschnitt 6.2 eingeführte Semantik an und verfeinern diese.

6.1 Syntax von ERDF

In [5] wird ERDF unter Verwendung von abstrakten Syntaxkategorien spezifiziert, die auf den elementaren Syntaxkategorien von RDF/RDFS aufbauen. Es existiert zusätzlich eine konkrete Syntax für ERDF, welche an die RDF/XML-Syntax angelehnt ist [73]. Wir führen ERDF auf Basis der abstrakten Syntax ein.

Definition 6.1 (positives ERDF-Tripel)
Sei $V_R = URI_{RDF} \dot{\cup} Lit$ ein R-Vokabular und Var_{RDF} eine Menge von leeren RDF-Knoten. Ein *positives ERDF-Tripel* $p(s, o)$ über V_R und Var_{RDF} besteht aus den folgenden drei Komponenten:

1. dem *Prädikat* $p \in URI_{RDF}$, welches eine RDF URI-Referenz ist,

[16] Unter einer Komposition im mathematischen Sinne verstehen wir die Hintereinanderausführung von binären Relationen. Seien R und S zwei binäre Relationen, dann ist die Komposition von R und S, bezeichnet mit $R \circ S$, definiert als: $(x, z) \in R \circ S$ genau dann, wenn ein y existiert mit $(x, y) \in R$ und $(y, z) \in S$.

2. dem *Subjekt* $s \in URI_{RDF} \cup Lit \cup Var_{RDF}$, welches entweder eine RDF URI-Referenz, ein RDF-Literal oder ein leerer RDF-Knoten ist und

3. dem *Objekt* $o \in URI_{RDF} \cup Lit \cup Var_{RDF}$, welches entweder eine RDF URI-Referenz, ein RDF-Literal oder ein leerer RDF-Knoten ist.

Die Menge der positiven ERDF-Tripel über V_R und Var_{RDF} bezeichnen wir mit $Tri^{+}_{ERDF}(V_R, Var_{RDF})$.

□

Nach dieser Definition ist also jedes RDF-Tripel ein positives ERDF-Tripel. Die Umkehrung gilt jedoch nicht, da RDF-Tripel keine RDF-Literale als Subjekt enthalten.

Die positiven ERDF-Tripel werden nun zu allgemeinen ERDF-Tripeln erweitert, um auch negative Informationen darstellen zu können.

Definition 6.2 (negatives ERDF-Tripel, ERDF-Tripel)
Sei V_R ein R-Vokabular und Var_{RDF} eine Menge von leeren RDF-Knoten.

(a) Ein *negatives ERDF-Tripel* über V_R und Var_{RDF} ist eine Zeichenfolge der Form $\neg p(s, o)$, wobei $p(s, o)$ ein positives ERDF-Tripel über V_R und Var_{RDF} ist.

(b) Ein *ERDF-Tripel* über V_R und Var_{RDF} ist entweder ein positives ERDF-Tripel über V_R und Var_{RDF} oder ein negatives ERDF-Tripel über V_R und Var_{RDF}.

Für die Menge aller ERDF-Tripel über V_R und Var_{RDF} führen wir die Bezeichnung $Tri_{ERDF}(V_R, Var_{RDF})$ ein.

□

Übereinstimmend mit RDF/RDFS werden Mengen von ERDF-Tripeln zu sogenannten ERDF-Graphen vereinigt.

Definition 6.3 (ERDF-Graph)
Sei V_R ein R-Vokabular und Var_{RDF} eine Menge von leeren RDF-Knoten.
Ein *ERDF-Graph* $G \subseteq Tri_{ERDF}(V_R, Var_{RDF})$ ist eine Menge von ERDF-Tripeln über V_R und Var_{RDF}.

□

Für die Menge aller ERDF-Graphen über V_R und Var_{RDF} vereinbaren wir die Bezeichnung $Gr_{ERDF}(V_R, Var_{RDF})$. Analog zu den RDF-Graphen kennzeichnen wir die Menge der RDF URI-Referenzen und RDF-Literale eines ERDF-Graphen G mit $V_R(G)$ sowie die Menge der leeren RDF-Knoten von G mit $Var_{RDF}(G)$. Aufgrund obiger Definition ist jeder RDF-Graph auch ein ERDF-Graph. Die Umkehrung dieser Aussage ist wiederum nicht gültig.

Zusätzlich zu ERDF-Tripeln und ERDF-Graphen werden weitere Syntaxkategorien eingeführt.

Definition 6.4 (ERDF-Formel)
Sei V_R ein R-Vokabular und Var_{RDF} eine Menge von leeren RDF-Knoten.
Die Menge $Fm_{ERDF}(V_R, Var_{RDF})$ der *ERDF-Formeln* über V_R und Var_{RDF} ist die kleinste Menge von Zeichenreihen, welche die folgenden Bedingungen erfüllt:

1. $Tri^{+}_{ERDF}(V_R, Var_{RDF}) \subseteq Fm_{ERDF}(V_R, Var_{RDF})$
2. Falls $\varphi_1, \varphi_2 \in Fm_{ERDF}(V_R, Var_{RDF})$, dann gehören auch die Zeichenreihen $\sim \varphi_1$, $\neg\varphi_1$, $(\varphi_1 \wedge \varphi_2)$, $(\varphi_1 \vee \varphi_2)$ und $(\varphi_1 \rightarrow \varphi_2)$ zu $Fm_{ERDF}(V_R, Var_{RDF})$.
3. Falls $\varphi \in Fm_{ERDF}(V_R, Var_{RDF})$ und $x \in Var_{RDF}$, dann gehören auch die Zeichenreihen $\exists x\varphi$ und $\forall x\varphi$ zu $Fm_{ERDF}(V_R, Var_{RDF})$.

□

Wir unterscheiden zwischen positiven und negativen ERDF-Formeln. Sei $\varphi = \neg\psi$ eine ERDF-Formel mit $\psi \in Fm_{ERDF}(V_R, Var_{RDF})$, dann heißt φ *negative ERDF-Formel*. Falls $\varphi \in Fm_{ERDF}(V_R, Var_{RDF})$ keine negative ERDF-Formel ist, dann nennen wir φ eine *positive ERDF-Formel*.

Sei $\varphi \in Fm_{ERDF}(V_R, Var_{RDF})$ eine ERDF-Formel über V_R und Var_{RDF}. Die Menge der leeren RDF-Knoten, welche in φ vorkommen, bezeichnen wir mit $Var_{RDF}(\varphi)$. Diese leeren RDF-Knoten werden weiter charakterisiert. Sei $x \in Var_{RDF}(\varphi)$ ein leerer RDF-Knoten von φ, der durch einen Quantor[17] gebunden ist. Dann wird dieser RDF-Knoten *gebundener RDF-Knoten* genannt. Falls x nicht zu dem Wirkungsbereich eines Quantors gehört, dann heißt x *freier RDF-Knoten*. Die Menge der gebundenen RDF-Knoten von φ bezeichnen wir mit $Var^{g}_{RDF}(\varphi)$. Analog dazu wird die Menge der freien RDF-Knoten von φ mit $Var^{f}_{RDF}(\varphi)$ gekennzeichnet.
Als nächstes führen wir die Syntax der ERDF-Regeln ein.

Definition 6.5 (ERDF-Regel)
Sei V_R ein R-Vokabular und Var_{RDF} eine Menge von leeren RDF-Knoten.
Eine *ERDF-Regel* über V_R und Var_{RDF} ist ein Ausdruck der Form $\psi_1 \Leftarrow \psi_2$, welcher die folgenden Bedingungen erfüllt:

1. Für den *Regelrumpf* ψ_2 gilt: $\psi_2 \in Fm_{ERDF}(V_R, Var_{RDF}) \cup \{true\}$.
2. Für den *Regelkopf* ψ_1 gilt: $\psi_1 \in Tri_{ERDF}(V_R, Var_{RDF}) \cup \{false\}$.
3. Kein leerer RDF-Knoten, der in ψ_2 gebunden ist, darf in ψ_1 frei auftreten.[18]

□

Für die Menge aller ERDF-Regeln über V_R und Var_{RDF} vereinbaren wir die Bezeichnung $Re_{ERDF}(V_R, Var_{RDF})$. Des Weiteren führen wir für die Menge der leeren

[17] Die Quantoren in ERDF werden durch die Zeichen $\exists$ und $\forall$ symbolisiert.
[18] Diese Bedingung ist keine echte Einschränkung, denn sie kann mittels Umbenennung der leeren RDF-Knoten von ψ_1 sichergestellt werden.

RDF-Knoten einer ERDF-Regel r die Schreibweise $Var_{RDF}(r)$ und für die Menge der freien RDF-Knoten von r die Bezeichnung $Var^{f}_{RDF}(r)$ ein. Den Regelrumpf einer ERDF-Regel r kennzeichnen wir mit $Rumpf(r)$. Analog dazu bezeichnen wir den Regelkopf von r mit $Kopf(r)$.

Weiterhin werden ERDF-Regeln zu ERDF-Programmen sowie ERDF-Graphen und ERDF-Programme zu ERDF-Ontologien zusammengefasst.

Definition 6.6 (ERDF-Programm, ERDF-Ontologie)
Sei V_R ein R-Vokabular und Var_{RDF} eine Menge von leeren RDF-Knoten.

(a) Ein *ERDF-Programm* $P \subseteq Re_{ERDF}(V_R, Var_{RDF})$ über V_R und Var_{RDF} ist eine Menge von ERDF-Regeln über V_R und Var_{RDF}.

(b) Eine *ERDF-Ontologie* $O = \langle G, P \rangle$ über V_R und Var_{RDF} ist ein Paar bestehend aus einem ERDF-Graphen G über V_R und Var_{RDF} und einem ERDF-Programm P über V_R und Var_{RDF}.

□

Für die Menge der ERDF-Programme und die Menge der ERDF-Ontologien führen wir ebenfalls abkürzende Schreibweisen ein. So bezeichnen wir die Menge aller ERDF-Programme über V_R und Var_{RDF} mit $Pr_{ERDF}(V_R, Var_{RDF})$. Die Menge aller ERDF-Ontologien über V_R und Var_{RDF} nennen wir $On_{ERDF}(V_R, Var_{RDF})$.

6.2 Formale Semantik

In Übereinstimmung mit RDF/RDFS wird die formale Semantik von ERDF ebenfalls schrittweise definiert. Wir werden in diesem Abschnitt die partiellen Interpretationen, die darauf aufbauenden ERDF-Interpretationen sowie die kohärenten ERDF-Interpretationen vorstellen. Da jede kohärente ERDF-Interpretation eine ERDF-Interpretation und jede ERDF-Interpretation eine partielle Interpretation ist, lässt sich die Beziehung zwischen diesen Interpretationsmengen, wie in Abbildung 6.1 auf der nächsten Seite dargestellt, charakterisieren.

6.2.1 Partielle Interpretationen

Die Grundlage aller Interpretationen der abstrakten Syntaxkategorien von ERDF bilden die partiellen Interpretationen. Diese Interpretationen deuten die syntaktischen ERDF-Ausdrücke ebenfalls über mengentheoretischen Strukturen, welche wir als semantische P-Strukturen bezeichnen.

Definition 6.7 (semantische P-Struktur)
Sei $V_R = URI_{RDF} \dot{\cup} Lit$ ein R-Vokabular.
Eine *semantische P-Struktur* $\mathcal{S} = \langle Res, Prop, LV, rel^{+}, rel^{-} \rangle$ über V_R besteht aus:

- Einer nichtleeren Menge Res, die *Universum* von $\mathcal{S}$ genannt wird.

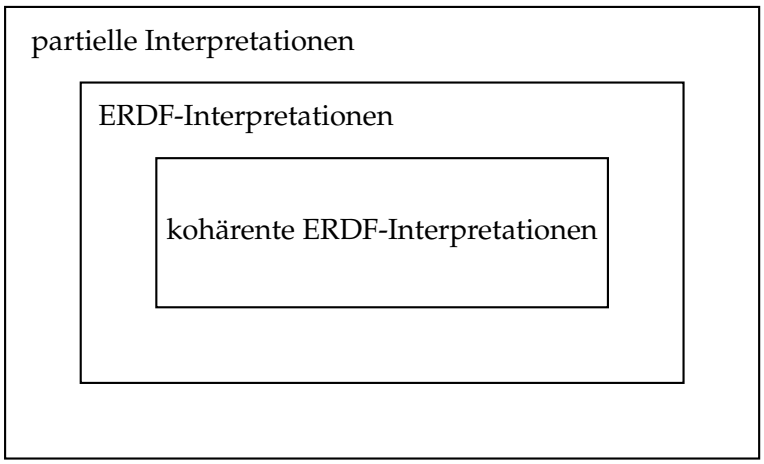

Abbildung 6.1: Die Interpretationen von ERDF

- Einer Menge $Prop$, die als *Menge der Propertys* von $\mathcal{S}$ bezeichnet wird.
- Einer Menge $LV \subseteq Res$, die *Menge der Literalwerte* heißt und alle ungetypten RDF-Literale aus V_R enthält, also $uLit \subseteq LV$.
- Einer Abbildung $rel^+ : Prop \to \mathcal{P}(Res \times Res)$.
- Einer Abbildung $rel^- : Prop \to \mathcal{P}(Res \times Res)$.

□

Eine semantische P-Struktur ist also einer semantischen RDF-Struktur sehr ähnlich. Der einzige Unterschied liegt in der Behandlung der Propertys. Während jede semantische RDF-Struktur einem Property genau eine Extension zuordnet, weist jede semantische P-Struktur einem Property zwei Extensionen zu.
Für semantische P-Strukturen führen wir nun die Identitätsrelation wie folgt ein:

Definition 6.8 (Identität zweier semantischer P-Strukturen)
Es sei V_R ein R-Vokabular. Des Weiteren seien $\mathcal{S}_1 = \langle Res_1, Prop_1, LV_1, rel_1^+, rel_1^- \rangle$ und $\mathcal{S}_2 = \langle Res_2, Prop_2, LV_2, rel_2^+, rel_2^- \rangle$ zwei semantische P-Strukturen über V_R.
Die beiden semantischen P-Strukturen $\mathcal{S}_1$ und $\mathcal{S}_2$ sind *identisch* genau dann, wenn die nachfolgenden Bedingungen erfüllt sind.

1. $Res_1 = Res_2$,
2. $Prop_1 = Prop_2$,
3. $LV_1 = LV_2$,
4. $rel_1^+(p) = rel_2^+(p)$ für alle $p \in Prop_1$ und

5. $rel_1^-(p) = rel_2^-(p)$ für alle $p \in Prop_1$.

□

Falls die beiden semantischen P-Strukturen $\mathcal{S}_1$ und $\mathcal{S}_2$ identisch sind, dann verwenden wir dafür die Schreibweise $\mathcal{S}_1 = \mathcal{S}_2$.

Als nächstes werden die semantischen P-Strukturen zu partiellen Interpretationen erweitert.

Definition 6.9 (partielle Interpretation)
Sei V_R ein R-Vokabular und Var_{RDF} eine Menge von leeren RDF-Knoten.
Eine *partielle Interpretation* $I = \langle \mathcal{S}, intU^P, intL^P, intV^P \rangle$ besteht aus:

- einer semantischen P-Struktur $\mathcal{S} = \langle Res, Prop, LV, rel^+, rel^- \rangle$,
- einer Abbildung $intU^P : URI_{RDF} \to Res \cup Prop$,
- einer Abbildung $intL^P : tLit \to Res$ und
- einer Abbildung $intV^P : Var_{RDF} \to Res$.

□

Die Menge der partiellen Interpretationen über dem R-Vokabular V_R und der Menge der leeren RDF-Knoten Var_{RDF} heißt *partieller Interpretationsbereich* über V_R und Var_{RDF} und wird mit $\mathcal{IB}_p(V_R, Var_{RDF})$ bezeichnet.

Die partiellen Interpretationen und die einfachen Interpretationen unterscheiden sich also lediglich in der Anzahl der Extensionen, die den Propertys zugeordnet werden. Auch bei ERDF sind weitere Definitionen notwendig, um in der Lage zu sein, die Erfüllbarkeitsrelation formal einführen zu können. Daher legen wir nun die folgenden Begriffe fest:

Definition 6.10 (ERDF-Modifikation)
Seien $I, K \in \mathcal{IB}_p(V_R, Var_{RDF})$ zwei partielle Interpretationen über V_R und Var_{RDF} mit $I = \langle \mathcal{S}_I, intU_I^P, intL_I^P, intV_I^P \rangle$ und $K = \langle \mathcal{S}_K, intU_K^P, intL_K^P, intV_K^P \rangle$. Weiterhin sei $VAR \subseteq Var_{RDF}$ eine Menge von leeren RDF-Knoten.
Die Interpretation K ist eine *ERDF-Modifikation* der Interpretation I bezüglich der Menge VAR, falls die folgenden Bedingungen erfüllt sind:

1. $\mathcal{S}_K = \mathcal{S}_I$, d.h. $\mathcal{S}_K$ und $\mathcal{S}_I$ sind identisch,
2. $intU_K^P(u) = intU_I^P(u)$ für alle $u \in URI_{RDF}$,
3. $intL_K^P(l) = intL_I^P(l)$ für alle $l \in tLit$ und
4. $intV_K^P(x) = intV_I^P(x)$ für alle $x \in Var_{RDF} \setminus VAR$.

□

Falls K eine ERDF-Modifikation von I bezüglich der Menge VAR ist, dann unterscheiden sich die beiden partiellen Interpretationen lediglich darin, wie die leeren RDF-Knoten aus VAR interpretiert werden. Die Menge aller ERDF-Modifikationen der partiellen Interpretation I bezüglich der Menge VAR nennen wir $Modi_{ERDF}^{VAR}(I)$.

Für die Interpretation der elementaren Syntaxkategorien führen wir nach dem Vorbild von RDF/RDFS die ERDF-Interpretationsabbildung ein.

Definition 6.11 (ERDF-Interpretationsabbildung)
Sei $V_R = URI_{RDF} \dot{\cup} Lit$ ein R-Vokabular und Var_{RDF} eine Menge von leeren RDF-Knoten. Weiterhin sei $I \in \mathcal{IB}_p(V_R, Var_{RDF})$ eine partielle Interpretation über V_R und Var_{RDF} mit $I = \langle \mathcal{S}_I, intU_I^P, intL_I^P, intV_I^P \rangle$ und der dazugehörigen semantischen P-Struktur $\mathcal{S}_I = \langle Res_I, Prop_I, LV_I, rel_I^+, rel_I^- \rangle$.
Die *ERDF-Interpretationsabbildung* $int_I^* : V_R \cup Var_{RDF} \to Res \cup Prop$ ist eine Abbildung, die wie folgt definiert ist:

1. Sei `"s"` $\in uLit$ ein ungetyptes RDF-Literal, dann gilt $int_I^*(\texttt{"s"}) = \texttt{s}$.
2. Sei `"s"@t` $\in uLit$ ein ungetyptes RDF-Literal mit Sprachkennzeichnung, dann gilt $int_I^*(\texttt{"s"@t}) = \langle \texttt{s}, \texttt{t} \rangle$.
3. Sei `l` $\in tLit$ ein getyptes RDF-Literal, dann gilt $int_I^*(\texttt{l}) = intL_I^P(\texttt{l})$.
4. Sei $u \in URI_{RDF}$ eine RDF URI-Referenz, dann gilt $int_I^*(u) = intU_I^P(u)$.
5. Sei $x \in Var_{RDF}$ ein leerer RDF-Knoten, dann gilt $int_I^*(x) = intV_I^P(x)$.

□

Eine ERDF-Interpretationsabbildung ordnet also jedem Element von Var_{RDF} und V_R ein Element des Universums oder eine Property zu.

Der nächste Schritt besteht darin, die Interpretation der komplexen Syntaxkategorien von ERDF festzulegen. Dazu führen wir sukzessive eine Erfüllbarkeitsrelation ein. Wir beginnen mit der Interpretation der positiven ERDF-Formeln.

Definition 6.12 (Erfüllbarkeit der positiven ERDF-Formeln)
Sei V_R ein R-Vokabular und Var_{RDF} eine Menge von leeren RDF-Knoten. Des Weiteren sei $I \in \mathcal{IB}_p(V_R, Var_{RDF})$ eine partielle Interpretation über V_R und Var_{RDF} mit $I = \langle \mathcal{S}_I, intU_I^P, intL_I^P, intV_I^P \rangle$ und $\mathcal{S}_I = \langle Res_I, Prop_I, LV_I, rel_I^+, rel_I^- \rangle$ die dazugehörige semantische P-Struktur. Weiter bezeichne $int_I^* : V_R \cup Var_{RDF} \to Res_I \cup Prop_I$ die durch I eindeutig festgelegte ERDF-Interpretationsabbildung.
Die *ERDF-Erfüllbarkeitsrelation* $\models$ ist für positive ERDF-Formeln wie folgt definiert:

1. Sei $\varphi = p(s, o)$ ein positives ERDF-Tripel. Dann gilt $I \models \varphi$ genau dann, wenn $p \in URI_{RDF}$, $s, o \in URI_{RDF} \cup Var_{RDF} \cup Lit$, $int_I^*(p) \in Prop_I$ und $\langle int_I^*(s), int_I^*(o) \rangle \in rel_I^+(int_I^*(p))$.
2. Sei $\varphi = \sim \psi_1$. Dann gilt $I \models \varphi$ genau dann, wenn nicht $I \models \psi_1$.

3. Sei $\varphi = (\psi_1 \wedge \psi_2)$. Dann gilt $I \models \varphi$ genau dann, wenn $I \models \psi_1$ und $I \models \psi_2$.
4. Sei $\varphi = (\psi_1 \vee \psi_2)$. Dann gilt $I \models \varphi$ genau dann, wenn $I \models \psi_1$ oder $I \models \psi_2$.
5. Sei $\varphi = (\psi_1 \rightarrow \psi_2)$. Dann gilt $I \models \varphi$ genau dann, wenn $I \models (\sim \psi_1 \vee \psi_2)$.
6. Sei $\varphi = \exists x \psi_1$. Dann gilt $I \models \varphi$ genau dann, wenn eine ERDF-Modifikation $K \in Modi^{VAR}_{ERDF}(I)$ von I bezüglich $VAR = \{x\}$ existiert, so dass $K \models \psi_1$.
7. Sei $\varphi = \forall x \psi_1$. Es gilt $I \models \varphi$ genau dann, wenn für alle ERDF-Modifikationen $K \in Modi^{VAR}_{ERDF}(I)$ von I bezüglich $VAR = \{x\}$ gilt $K \models \psi_1$.

□

Nun legen wir die Erfüllbarkeitsrelation für die negativen ERDF-Formeln fest.

Definition 6.13 (Erfüllbarkeit der negativen ERDF-Formeln)
Sei V_R ein R-Vokabular und Var_{RDF} eine Menge von leeren RDF-Knoten. Des Weiteren sei $I \in \mathcal{IB}_p(V_R, Var_{RDF})$ eine partielle Interpretation über V_R und Var_{RDF} mit $I = \langle \mathcal{S}_I, intU^P_I, intL^P_I, intV^P_I \rangle$ und $\mathcal{S}_I = \langle Res_I, Prop_I, LV_I, rel^+_I, rel^-_I \rangle$ die dazugehörige semantische P-Struktur. Weiter bezeichne $int^*_I : V_R \cup Var_{RDF} \rightarrow Res_I \cup Prop_I$ die durch I eindeutig festgelegte ERDF-Interpretationsabbildung.
Die *ERDF-Erfüllbarkeitsrelation* $\models$ ist für negative ERDF-Formeln wie folgt definiert:

1. Sei $\varphi = \neg p(s,o)$ ein negatives ERDF-Tripel. Dann gilt $I \models \varphi$ genau dann, wenn $p \in URI_{RDF}$, $s, o \in URI_{RDF} \cup Var_{RDF} \cup Lit$, $int^*_I(p) \in Prop_I$ und $\langle int^*_I(s), int^*_I(o) \rangle \in rel^-_I(int^*_I(p))$.
2. Sei $\varphi = \neg \sim \psi_1$. Dann gilt $I \models \varphi$ genau dann, wenn $I \models \psi_1$.
3. Sei $\varphi = \neg (\psi_1 \wedge \psi_2)$. Dann gilt $I \models \varphi$ genau dann, wenn $I \models \neg\psi_1$ oder $I \models \neg\psi_2$.
4. Sei $\varphi = \neg (\psi_1 \vee \psi_2)$. Dann gilt $I \models \varphi$ genau dann, wenn $I \models \neg\psi_1$ und $I \models \neg\psi_2$.
5. Sei $\varphi = \neg (\psi_1 \rightarrow \psi_2)$. Dann gilt $I \models \varphi$ genau dann, wenn $I \models \psi_1$ und $I \models \neg\psi_2$.
6. Sei $\varphi = \neg (\exists x \psi_1)$. Dann gilt $I \models \varphi$ genau dann, wenn $I \models \forall x \neg\psi_1$.
7. Sei $\varphi = \neg (\forall x \psi_1)$. Dann gilt $I \models \varphi$ genau dann, wenn $I \models \exists x \neg\psi_1$.
8. Sei $\varphi = \neg (\neg\psi_1)$. Dann gilt $I \models \varphi$ genau dann, wenn $I \models \psi_1$.

□

Abschließend definieren wir die Erfüllbarkeitsrelation für die restlichen Syntaxkategorien von ERDF.

Definition 6.14 (Erfüllbarkeit der ERDF-Graphen, ERDF-Regeln, ERDF-Ontologien)
Sei V_R ein R-Vokabular und Var_{RDF} eine Menge von leeren RDF-Knoten. Des Weiteren sei $I \in \mathcal{IB}_p(V_R, Var_{RDF})$ eine partielle Interpretation über V_R und Var_{RDF} mit $I = \langle \mathcal{S}_I, intU_I^P, intL_I^P, intV_I^P \rangle$ und $\mathcal{S}_I = \langle Res_I, Prop_I, LV_I, rel_I^+, rel_I^- \rangle$ die dazugehörige semantische P-Struktur. Weiter bezeichne $int_I^* : V_R \cup Var_{RDF} \rightarrow Res_I \cup Prop_I$ die durch I eindeutig festgelegte ERDF-Interpretationsabbildung.
Die *ERDF-Erfüllbarkeitsrelation* $\models$ ist für ERDF-Graphen, ERDF-Regeln und ERDF-Ontologien wie folgt definiert:

1. Sei $G = \{t_1, t_2, \ldots, t_n\}$ ein ERDF-Graph über V_R und Var_{RDF}. Dann gilt $I \models G$ genau dann, wenn eine ERDF-Modifikation $K \in Modi_{ERDF}^{VAR}$ von I bezüglich der Menge $VAR = Var_{RDF}(G)$ existiert mit $K \models t_i$ für alle $1 \leq i \leq n$.

2. Sei $\varphi = true$, dann gilt $I \models \varphi$.

3. Sei $\varphi = false$, dann gilt $I \not\models \varphi$.

4. Sei $r = \psi_1 \Leftarrow \psi_2$ eine ERDF-Regel über V_R und Var_{RDF}. Dann gilt $I \models r$ genau dann, wenn für alle ERDF-Modifikationen $K \in Modi_{ERDF}^{VAR}(I)$ von I bezüglich der Menge $VAR = Var_{RDF}(r)$ gilt: falls $K \models \psi_2$, dann $K \models \psi_1$.

5. Sei $O = \langle G, P \rangle$ eine ERDF-Ontologie über V_R und Var_{RDF}. Dann gilt $I \models O$ genau dann, wenn $I \models G$ und $I \models r$ für alle ERDF-Regeln $r \in P$.

□

Nach Definition 6.14 sind die leeren RDF-Knoten eines ERDF-Graphen implizit durch Existenzquantoren gebunden und werden damit wie leere RDF-Knoten eines RDF-Graphen behandelt. Im Gegensatz dazu werden die freien RDF-Knoten einer ERDF-Regel stets als allquantifizierte leere RDF-Knoten angesehen.

6.2.2 ERDF-Interpretationen

Die partiellen Interpretationen werden nun zu ERDF-Interpretationen verfeinert, um eine erweiterte Funktionalität zur Verfügung zu stellen, die sich an den RDFS-Interpretationen orientiert. Neben dem RDF-Vokabular und dem RDFS-Vokabular definiert eine ERDF-Interpretation außerdem Bedingungen für weitere RDF URI-Referenzen.
Diese RDF URI-Referenzen werden als ERDF-Vokabular bezeichnet.

Definition 6.15 (ERDF-Vokabular)
Es sei `erdf` der Namensraumpräfix des ERDF-Vokabulars.
Das *ERDF-Vokabular* V_{ERDF} ist eine Menge von RDF URI-Referenzen, die folgendermaßen definiert ist:
$V_{ERDF} = \{$`erdf:TotalClass`, `erdf:TotalProperty`$\}$.

□

Der nächste Schritt besteht darin, die semantischen P-Strukturen zu semantischen ERDF-Strukturen zu erweitern.

Definition 6.16 (semantische ERDF-Struktur)
Sei $V_R = URI_{RDF} \dot{\cup} Lit$ ein R-Vokabular.
Eine *semantische ERDF-Struktur* $\mathcal{S} = \langle \overline{\mathcal{S}}, Kl, TKl, TProp, Kext^+, Kext^- \rangle$ besteht aus:

- Einer semantischen P-Struktur $\overline{\mathcal{S}} = \langle Res, Prop, LV, rel^+, rel^- \rangle$.
- Einer Menge $Kl \subseteq Res$, die *Menge der Klassen* von $\mathcal{S}$ genannt wird.
- Einer Menge $TKl \subseteq Kl$, die als *Menge der totalen Klassen* von $\mathcal{S}$ bezeichnet wird.
- Einer Menge $TProp \subseteq Prop$, die *Menge der totalen Propertys* von $\mathcal{S}$ heißt.
- Einer Abbildung $Kext^+ : Kl \to \mathcal{P}(Res)$.
- Einer Abbildung $Kext^- : Kl \to \mathcal{P}(Res)$.

□

Die semantischen ERDF-Strukturen bereichern die semantischen P-Strukturen um eine Menge von Klassen. Im Gegensatz zu RDFS werden auch den Klassen der semantischen ERDF-Strukturen zwei Extensionen zugewiesen. Ein weiterer Unterschied zwischen den semantischen ERDF-Strukturen und den semantischen RDFS-Strukturen liegt in der zusätzlichen Erweiterung des ERDF-Universums um die Menge der totalen Klassen und die Menge der totalen Propertys.

Unter Verwendung der semantischen ERDF-Strukturen sind wir nun in der Lage, die ERDF-Interpretationen zu definieren.

Definition 6.17 (ERDF-Interpretation)
Sei V_R ein R-Vokabular und Var_{RDF} eine Menge von leeren RDF-Knoten.
Eine *ERDF-Interpretation* $I = \langle \mathcal{S}, intU^P, intL^P, intV^P \rangle$ über V_R und Var_{RDF} erfüllt die folgenden Bedingungen:

1. $\mathcal{S} = \langle \overline{\mathcal{S}}, Kl, TKl, TProp, Kext^+, Kext^- \rangle$ ist eine semantische ERDF-Struktur.
2. $\langle \overline{\mathcal{S}}, intU^P, intL^P, intV^P \rangle$ ist eine partielle Interpretation über $V_R \cup V_{RDF} \cup V_{RDFS} \cup V_{ERDF}$ und Var_{RDF}.
3. $x \in Kext^+(y)$ genau dann, wenn $\langle x, y \rangle \in rel^+(int^*(\texttt{rdf:type}))$ und $x \in Kext^-(y)$ genau dann, wenn $\langle x, y \rangle \in rel^-(int^*(\texttt{rdf:type}))$.
4. $Prop = Kext^+(int^*(\texttt{rdf:Property}))$
5. $Kl = Kext^+(int^*(\texttt{rdfs:Class}))$
6. $Res = Kext^+(int^*(\texttt{rdfs:Resource}))$

7. $LV = Kext^+(int^*(\texttt{rdfs:Literal}))$
8. $TKl = Kext^+(int^*(\texttt{erdf:TotalClass}))$
9. $TProp = Kext^+(int^*(\texttt{erdf:TotalProperty}))$
10. Falls $\langle x, y\rangle \in rel^+(int^*(\texttt{rdfs:domain}))$ und $\langle z, w\rangle \in rel^+(x)$, dann $z \in Kext^+(y)$.
11. Falls $\langle x, y\rangle \in rel^+(int^*(\texttt{rdfs:range}))$ und $\langle z, w\rangle \in rel^+(x)$, dann $w \in Kext^+(y)$.
12. Falls $x \in Kl$, dann $\langle x, int^*(\texttt{rdfs:Resource})\rangle \in rel^+(int^*(\texttt{rdfs:subClassOf}))$.
13. Falls $\langle x, y\rangle \in rel^+(int^*(\texttt{rdfs:subClassOf}))$, dann $x, y \in Kl$, $Kext^+(x) \subseteq Kext^+(y)$ und $Kext^-(y) \subseteq Kext^-(x)$.
14. $rel^+(int^*(\texttt{rdfs:subClassOf}))$ ist reflexiv und transitiv auf der Menge Kl.
15. Falls $\langle x, y\rangle \in rel^+(int^*(\texttt{rdfs:subPropertyOf}))$, dann $x, y \in Prop$, $rel^+(x) \subseteq rel^+(y)$ und $rel^-(y) \subseteq rel^-(x)$.
16. $rel^+(int^*(\texttt{rdfs:subPropertyOf}))$ ist reflexiv und transitiv auf der Menge $Prop$.
17. Falls $x \in Kext^+(int^*(\texttt{rdfs:Datatype}))$, dann $\langle x, int^*(\texttt{rdfs:Literal})\rangle \in rel^+(int^*(\texttt{rdfs:subClassOf}))$.
18. Falls $x \in Kext^+(int^*(\texttt{rdfs:ContainerMembershipProperty}))$, dann $\langle x, int^*(\texttt{rdfs:member})\rangle \in rel^+(int^*(\texttt{rdfs:subPropertyOf}))$.
19. Falls $x \in TKl$, dann $Kext^+(x) \cup Kext^-(x) = Res$.
20. Falls $x \in TProp$, dann $rel^+(x) \cup rel^-(x) = Res \times Res$.
21. Falls $\texttt{"s"\^\^rdf:XMLLiteral} \in V_R$ und s ein wohlgeformtes XML-Literal ist, dann gilt:
    - $intL^P(\texttt{"s"\^\^rdf:XMLLiteral})$ ist der XML-Wert von s
    - $intL^P(\texttt{"s"\^\^rdf:XMLLiteral}) \in Kext^+(int^*(\texttt{rdf:XMLLiteral}))$
22. Falls $\texttt{"s"\^\^rdf:XMLLiteral} \in V_R$ und s ein nicht wohlgeformtes XML-Literal ist, dann gilt:
    - $intL^P(\texttt{"s"\^\^rdf:XMLLiteral}) \notin LV$
    - $intL^P(\texttt{"s"\^\^rdf:XMLLiteral}) \in Kext^-(int^*(\texttt{rdfs:Literal}))$
23. I erfüllt die axiomatischen RDF-Tripel von Abbildung 5.2 auf Seite 99.

24. I erfüllt die axiomatischen RDFS-Tripel von Abbildung 5.3 auf Seite 103 und Abbildung 5.4 auf Seite 104.

25. I erfüllt die axiomatischen ERDF-Tripel von Abbildung 6.2.

□

Die Menge der ERDF-Interpretationen über dem R-Vokabular V_R und der Menge der leeren RDF-Knoten Var_{RDF} nennen wir *ERDF-Interpretationsbereich* über V_R und Var_{RDF}. Wir bezeichnen diese Menge mit $\mathcal{IB}_{ERDF}(V_R, Var_{RDF})$.

6.2.3 Kohärente ERDF-Interpretationen

In [5] sowie Kapitel 6.6 werden nicht alle ERDF-Interpretationen betrachtet, sondern nur die sogenannten kohärenten ERDF-Interpretationen.

Definition 6.18 (kohärente ERDF-Interpretation)
Sei V_R ein R-Vokabular und Var_{RDF} eine Menge von leeren RDF-Knoten. Weiterhin sei $I \in \mathcal{IB}_{ERDF}(V_R, Var_{RDF})$ eine ERDF-Interpretation über V_R und Var_{RDF} mit $I = \langle \mathcal{S}, intU^P, intL^P, intV^P \rangle$ und der dazugehörigen semantischen ERDF-Struktur $\mathcal{S} = \langle \langle Res, Prop, LV, rel^+, rel^- \rangle, Kl, TKl, TProp, Kext^+, Kext^- \rangle$.
Die ERDF-Interpretation I ist eine *kohärente ERDF-Interpretation* genau dann, wenn $rel^+(x) \cap rel^-(x) = \emptyset$ für alle $x \in Prop$.

□

Entsprechend der Festlegung für ERDF-Interpretationen nennen wir die Menge der kohärenten ERDF-Interpretationen über V_R und Var_{RDF} den *kohärenten ERDF-Interpretationsbereich* und vereinbaren dafür die Bezeichnung $\mathcal{IB}^{ko}_{ERDF}(V_R, Var_{RDF})$.
Für kohärente ERDF-Interpretationen gilt das nachfolgende Lemma, welches wir aus [4] übernehmen.

Lemma 6.19 (Proposition 3, Seite 14 in [4])
Sei $I \in \mathcal{IB}^{ko}_{ERDF}(V_R, Var_{RDF})$ eine kohärente ERDF-Interpretation über V_R und Var_{RDF} mit $I = \langle \mathcal{S}, intU^P, intL^P, intV^P \rangle$ und der dazugehörigen semantischen ERDF-Struktur

`rdfs:subClassOf` (`erdf:TotalClass`, `rdfs:Class`)
`rdfs:subClassOf` (`erdf:TotalProperty`, `rdf:Property`)[19]

Abbildung 6.2: Die axiomatischen ERDF-Tripel

[19] Wir verwenden die strengere Forderung aus [4] anstatt des in [5] angegebenen ERDF-Tripels `rdfs:subClassOf` (`erdf:TotalProperty`, `rdfs:Class`).

$\mathcal{S} = \langle \langle Res, Prop, LV, rel^+, rel^- \rangle, Kl, TKl, TProp, Kext^+, Kext^- \rangle$.
Dann gilt $Kext^+(x) \cap Kext^-(x) = \emptyset$ *für alle* $x \in Kl$.

Für eine kohärente ERDF-Interpretation I sind nicht nur die beiden Extensionen jeder Property disjunkt, sondern auch die zwei Extensionen jeder Klasse von I.

Abschließend wird die Folgerungsrelation für kohärente ERDF-Interpretationen betrachtet.

Definition 6.20 (Modellmenge, Folgerungsrelation)
Sei V_R ein R-Vokabular und Var_{RDF} eine Menge von leeren RDF-Knoten. Weiterhin seien $G_1, G_2 \in Gr_{ERDF}(V_R, Var_{RDF})$ zwei ERDF-Graphen über V_R und Var_{RDF}.

(a) Die *kERDF-Modellmenge* eines ERDF-Graphen G_1 definieren wir wie folgt:
$Mod^{ko}_{ERDF}(G_1) = \left\{ I \in \mathcal{IB}^{ko}_{ERDF}(V_R, Var_{RDF}) \;\middle|\; I \models G_1 \right\}$.

(b) Die *kERDF-Folgerungsrelation* $\models_{kERDF}$ ist definiert als:
$G_1 \models_{kERDF} G_2$ genau dann, wenn $Mod^{ko}_{ERDF}(G_1) \subseteq Mod^{ko}_{ERDF}(G_2)$. □

In [5] wurde folgender Zusammenhang zwischen der RDFS-Folgerungsrelation und der kERDF-Folgerungsrelation gezeigt:

Satz 6.21 (Proposition 3.2, Seite 49 in [5])
Es sei V_R *ein R-Vokabular und* Var_{RDF} *eine Menge von leeren RDF-Knoten. Des Weiteren seien* $G_1, G_2 \in Gr_{RDF}(V_R, Var_{RDF})$ *zwei RDF-Graphen über* V_R *und* Var_{RDF} *mit* $V_R(G_1) \cap V_{ERDF} = \emptyset$ *und* $V_R(G_2) \cap V_{ERDF} = \emptyset$.
Dann $G_1 \models_{kERDF} G_2$ *genau dann, wenn* $G_1 \models_{RDFS} G_2$.

6.3 Charakterisierung des ERDF-Universums

In diesem Unterkapitel werden wir den Aufbau des ERDF-Universums genauer untersuchen. Als Erstes halten wir die folgende Beobachtung fest:

Beobachtung 6.22
Es sei $I \in \mathcal{IB}_{ERDF}(V_R, Var_{RDF})$ *eine ERDF-Interpretation über* V_R *und* Var_{RDF} *mit* $I = \langle \mathcal{S}, intU^P, intL^P, intV^P \rangle$ *und der dazugehörigen semantischen ERDF-Struktur* $\mathcal{S} = \langle \langle Res, Prop, LV, rel^+, rel^- \rangle, Kl, TKl, TProp, Kext^+, Kext^- \rangle$.
Dann gilt $Prop \subseteq Res$.

Beweis:
Nach Definition 6.16 gilt $Kext^+ : Kl \to \mathcal{P}(Res)$.
Damit ist $Kext^+(int^*(\texttt{rdf:Property})) \subseteq Res$.
Also $Prop \subseteq Res$, nach 4. Bedingung von Definition 6.17.

■

Für partielle Interpretationen ist diese Beobachtung nicht allgemeingültig.

Nun sehen wir uns eine konkrete ERDF-Interpretation an, die zur Veranschaulichung des ERDF-Universums helfen soll.

Beispiel 6.23
Es sei $V_R = URI_{RDF} \dot{\cup} Lit$ ein R-Vokabular bestehend aus $URI_{RDF} = \{\texttt{ex:a}\}$, $uLit = \{\texttt{"s"}\}$ und $tLit = \{\texttt{1}\}$. Die Menge der leeren RDF-Knoten sei definiert als $Var_{RDF} = \{x\}$.
Des Weiteren seien die ERDF-Interpretation $I \in \mathcal{IB}_{ERDF}(V_R, Var_{RDF})$ über V_R und Var_{RDF} mit $I = \langle \mathcal{S}, intU^P, intL^P, intV^P \rangle$ und die zu I gehörende semantische ERDF-Struktur $\mathcal{S} = \left\langle \langle Res, Prop, LV, rel^+, rel^- \rangle, Kl, TKl, TProp, Kext^+, Kext^- \right\rangle$ wie folgt definiert:

1. $Res = \{\texttt{s}, u_1, u_2, u_3, u_4, u_5\}$
2. $Prop = \{u_2, u_3\}$
3. $LV = \{\texttt{s}, u_1\}$
4. $Kl = \{u_4, u_5\}$
5. $TKl = \{u_5\}$
6. $TProp = \{u_2\}$
7. die Abbildung $rel^+ : Prop \to \mathcal{P}(Res \times Res)$ ist definiert als:
   - $rel^+(u_2) = \{\langle \texttt{s}, u_1 \rangle, \ldots, \langle \texttt{s}, u_5 \rangle, \langle u_1, u_1 \rangle\}$
   - $rel^+(u_3) = \{\langle u_1, u_2 \rangle, \langle u_1, u_3 \rangle\}$
8. die Abbildung $rel^- : Prop \to \mathcal{P}(Res \times Res)$ ist definiert als:
   - $rel^-(u_2) = \{\langle u_1, u_2 \rangle, \ldots, \langle u_1, u_5 \rangle, \langle u_2, u_1 \rangle, \ldots, \langle u_5, u_5 \rangle\}$
   - $rel^-(u_3) = \{\langle u_1, u_4 \rangle, \langle u_1, u_5 \rangle, \langle u_2, u_1 \rangle, \ldots, \langle u_5, u_5 \rangle\}$
9. die Abbildung $Kext^+ : Kl \to \mathcal{P}(Res)$ ist definiert als:
   - $Kext^+(u_4) = \{u_2\}$
   - $Kext^+(u_5) = \{\texttt{s}, u_1, u_2, u_3\}$
10. die Abbildung $Kext^- : Kl \to \mathcal{P}(Res)$ ist definiert als:
    - $Kext^-(u_4) = \{u_4, u_5\}$
    - $Kext^-(u_5) = \{u_4, u_5\}$
11. die Abbildung $intU^P : URI_{RDF} \to Res \cup Prop$ ist definiert als:
    - $intU^P(\texttt{ex:a}) = u_3$
12. die Abbildung $intL^P : tLit \to Res$ ist definiert als:
    - $intL^P(\texttt{1}) = u_1$
13. die Abbildung $intV^P : Var_{RDF} \to Res$ ist definiert als:

- $intV^P(x) = u_2$

□

Das ERDF-Universum ist mit dem RDFS-Universum vergleichbar, es erweitert das RDFS-Universum jedoch an einigen Stellen. Daher ähnelt die Abbildung 6.3 auf der nächsten Seite, welche das ERDF-Universum von Beispiel 6.23 skizziert, der Abbildung 5.5.

Sowohl in RDFS als auch in ERDF wird auf der syntaktischen Ebene nicht zwischen Relationsbezeichnern und Konstantenbezeichnern unterschieden, sondern man differenziert auch in ERDF zwischen leeren RDF-Knoten, RDF-Literalen und RDF URI-Referenzen. Die Interpretation dieser Bezeichner erfolgt in ERDF ebenfalls in zwei getrennten Stufen. Zunächst werden alle Namen unter Verwendung der Funktionen $intU^P$, $intL^P$ und $intV^P$ in die Menge *Res* abgebildet, bevor den einzelnen Relationen des Universums ihre Extensionen zugewiesen werden. In Übereinstimmung mit RDFS werden dabei für Klassen und Propertys verschiedene Abbildungen verwendet. Da in ERDF die Menge der Klassen und die Menge der Propertys nicht disjunkt sein müssen, existieren in ERDF ebenfalls Relationen mit einer variablen Stelligkeit. Eine weitere Gemeinsamkeit von RDFS und ERDF ist das Vorhandensein spezieller RDF URI-Referenzen, die eine festgelegte Semantik besitzen. Die ERDF-Interpretationen erweitern jedoch das RDF-Vokabular und das RDFS-Vokabular um die beiden RDF URI-Referenzen `erdf:TotalClass` und `erdf:TotalProperty`.

Das ERDF-Universum und das RDFS-Universum unterscheiden sich vor allem in zwei Eigenschaften voneinander. Einerseits beinhaltet das ERDF-Universum mit der Menge der totalen Klassen und der Menge der totalen Propertys zwei zusätzliche Mengen. Andererseits werden in ERDF jedem Relationssymbol mittels der Abbildungen rel^+ und rel^- bzw. $Kext^+$ und $Kext^-$ zwei Extensionen zugeordnet, während in RDFS jede Relation genau eine Extension besitzt.

6.4 Partielle Propertys und Klassen

Wie im vorangegangenen Abschnitt bereits angesprochen, verwendet ERDF für jede Relation eine positive und eine negative Extension. Dies resultiert aus der Tatsache, dass die Semantik von ERDF auf *Partial Logic* [42] basiert, die eine Erweiterung der Prädikatenlogik erster Stufe ist. Nach Definition 6.12 ist für die Erfüllbarkeit eines positiven ERDF-Tripels $p(s,o)$ die positive Extension von p, also $rel^+(int^*(p))$, von Bedeutung. Für die Erfüllbarkeit des negativen ERDF-Tripels $\neg p(s,o)$ ist hingegen die negative Extension von p, welche mit $rel^-(int^*(p))$ bezeichnet wird, relevant. Da diese beiden Extensionen unabhängig voneinander sind, ist in ERDF das Bivalenzprinzip[20] nicht gültig. Stattdessen ist für jede ERDF-Formel φ und jede

[20] Das Bivalenzprinzip besagt, dass für jede ERDF-Formel φ und jede ERDF-Interpretation I entweder $I \models \varphi$ und $I \not\models \neg\varphi$ oder $I \models \neg\varphi$ und $I \not\models \varphi$ gilt.

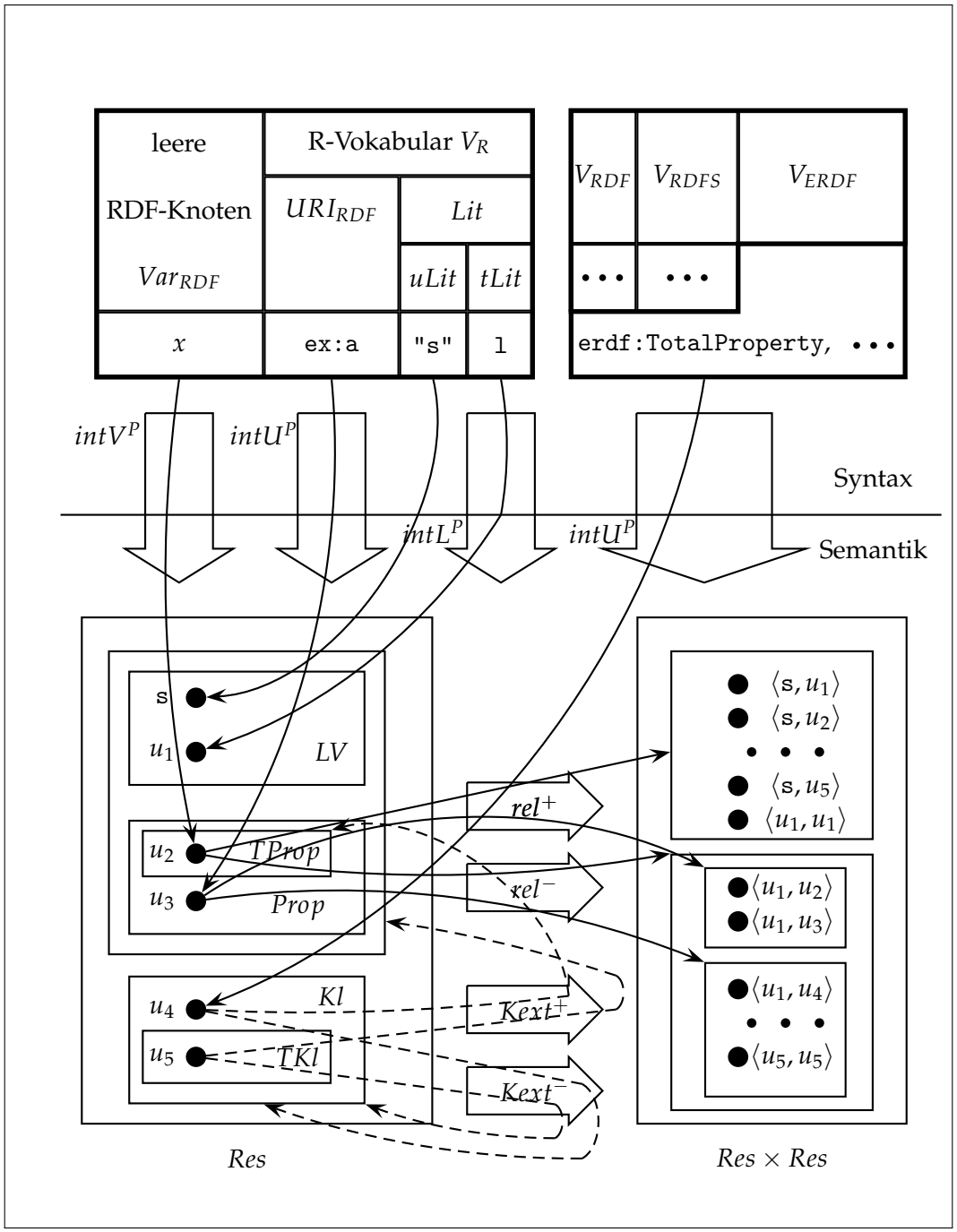

Abbildung 6.3: Schematische Darstellung des ERDF-Universums von Beispiel 6.23

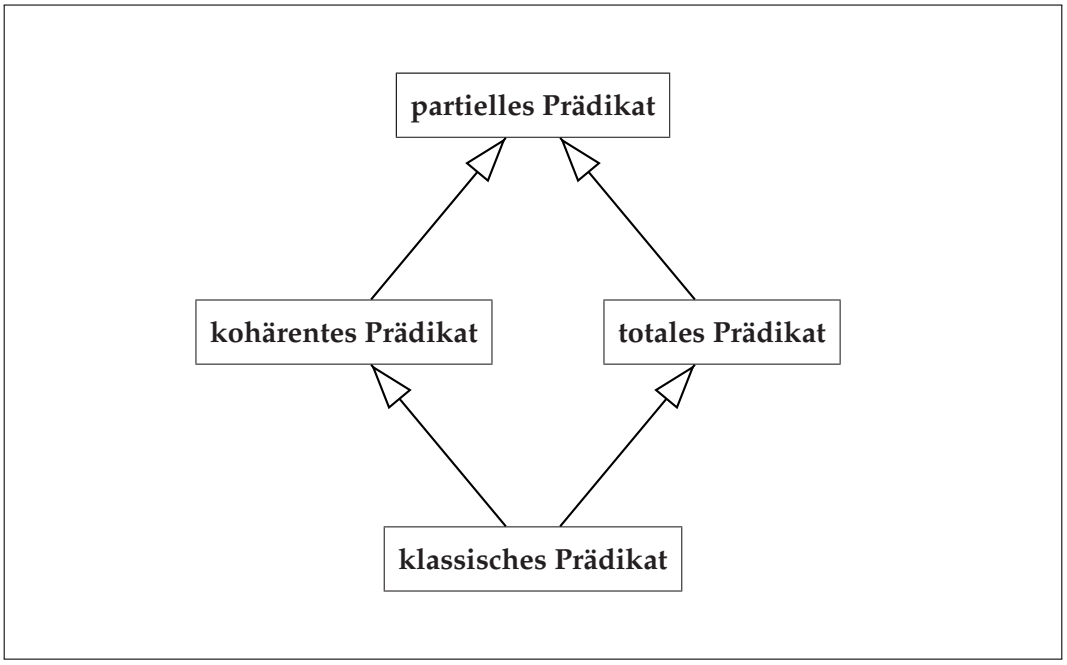

Abbildung 6.4: Die Klassifizierung der Prädikate von Partial Logic nach [4]

ERDF-Interpretation I genau einer der folgenden Fälle zutreffend:

1. Fall: $I \not\models \varphi$ und $I \not\models \neg\varphi$ (unbekannt)
2. Fall: $I \models \varphi$ und $I \not\models \neg\varphi$ (wahr)
3. Fall: $I \not\models \varphi$ und $I \models \neg\varphi$ (falsch)
4. Fall: $I \models \varphi$ und $I \models \neg\varphi$ (widersprüchlich)

Unter Verwendung von zusätzlichen Bedingungen kann die Anzahl der möglichen Fälle eingeschränkt werden. Je nachdem welche der oben genannten Situationen eintreten können, lassen sich die Prädikate von Partial Logic in verschiedene Klassen einteilen. Die Abbildung 6.4 gibt diese Klassifizierung der Prädikate von Partial Logic an. Die *partiellen Prädikate* sind die allgemeinsten Prädikate in Partial Logic. Bei diesen können alle vier Zustände, die wir weiter oben genannt haben, eintreten. Falls keine widersprüchlichen Informationen zulässig sind, d.h. der 4. Fall nicht möglich ist, dann spricht man von *kohärenten Prädikaten*. Andererseits werden die Prädikate als *totale Prädikate* bezeichnet, wenn der 1. Fall ausgeschlossen ist. Angenommen jedes Prädikat verhält sich wie ein Prädikat der klassischen Prädikatenlogik erster Stufe, also entweder der 2. Fall oder der 3. Fall treten ein, dann nennen wir diese *klassische Prädikate*.

Die ERDF-Interpretationen erlauben prinzipiell alle der oben genannten Fälle, so dass sowohl partielle Propertys als auch partielle Klassen auftreten. Um einer Property bzw. einer Klasse explizit die Eigenschaft der Totalität zuweisen zu können, wurden in ERDF die beiden RDF URI-Referenzen `erdf:TotalProperty` und `erdf:TotalClass` eingeführt. Entsprechende URI-Referenzen, welche die Kohärenz sicherstellen existieren jedoch nicht.

Bei Anwendung der kohärenten ERDF-Interpretationen ergibt sich ein anderer Sachverhalt. Aufgrund der in Definition 6.18 geforderten Bedingung, dass die beiden Extensionen jeder Property zueinander disjunkt sind, ist für Propertys und nach Lemma 6.19 auch für Klassen der 4. Fall prinzipiell ausgeschlossen. Daher ist jede Property und jede Klasse bezüglich einer kohärenten ERDF-Interpretation eine kohärente Property bzw. eine kohärente Klasse. Falls nun mithilfe des ERDF-Vokabulars die Totalität einer Property gefordert wird, dann ist diese Property aufgrund der Kohärenzeigenschaft eine klassische Property. Für Klassen trifft die selbe Beobachtung zu.

Unter Verwendung der kohärenten ERDF-Interpretationen sind also keine widersprüchlichen Informationen zulässig, aber im Gegensatz zur Prädikatenlogik der ersten Stufe, zu Common Logic und zu RDF/RDFS lassen sich damit unvollständige Informationen modellieren.

6.5 Übertragung nach Common Logic

Obwohl in ERDF unvollständige und widersprüchliche Informationen dargestellt werden können, ist es möglich ERDF-Dokumente, unter Beibehaltung ihrer Semantik, nach Common Logic abzubilden. Wir werden dabei ähnlich wie in Kapitel 5.5 vorgehen und auch für ERDF-Dokumente die sogenannte Einbettung anwenden. Da in Common Logic jede Relation genau eine Extension besitzt und nicht deren zwei, besteht die Hauptaufgabe bei der Übertragung nach Common Logic darin, jedes Relationssymbol einer ERDF-Ontologie durch zwei neue Relationssymbole zu ersetzen und diesen jeweils eine der beiden Extensionen zuzuordnen. Diese Idee geht auf die sogenannte *Gilmore-Übersetzung* [42] zurück. Aufgrund der Tatsache, dass in Common Logic keine syntaktische Trennung zwischen Relations- und Individuenbezeichnern existiert, ist es möglich, dass ein Relationssymbol auch als Individuenname interpretiert wird. Daher ersetzen wir die bisherigen Relationssymbole nicht direkt, sondern wir führen die beiden Funktionsbezeichner `pos` und `neg` ein und verwenden den CL-Funktionsterm $(\texttt{pos}\ P)$ um die positive Extension der Relation P zu bezeichnen. In gleicher Weise sprechen wir mit $(\texttt{neg}\ P)$ die negative Extension von P an. Um zwischen positiven und negativen ERDF-Formeln unterscheiden zu können und damit zu wissen, welche der beiden Extensionen von Bedeutung ist, benötigen wir für die Übertragung nach Common Logic zwei Abbildungen. Diese bezeichnen wir mit $ERDFtoCL^{+}$ und $ERDFtoCL^{-}$.

Die Übertragung nach Common Logic erfolgt auch hier in mehreren Schritten. Wir beginnen zunächst damit die partiellen Interpretationen in Common Logic einzubetten.

6.5.1 Einbettung der partiellen Interpretationen

Für die Übertragung der ERDF-Dokumente nach Common Logic ist eine Funktion notwendig, welche die ERDF-Ausdrücke auf äquivalente Konstruktionen in Common Logic abbildet. Wir führen für diesen Zweck die Abbildung

$$ERDFtoCL^{+} : On_{ERDF}\left(V_R, Var_{RDF}\right) \rightarrow Fm(V_{CL}) \tag{6.1}$$

ein, die jeder ERDF-Ontologie eine Boolesche CL-Formel zuordnet. Zusätzlich verwenden wir die Abbildung

$$ERDFtoCL^{-} : Fm_{ERDF}\left(V_R, Var_{RDF}\right) \rightarrow Fm(V_{CL}) \tag{6.2}$$

Diese Funktion weist jeder ERDF-Formel eine entsprechende CL-Formel zu und dient der Kennzeichnung, dass gerade eine negative ERDF-Formel nach Common Logic übersetzt wird. Die Definition der beiden Funktionen erfolgt induktiv über den Aufbau der ERDF-Ausdrücke.

Die Grundlage der syntaktischen Konstruktionen von ERDF bilden die elementaren Syntaxkategorien von RDF/RDFS. Diese werden wie in Kapitel 5.5.1 angegeben nach Common Logic überführt. Es gilt also:

$$ERDFtoCL^{+}\left(x\right) := RDFtoCL\left(x\right) \text{ für alle } x \in URI_{RDF} \cup Var_{RDF} \cup Lit \tag{6.3}$$

und

$$ERDFtoCL^{-}\left(x\right) := RDFtoCL\left(x\right) \text{ für alle } x \in URI_{RDF} \cup Var_{RDF} \cup Lit \tag{6.4}$$

Der nächste Schritt besteht darin, die positiven ERDF-Formeln nach Common Logic abzubilden. Wir führen auch diesmal spezielle Relationsnamen ein, um die ERDF-Tripel in Common Logic zu kennzeichnen. Für positive ERDF-Tripel verwenden wir die Bezeichnung `ERDF_TRIPEL+`. Negative ERDF-Tripel werden mittels `ERDF_TRIPEL-` kenntlich gemacht.
Sei $p\left(s, o\right)$ ein positives ERDF-Tripel, dann definieren wir:

$$\begin{aligned} ERDFtoCL^{+}\left(p\left(s,o\right)\right) = (&\texttt{ERDF\_TRIPEL+} \\ &ERDFtoCL^{+}\left(p\right) \\ &ERDFtoCL^{+}\left(s\right) \\ &ERDFtoCL^{+}\left(o\right) \\ &) \end{aligned} \tag{6.5}$$

und

$$\begin{aligned} ERDFtoCL^{-}(p(s,o)) = (&\texttt{ERDF\_TRIPEL-} \\ &ERDFtoCL^{-}(p) \\ &ERDFtoCL^{-}(s) \\ &ERDFtoCL^{-}(o) \\ &) \end{aligned} \tag{6.6}$$

Zusätzlich führen wir für ERDF-Tripel die beiden Axiome Ax_1 und Ax_2 ein, welche das Axiom aus 5.10 nachbilden, jedoch zwischen positiven und negativen ERDF-Tripeln unterscheiden.

$$\begin{aligned} Ax_1 = (&\mathsf{forall}\,(P\ S\ O)\,(\mathsf{iff} \\ &(\texttt{ERDF\_TRIPEL+}\ P\ S\ O) \\ &((\texttt{pos}\ P)\ S\ O) \\ &)) \end{aligned} \tag{6.7}$$

$$\begin{aligned} Ax_2 = (&\mathsf{forall}\,(P\ S\ O)\,(\mathsf{iff} \\ &(\texttt{ERDF\_TRIPEL-}\ P\ S\ O) \\ &((\texttt{neg}\ P)\ S\ O) \\ &)) \end{aligned} \tag{6.8}$$

Unter Verwendung der Junktoren $\sim$, $\wedge$, $\vee$, $\rightarrow$ sowie der beiden Quantoren $\exists$ und $\forall$ werden aus den ERDF-Tripeln komplexe ERDF-Formeln erstellt. Diese Konstruktionsschritte sind in Common Logic ebenfalls möglich, so dass für jede positive ERDF-Formel eine semantisch äquivalente CL-Formel existiert. Für positive ERDF-Formeln $\varphi \in Fm_{ERDF}(V_R, Var_{RDF})$ ist die Abbildung $ERDFtoCL^{+}$ daher wie folgt definiert:

$$ERDFtoCL^{+}(\varphi) = \begin{cases} \Big(\mathsf{not}\ ERDFtoCL^{+}(\psi)\Big) & \text{falls } \varphi = \sim\psi \\ \Big(\mathsf{and}\ ERDFtoCL^{+}(\psi)\ ERDFtoCL^{+}(\chi)\Big) & \text{falls } \varphi = (\psi \wedge \chi) \\ \Big(\mathsf{or}\ ERDFtoCL^{+}(\psi)\ ERDFtoCL^{+}(\chi)\Big) & \text{falls } \varphi = (\psi \vee \chi) \\ \Big(\mathsf{if}\ ERDFtoCL^{+}(\psi)\ ERDFtoCL^{+}(\chi)\Big) & \text{falls } \varphi = (\psi \rightarrow \chi) \\ \Big(\mathsf{exists}\ (ERDFtoCL^{+}(x))\ ERDFtoCL^{+}(\psi)\Big) & \text{falls } \varphi = \exists x\psi \\ \Big(\mathsf{forall}\ (ERDFtoCL^{+}(x))\ ERDFtoCL^{+}(\psi)\Big) & \text{falls } \varphi = \forall x\psi \end{cases} \tag{6.9}$$

Die Abbildung $ERDFtoCL^{-}$ legen wir für positive ERDF-Formeln in ähnlicher Weise fest. Allerdings bedeutet die Anwendung von $ERDFtoCL^{-}$, dass es sich in

Wirklichkeit um eine negative ERDF-Formel handelt.
Sei $\varphi \in Fm_{ERDF}(V_R, Var_{RDF})$, dann gilt:

$$ERDFtoCL^-(\varphi) = \begin{cases} ERDFtoCL^+(\psi) & \text{falls } \varphi = \sim\psi \\ \Big(\mathsf{or}\ ERDFtoCL^-(\psi)\ ERDFtoCL^-(\chi)\Big) & \text{falls } \varphi = (\psi \wedge \chi) \\ \Big(\mathsf{and}\ ERDFtoCL^-(\psi)\ ERDFtoCL^-(\chi)\Big) & \text{falls } \varphi = (\psi \vee \chi) \\ \Big(\mathsf{and}\ ERDFtoCL^+(\psi)\ ERDFtoCL^-(\chi)\Big) & \text{falls } \varphi = (\psi \rightarrow \chi) \\ \Big(\mathsf{forall}\ (ERDFtoCL^+(x))\ ERDFtoCL^-(\psi)\Big) & \text{falls } \varphi = \exists x\psi \\ \Big(\mathsf{exists}\ (ERDFtoCL^+(x))\ ERDFtoCL^-(\psi)\Big) & \text{falls } \varphi = \forall x\psi \end{cases} \tag{6.10}$$

Nach Definition 6.13 ist die ERDF-Formel $\neg\neg\varphi$ semantisch äquivalent mit der ERDF-Formel φ. Daher bewirkt eine negative ERDF-Formel einen Wechsel zwischen den beiden Abbildungen, welche wir für die Übertragung nach Common Logic nutzen. Sei $\neg\varphi \in Fm_{ERDF}(V_R, Var_{RDF})$ eine negative ERDF-Formel, dann gilt:

$$ERDFtoCL^+(\neg\varphi) = ERDFtoCL^-(\varphi) \tag{6.11}$$

und

$$ERDFtoCL^-(\neg\varphi) = ERDFtoCL^+(\varphi) \tag{6.12}$$

Damit ist die Abbildung $ERDFtoCL^- : Fm_{ERDF}(V_R, Var_{RDF}) \rightarrow Fm(V_{CL})$ vollständig spezifiziert. Eine zusammenfassende Übersicht dieser Funktion wird mithilfe der Tabelle 6.1 auf Seite 147 gegeben.

Unter Verwendung der Abbildung $ERDFtoCL^+$ werden wir nun die noch verbleibenden ERDF-Konstruktionen nach Common Logic übertragen. Wir wenden uns zunächst den ERDF-Graphen zu, welche auf die gleiche Weise interpretiert werden wie die RDF-Graphen. Also ist ein ERDF-Graph G semantisch mit einer Konjunktion über alle ERDF-Tripel von G vergleichbar. Zusätzlich befinden sich diese ERDF-Tripel in dem Wirkungsbereich eines Existenzquantors, welcher alle leeren RDF-Knoten von G bindet.
Sei $G = \{t_1, t_2, \ldots, t_n\}$ ein ERDF-Graph, welcher aus den ERDF-Tripeln $t_1, t_2, \ldots, t_n$ besteht. Weiterhin sei $Var_{RDF}(G) = \{x_1, x_2, \ldots, x_m\}$ die Menge der leeren RDF-Knoten von G, dann gilt:

$$
\begin{aligned}
&ERDFtoCL^{+}(G) = \\
&\quad (\text{exists } (ERDFtoCL^{+}(x_1)\ \ldots\ ERDFtoCL^{+}(x_m))\ (\text{and} \\
&\qquad\qquad ERDFtoCL^{+}(t_1) \\
&\qquad\qquad ERDFtoCL^{+}(t_2) \\
&\qquad\qquad \ldots \\
&\qquad\qquad ERDFtoCL^{+}(t_n) \\
&\qquad))
\end{aligned}
\tag{6.13}
$$

Als nächstes überführen wir die ERDF-Regeln nach Common Logic. Dazu ist es notwendig den Funktionswert der Abbildung $ERDFtoCL^{+}$ für die beiden Konstanten $true$ und $false$ zu definieren. Da $true$ dem Wahrheitswert `WAHR` und $false$ dem Wahrheitswert `FALSCH` entspricht und diese in Common Logic mithilfe der leeren Konjunktion bzw. der leeren Disjunktion simuliert werden, erweitern wir die Abbildung $ERDFtoCL^{+}$ wie folgt:

$$
ERDFtoCL^{+}(x) = \begin{cases} (\text{and}) & \text{falls } x = true \\ (\text{or}) & \text{falls } x = false \end{cases}
\tag{6.14}
$$

Die Semantik einer ERDF-Regel stimmt mit der Semantik einer Implikation überein in der alle leeren RDF-Knoten durch einen Allquantor gebunden sind.
Es sei $r = \psi_1 \Leftarrow \psi_2$ eine ERDF-Regel, welche die leeren RDF-Knoten $Var_{RDF}(r) = \{x_1, x_2, \ldots, x_n\}$ beinhaltet, dann gilt also:
$ERDFtoCL^{+}(r) =$

$$
\begin{aligned}
&(\text{forall } (ERDFtoCL^{+}(x_1)\ \ldots\ ERDFtoCL^{+}(x_n))\ (\text{if} \\
&\qquad\qquad ERDFtoCL^{+}(\psi_2) \\
&\qquad\qquad ERDFtoCL^{+}(\psi_1) \\
&\qquad))
\end{aligned}
\tag{6.15}
$$

Ein ERDF-Programm P besteht aus einer Menge von ERDF-Regeln. Nach Definition 6.14 müssen alle ERDF-Regeln von P durch eine partielle Interpretation erfüllt sein, um auch P zu erfüllen. Somit ist ein ERDF-Programm semantisch äquivalent zu einer Konjunktion über die entsprechenden ERDF-Regeln. Wir definieren die Abbildung $ERDFtoCL^{+}$ für das ERDF-Programm $P = \{r_1, r_2, \ldots, r_n\}$ bestehend aus den ERDF-Regeln $r_1, r_2, \ldots, r_n$ daher folgendermaßen:

$$
\begin{aligned}
ERDFtoCL^{+}(P) = (&\text{and} \\
&ERDFtoCL^{+}(r_1) \\
&ERDFtoCL^{+}(r_2) \\
&\ldots \\
&ERDFtoCL^{+}(r_n) \\
&)
\end{aligned}
\tag{6.16}
$$

Ein ERDF-Graph und ein ERDF-Programm werden zu einer ERDF-Ontologie zusammengefasst. Diese wird genau dann von einer partiellen Interpretation I erfüllt, wenn I die beiden Komponenten der ERDF-Ontologie erfüllt. Demzufolge entspricht die Konjunktion über diese Komponenten genau der Semantik der betreffenden ERDF-Ontologie.
Sei $O = \langle G, P \rangle$ eine ERDF-Ontologie, dann gilt:

$$ERDFtoCL^{+}(O) = \big(\mathsf{and}\ ERDFtoCL^{+}(G)\ ERDFtoCL^{+}(P)\big) \tag{6.17}$$

Wir haben nun für jede abstrakte Syntaxkategorie von ERDF angegeben, wie diese mithilfe der Abbildung $ERDFtoCL^{+} : On_{ERDF}(V_R, Var_{RDF}) \rightarrow Fm(V_{CL})$ nach Common Logic übertragen wird. Die vollständige Definition dieser Funktion ist durch Tabelle 6.2 auf Seite 148 und Tabelle 6.3 auf Seite 149 noch einmal zusammenfassend dargestellt.

6.5.2 Einbettung der ERDF-Interpretationen

Im vorherigen Abschnitt haben wir unter Verwendung der beiden Abbildungen $ERDFtoCL^{+}$ und $ERDFtoCL^{-}$ sowie den beiden Axiomen Ax_1 und Ax_2 die partiellen Interpretationen nach Common Logic übertragen. In diesem Teil der Diplomarbeit werden wir eine ähnliche Einbettung für die ERDF-Interpretationen konstruieren. Da die ERDF-Interpretationen auf den selben abstrakten Syntaxkategorien wie die partiellen Interpretationen basieren, werden wir für die Überführung nach Common Logic ebenfalls die in Kapitel 6.5.1 definierten Abbildungen verwenden. Aufgrund der zusätzlichen Bedingungen für ERDF-Interpretationen benötigen wir jedoch weitere Axiome, um die Semantik vollständig in Common Logic nachzubilden.

Des Weiteren nutzen die ERDF-Interpretationen sowohl das RDF-Vokabular als auch das RDFS-Vokabular. So bezeichnet zum Beispiel die positive Extension der RDF URI-Referenz `rdf:Property` die Menge aller Propertys. Dadurch sind wir in der Lage, die beiden Axiome Ax_1 (6.7) und Ax_2 (6.8) wie folgt zu erweitern:

$$\begin{aligned}
Ax_1' = (&\mathsf{forall}\ (P\ S\ O)\ (\mathsf{iff} \\
&\quad (\texttt{ERDF_TRIPEL+}\ P\ S\ O) \\
&\quad (\mathsf{and} \\
&\qquad ((\texttt{pos}\ P)\ S\ O) \\
&\qquad ((\texttt{pos}\ \texttt{rdf:type})\ P\ \texttt{rdf:Property}) \\
&\quad) \\
&))
\end{aligned} \tag{6.18}$$

ERDF-Syntaxkategorie *Aus*	**Definition von** $ERDFtoCL^{-}(Aus)$	**Nr.**
RDF URI-Referenz $<aaa>$	interpretierbarer CL-Name *aaa*	6.4
QName *bb*:*aaa*	interpretierbarer CL-Name *bb*:*aaa*	6.4
leerer RDF-Knoten *xxx*	interpretierbarer CL-Name *xxx*	6.4
ungetyptes RDF-Literal ohne Sprachkennzeichnung `"s"`	quoted String `'s'`	6.4
ungetyptes RDF-Literal mit Sprachkennzeichnung `"s"@t`	CL-Funktionsterm (`CLungeLit 's' 't'`)	6.4
getyptes RDF-Literal *"aaa"*`^^ddd`	CL-Funktionsterm (`ddd 'aaa'`)	6.4
positives ERDF-Tripel $p(s,o)$	CL-Atomformel $\Big($`ERDF_TRIPEL-` $ERDFtoCL^{-}(p)$ $ERDFtoCL^{-}(s)$ $ERDFtoCL^{-}(o)\Big)$	6.6
positive ERDF-Formel $\sim\psi_1$ mit $\psi_1 \in Fm_{ERDF}$	$ERDFtoCL^{+}(\psi_1)$	6.10
positive ERDF-Formel $(\psi_1 \wedge \psi_2)$ mit $\psi_1,\psi_2 \in Fm_{ERDF}$	Boolesche CL-Formel $\Big($or $ERDFtoCL^{-}(\psi_1)\, ERDFtoCL^{-}(\psi_2)\Big)$	6.10
positive ERDF-Formel $(\psi_1 \vee \psi_2)$ mit $\psi_1,\psi_2 \in Fm_{ERDF}$	Boolesche CL-Formel $\Big($and $ERDFtoCL^{-}(\psi_1)\, ERDFtoCL^{-}(\psi_2)\Big)$	6.10
positive ERDF-Formel $(\psi_1 \rightarrow \psi_2)$ mit $\psi_1,\psi_2 \in Fm_{ERDF}$	Boolesche CL-Formel $\Big($and $ERDFtoCL^{+}(\psi_1)\, ERDFtoCL^{-}(\psi_2)\Big)$	6.10
positive ERDF-Formel $\exists x\psi_1$ mit $\psi_1 \in Fm_{ERDF}$	quantifizierte CL-Formel $\Big($forall $(ERDFtoCL^{+}(x))\, ERDFtoCL^{-}(\psi_1)\Big)$	6.10
positive ERDF-Formel $\forall x\psi_1$ mit $\psi_1 \in Fm_{ERDF}$	quantifizierte CL-Formel $\Big($exists $(ERDFtoCL^{+}(x))\, ERDFtoCL^{-}(\psi_1)\Big)$	6.10
negative ERDF-Formel $\neg\psi_1$ mit $\psi_1 \in Fm_{ERDF}$	$ERDFtoCL^{+}(\psi_1)$	6.12

Tabelle 6.1: Definition der Abbildung $ERDFtoCL^{-}$

ERDF-Syntaxkategorie Aus	**Definition von** $ERDFtoCL^+(Aus)$	**Nr.**
RDF URI-Referenz $<aaa>$	interpretierbarer CL-Name aaa	6.3
QName $bb:aaa$	interpretierbarer CL-Name $bb:aaa$	6.3
leerer RDF-Knoten xxx	interpretierbarer CL-Name xxx	6.3
ungetyptes RDF-Literal ohne Sprachkennzeichnung `"s"`	quoted String `'s'`	6.3
ungetyptes RDF-Literal mit Sprachkennzeichnung `"s"@t`	CL-Funktionsterm (`CLungeLit 's' 't'`)	6.3
getyptes RDF-Literal $"aaa"\hat{}\hat{}ddd$	CL-Funktionsterm (`ddd` *'aaa'*)	6.3
positives ERDF-Tripel $p(s,o)$	CL-Atomformel (`ERDF_TRIPEL+` $ERDFtoCL^+(p)$ $ERDFtoCL^+(s)$ $ERDFtoCL^+(o)$)	6.5
positive ERDF-Formel $\sim\psi_1$ mit $\psi_1 \in Fm_{ERDF}$	Boolesche CL-Formel $(\text{not } ERDFtoCL^+(\psi_1))$	6.9
positive ERDF-Formel $(\psi_1 \wedge \psi_2)$ mit $\psi_1, \psi_2 \in Fm_{ERDF}$	Boolesche CL-Formel $(\text{and } ERDFtoCL^+(\psi_1)\ ERDFtoCL^+(\psi_2))$	6.9
positive ERDF-Formel $(\psi_1 \vee \psi_2)$ mit $\psi_1, \psi_2 \in Fm_{ERDF}$	Boolesche CL-Formel $(\text{or } ERDFtoCL^+(\psi_1)\ ERDFtoCL^+(\psi_2))$	6.9
positive ERDF-Formel $(\psi_1 \rightarrow \psi_2)$ mit $\psi_1, \psi_2 \in Fm_{ERDF}$	Boolesche CL-Formel $(\text{if } ERDFtoCL^+(\psi_1)\ ERDFtoCL^+(\psi_2))$	6.9
positive ERDF-Formel $\exists x\psi_1$ mit $\psi_1 \in Fm_{ERDF}$	quantifizierte CL-Formel $(\text{exists } (ERDFtoCL^+(x))\ ERDFtoCL^+(\psi_1))$	6.9
positive ERDF-Formel $\forall x\psi_1$ mit $\psi_1 \in Fm_{ERDF}$	quantifizierte CL-Formel $(\text{forall } (ERDFtoCL^+(x))\ ERDFtoCL^+(\psi_1))$	6.9
negative ERDF-Formel $\neg\psi_1$ mit $\psi_1 \in Fm_{ERDF}$	$ERDFtoCL^-(\psi_1)$	6.11

Tabelle 6.2: Definition der Abbildung $ERDFtoCL^+$ – Teil 1

ERDF-Syntaxkategorie Aus	**Definition von** $ERDFtoCL^+(Aus)$	**Nr.**
ERDF-Graph $G = \{t_1, t_2, \ldots, t_n\}$ mit $Var_{RDF}(G) = \{x_1, \ldots, x_m\}$	quantifizierte CL-Formel $\Big($exists$($ $ERDFtoCL^+(x_1)$ $\ldots$ $ERDFtoCL^+(x_m))$ (and $ERDFtoCL^+(t_1)$ $\ldots$ $ERDFtoCL^+(t_n))\Big)$	6.13
$true$	Boolesche CL-Formel (and)	6.14
$false$	Boolesche CL-Formel (or)	6.14
ERDF-Regel $r = \psi_1 \Leftarrow \psi_2$ mit $Var_{RDF}(r) = \{x_1, \ldots, x_n\}$	quantifizierte CL-Formel $\Big($forall$($ $ERDFtoCL^+(x_1)$ $\ldots$ $ERDFtoCL^+(x_n))$ (if $ERDFtoCL^+(\psi_2)$ $ERDFtoCL^+(\psi_1))\Big)$	6.15
ERDF-Programm $P = \{r_1, r_2, \ldots, r_n\}$	Boolesche CL-Formel $\Big($and $ERDFtoCL^+(r_1)$ $\ldots$ $ERDFtoCL^+(r_n)\Big)$	6.16
ERDF-Ontologie $O = \langle G, P \rangle$	Boolesche CL-Formel $\Big($and $ERDFtoCL^+(G)\, ERDFtoCL^+(P)\Big)$	6.17

Tabelle 6.3: Definition der Abbildung $ERDFtoCL^+$ – Teil 2

$$
\begin{aligned}
Ax_2' = (&\textsf{forall}\,(P\ S\ O)\ (\textsf{iff}\\
&\quad (\texttt{ERDF\_TRIPEL-}\ P\ S\ O)\\
&\quad (\textsf{and}\\
&\qquad ((\texttt{neg}\ P)\ S\ O)\\
&\qquad ((\texttt{pos}\ \texttt{rdf:type})\ P\ \texttt{rdf:Property})\\
&\quad)\\
&))
\end{aligned}
\tag{6.19}
$$

In Übereinstimmung mit RDFS werden in ERDF die Klassenzugehörigkeiten ebenfalls mithilfe der RDF URI-Referenz `rdf:type` modelliert. Bei jeder Klasse in ERDF wird allerdings zwischen der positiven und negativen Klassenextension unterschieden. Für die positive Extension gilt nach der dritten Bedingung von Definition 6.17 das folgende Axiom:

$$
\begin{aligned}
Ax_3 = (&\textsf{forall}\,(X\ Y)\ (\textsf{iff}\\
&\quad ((\texttt{pos}\ \texttt{rdf:type})\ X\ Y)\\
&\quad (\textsf{and}\\
&\qquad ((\texttt{pos}\ Y)\ X)\\
&\qquad ((\texttt{pos}\ \texttt{rdf:type})\ Y\ \texttt{rdfs:Class})\\
&\quad)\\
&))
\end{aligned}
\tag{6.20}
$$

In ähnlicher Weise charakterisieren wir die negative Extension von `rdf:type`.

$$
\begin{aligned}
Ax_4 = (&\textsf{forall}\,(X\ Y)\ (\textsf{iff}\\
&\quad ((\texttt{neg}\ \texttt{rdf:type})\ X\ Y)\\
&\quad (\textsf{and}\\
&\qquad ((\texttt{neg}\ Y)\ X)\\
&\qquad ((\texttt{pos}\ \texttt{rdf:type})\ Y\ \texttt{rdfs:Class})\\
&\quad)\\
&))
\end{aligned}
\tag{6.21}
$$

Nach Definition der ERDF-Interpretationen sowie Beobachtung 6.22 beinhaltet die Menge *Res* alle Elemente des ERDF-Universums. Dieser Sachverhalt ist unter Verwendung der RDF URI-Referenz `rdfs:Resource` folgendermaßen in Common Logic formulierbar:

$$
Ax_5 = (\textsf{forall}\,(x)\ ((\texttt{pos}\ \texttt{rdf:type})\ x\ \texttt{rdfs:Resource})) \tag{6.22}
$$

Die zehnte Bedingung von Definition 6.17 stellen wir mithilfe des Axioms Ax_6 sicher.

$$
\begin{aligned}
Ax_6 = (&\textsf{forall}\,(X\ Y\ u\ v)\ (\textsf{if}\\
&\quad (\textsf{and}\\
&\qquad (\texttt{ERDF\_TRIPEL+}\ \texttt{rdfs:domain}\ X\ Y)\\
&\qquad (\texttt{ERDF\_TRIPEL+}\ X\ u\ v)\\
&\quad)\\
&\quad ((\texttt{pos}\ \texttt{rdf:type})\ u\ Y)\\
&))
\end{aligned}
\tag{6.23}
$$

Die Semantik der RDF URI-Referenz `rdfs:range` wird in ähnlicher Weise unter Verwendung des nachfolgenden Axioms garantiert.

$$
\begin{aligned}
Ax_7 = (&\textsf{forall}\,(X\ Y\ u\ v)\ (\textsf{if}\\
&\quad (\textsf{and}\\
&\qquad (\texttt{ERDF\_TRIPEL+}\ \texttt{rdfs:range}\ X\ Y)\\
&\qquad (\texttt{ERDF\_TRIPEL+}\ X\ u\ v)\\
&\quad)\\
&\quad ((\texttt{pos}\ \texttt{rdf:type})\ v\ Y)\\
&))
\end{aligned}
\tag{6.24}
$$

Die zwölfte Bedingung der ERDF-Interpretationen besagt, dass jede Klasse des ERDF-Universums eine Teilmenge von *Res* ist. Diese Forderung wird in Common Logic wie folgt umgesetzt:

$$
\begin{aligned}
Ax_8 = (&\textsf{forall}\,(x)\ (\textsf{if}\\
&\quad ((\texttt{pos}\ \texttt{rdf:type})\ x\ \texttt{rdfs:Class})\\
&\quad (\texttt{ERDF\_TRIPEL+}\ \texttt{rdfs:subClassOf}\ x\ \texttt{rdfs:Resource})\\
&))
\end{aligned}
\tag{6.25}
$$

Als nächstes axiomatisieren wir die Semantik der Property `rdfs:subClassOf`. Dabei sind nicht nur die positiven Extensionen der Klassen von Bedeutung, son-

dern auch die negativen Klassenextensionen.

$$
\begin{aligned}
Ax_9 = (&\textsf{forall}\ (X\ Y)\ (\textsf{if}\\
&\quad (\texttt{ERDF\_TRIPEL+}\ \texttt{rdfs:subClassOf}\ X\ Y)\\
&\quad (\textsf{and}\\
&\qquad (\textsf{forall}\ (a)\ (\textsf{if}\\
&\qquad\quad ((\texttt{pos}\ \texttt{rdf:type})\ a\ X)\\
&\qquad\quad ((\texttt{pos}\ \texttt{rdf:type})\ a\ Y)\\
&\qquad))\\
&\qquad (\textsf{forall}\ (a)\ (\textsf{if}\\
&\qquad\quad ((\texttt{neg}\ \texttt{rdf:type})\ a\ Y)\\
&\qquad\quad ((\texttt{neg}\ \texttt{rdf:type})\ a\ X)\\
&\qquad))\\
&\quad)\\
&))
\end{aligned}
\tag{6.26}
$$

Es ist nicht notwendig die beiden Axiome $((\texttt{pos}\ \texttt{rdf:type})\ X\ \texttt{rdfs:Class})$ und $((\texttt{pos}\ \texttt{rdf:type})\ Y\ \texttt{rdfs:Class})$ für Common Logic explizit zu fordern, denn diese folgen unter Anwendung der Axiome Ax_6 und Ax_7 sowie der beiden CL-Formeln $(\texttt{ERDF\_TRIPEL+}\ \texttt{rdfs:domain}\ \texttt{rdfs:subClassOf}\ \texttt{rdfs:Class})$ und $(\texttt{ERDF\_TRIPEL+}\ \texttt{rdfs:range}\ \texttt{rdfs:subClassOf}\ \texttt{rdfs:Class})$, welche Teil der eingebetteten axiomatischen RDFS-Tripel sind, direkt aus dem Axiom Ax_9.

Weiterhin müssen wir die Reflexivität und Transitivität der positiven Extension von `rdfs:subClassOf` sicherstellen. Das Axiom Ax_{10} garantiert die Reflexivität.

$$
\begin{aligned}
Ax_{10} = (&\textsf{forall}\ (x)\ (\textsf{if}\\
&\quad ((\texttt{pos}\ \texttt{rdf:type})\ x\ \texttt{rdfs:Class})\\
&\quad (\texttt{ERDF\_TRIPEL+}\ \texttt{rdfs:subClassOf}\ x\ x)\\
&))
\end{aligned}
\tag{6.27}
$$

Die Transitivität wird aufgrund der folgenden CL-Formel gewährleistet:

$$
\begin{aligned}
Ax_{11} = (&\textsf{forall}\ (x\ y\ z)\ (\textsf{if}\\
&\quad (\textsf{and}\\
&\qquad (\texttt{ERDF\_TRIPEL+}\ \texttt{rdfs:subClassOf}\ x\ y)\\
&\qquad (\texttt{ERDF\_TRIPEL+}\ \texttt{rdfs:subClassOf}\ y\ z)\\
&\quad)\\
&\quad (\texttt{ERDF\_TRIPEL+}\ \texttt{rdfs:subClassOf}\ x\ z)\\
&))
\end{aligned}
\tag{6.28}
$$

Nun werden wir die Eigenschaften der RDF URI-Referenz `rdfs:subPropertyOf` modellieren. Auch diese Property hat Auswirkungen auf die positive und negative Extension ihrer Argumente. Das Axiom Ax_{12} realisiert die fünfzehnte Bedingung von Definition 6.17.

$$
\begin{aligned}
Ax_{12} = (\text{forall}\ (X\ Y)\ (\text{if}\ & \\
& (\texttt{ERDF\_TRIPEL+}\ \texttt{rdfs:subPropertyOf}\ X\ Y) \\
& (\text{and} \\
& \quad (\text{forall}\ (a\ b)\ (\text{if} \\
& \qquad (\texttt{ERDF\_TRIPEL+}\ X\ a\ b) \\
& \qquad (\texttt{ERDF\_TRIPEL+}\ Y\ a\ b) \\
& \quad)) \\
& \quad (\text{forall}\ (a\ b)\ (\text{if} \\
& \qquad (\texttt{ERDF\_TRIPEL-}\ Y\ a\ b) \\
& \qquad (\texttt{ERDF\_TRIPEL-}\ X\ a\ b) \\
& \quad)) \\
&) \\
&))
\end{aligned}
\tag{6.29}
$$

Es ist anzumerken, dass die CL-Formeln $((\texttt{pos rdf:type})\ X\ \texttt{rdf:Property})$ und $((\texttt{pos rdf:type})\ Y\ \texttt{rdf:Property})$ aufgrund der axiomatischen RDFS-Tripel sowie der beiden Axiome Ax_6 und Ax_7 aus der obigen CL-Formel folgen. Daher ist die Formulierung dieser Axiome in Common Logic nicht erforderlich.

Nach Definition 6.17 ist die positive Extension von `rdfs:subPropertyOf` reflexiv und transitiv auf der Menge aller Propertys.
Die Reflexivität stellen wir mithilfe der CL-Formel Ax_{13} sicher.

$$
\begin{aligned}
Ax_{13} = (\text{forall}\ (x)\ (\text{if}\ & \\
& ((\texttt{pos rdf:type})\ x\ \texttt{rdf:Property}) \\
& (\texttt{ERDF\_TRIPEL+}\ \texttt{rdfs:subPropertyOf}\ x\ x) \\
&))
\end{aligned}
\tag{6.30}
$$

Das nachfolgend angegebene Axiom Ax_{14} gewährleistet die Transitivität.

$$\begin{aligned} Ax_{14} = (&\text{forall } (x\ y\ z)\ (\text{if} \\ &\quad (\text{and} \\ &\qquad (\texttt{ERDF\_TRIPEL+}\ \texttt{rdfs:subPropertyOf}\ x\ y) \\ &\qquad (\texttt{ERDF\_TRIPEL+}\ \texttt{rdfs:subPropertyOf}\ y\ z) \\ &\quad) \\ &\quad (\texttt{ERDF\_TRIPEL+}\ \texttt{rdfs:subPropertyOf}\ x\ z) \\ &))) \end{aligned} \tag{6.31}$$

Die siebzehnte Bedingung der ERDF-Interpretationen formulieren wir in Common Logic wie folgt:

$$\begin{aligned} Ax_{15} = (&\text{forall } (x)\ (\text{if} \\ &\quad ((\texttt{pos}\ \texttt{rdf:type})\ x\ \texttt{rdfs:Datatype}) \\ &\quad (\texttt{ERDF\_TRIPEL+}\ \texttt{rdfs:subClassOf}\ x\ \texttt{rdfs:Literal}) \\ &)) \end{aligned} \tag{6.32}$$

Unter Verwendung von Axiom Ax_{16} stellen wir die Semantik der RDF URI-Referenz `rdfs:ContainerMembershipProperty` sicher. Es gilt:

$$\begin{aligned} Ax_{16} = (&\text{forall } (x)\ (\text{if} \\ &\quad ((\texttt{pos}\ \texttt{rdf:type})\ x\ \texttt{rdfs:ContainerMembershipProperty}) \\ &\quad (\texttt{ERDF\_TRIPEL+}\ \texttt{rdfs:subPropertyOf}\ x\ \texttt{rdfs:member}) \\ &)) \end{aligned} \tag{6.33}$$

Als nächstes legen wir die Semantik der totalen Klassen fest. Auch diese Bedingung ist mithilfe von eingebetteten ERDF-Tripeln in Common Logic modellierbar.

$$\begin{aligned} Ax_{17} = (&\text{forall } (X)\ (\text{if} \\ &\quad ((\texttt{pos}\ \texttt{rdf:type})\ X\ \texttt{erdf:TotalClass}) \\ &\quad (\text{forall } (a)\ (\text{or} \\ &\qquad ((\texttt{pos}\ \texttt{rdf:type})\ a\ X) \\ &\qquad ((\texttt{neg}\ \texttt{rdf:type})\ a\ X) \\ &\quad)) \\ &)) \end{aligned} \tag{6.34}$$

In ähnlicher Weise formulieren wir die in Definition 6.17 geforderte Bedingung

für die Menge der totalen Propertys.

$$\begin{aligned} Ax_{18} = (&\textsf{forall}\,(X)\,(\textsf{if} \\ &((\texttt{pos rdf:type})\ X\ \texttt{erdf:TotalProperty}) \\ &(\textsf{forall}\,(a\ b)\,(\textsf{or} \\ &\qquad (\texttt{ERDF\_TRIPEL+}\ X\ a\ b) \\ &\qquad (\texttt{ERDF\_TRIPEL-}\ X\ a\ b) \\ &))) \\ &)) \end{aligned} \tag{6.35}$$

Der nächste Schritt besteht darin, die Anforderungen für XML-Literale in Common Logic umzusetzen.
Sei s ein wohlgeformtes XML-Literal und $\texttt{'}s\texttt{'} \in N$, dann gilt:

$$Ax_{19}^{s} = (\texttt{ERDF\_TRIPEL+ rdf:type}\ (\texttt{rdf:XMLLiteral}\texttt{'}s\texttt{'})\ \texttt{rdf:XMLLiteral}) \tag{6.36}$$

Sei s ein nicht wohlgeformtes XML-Literal und $\texttt{'}s\texttt{'} \in N$, dann gilt:

$$Ax_{20}^{s} = (\texttt{ERDF\_TRIPEL- rdf:type}\ (\texttt{rdf:XMLLiteral}\texttt{'}s\texttt{'})\ \texttt{rdfs:Literal}) \tag{6.37}$$

Abschließend müssen wir die Gültigkeit der axiomatischen RDF-Tripel, RDFS-Tripel und ERDF-Tripel sicherstellen. Da diese Tripelmengen ausschließlich RDF URI-Referenzen umfassen, können sie mithilfe der Abbildung $ERDFtoCL^{+}$ problemlos nach Common Logic übertragen werden.

6.5.3 Einbettung der kohärenten ERDF-Interpretationen

Die kohärenten ERDF-Interpretationen erfüllen zusätzlich zu den Bedingungen von Definition 6.17 die Kohärenzeigenschaft. Diese Besonderheit der kohärenten ERDF-Interpretationen garantieren wir mithilfe der folgenden CL-Formel:

$$\begin{aligned} Ax_{21} = (&\textsf{forall}\,(P)\,(\textsf{if} \\ &((\texttt{pos rdf:type})\ P\ \texttt{rdf:Property}) \\ &(\textsf{not}\,(\textsf{exists}\,(a\ b)\,(\textsf{and} \\ &\qquad (\texttt{ERDF\_TRIPEL+}\ P\ a\ b) \\ &\qquad (\texttt{ERDF\_TRIPEL-}\ P\ a\ b) \\ &))) \\ &)) \end{aligned} \tag{6.38}$$

Das Axiom Ax_{21} hat große Auswirkungen auf das CL-Universum. So lässt sich Lemma 6.19 in Common Logic auf äquivalente Art und Weise wie folgt formulieren:

Lemma 6.24
Es sei $V := V_{RDF} \cup V_{RDFS} \cup V_{ERDF}$ *die Menge der RDF URI-Referenzen des RDF-Vokabulars, des RDFS-Vokabulars und des ERDF-Vokabulars. Des Weiteren bezeichne* $AX = \{Ax_1', Ax_2', Ax_3, \ldots, Ax_{21}\}$ *die Menge der CL-Axiome.*
Sei nun $I \in \mathcal{IB}_{CL}(V_{CL}, IFm)$ *eine CL-Interpretation über* V_{CL} *und IFm mit* $V \subseteq V_{CL}$ *und* $I = \langle \langle UR_I, UD_I, rel_I, fun_I \rangle, int_I, seq_I, irr_I \rangle$.
Falls $I \models AX$, *dann gilt* $rel_I\,(int_I^*\,((\texttt{pos}\;K))) \cap rel_I\,(int_I^*\,((\texttt{neg}\;K))) = \emptyset$ *für alle* $K \in V_{CL}$ *mit* $int_I^*\,(K) \in rel_I\,(int_I^*\,((\texttt{pos rdfs:Class})))$.

Beweis:
Aufgrund der CL-Formel (`ERDF_TRIPEL+ rdf:type rdf:type rdf:Property`), welche wir durch Einbettung der axiomatischen RDF-Tripel erhalten, gilt nach Ax_1'
$I \models ((\texttt{pos rdf:type})\,\texttt{rdf:type rdf:Property})$. $(*)$
Angenommen es existiert ein $K \in V_{CL}$ mit $int_I^*\,(K) \in rel_I\,(int_I^*\,((\texttt{pos rdfs:Class})))$ und $rel_I\,(int_I^*\,((\texttt{pos}\;K))) \cap rel_I\,(int_I^*\,((\texttt{neg}\;K))) \neq \emptyset$.
Dann existiert ein $\widehat{z} \in UD$ mit $\widehat{z} \in rel_I\,(int_I^*\,((\texttt{pos}\;K))) \cap rel_I\,(int_I^*\,((\texttt{neg}\;K)))$.
Es sei nun $z \in N$ ein interpretierbarer CL-Name mit $int_I\,(z) = \widehat{z}$, dann gilt also $I \models ((\texttt{pos}\;K)\,z)$ und $I \models ((\texttt{neg}\;K)\,z)$.
Da $int_I^*\,(K) \in rel_I\,(int_I^*\,((\texttt{pos rdfs:Class})))$ folgt aufgrund der beiden Axiome Ax_3 und Ax_4: $I \models ((\texttt{pos rdf:type})\,z\;K)$ und $I \models ((\texttt{neg rdf:type})\,z\;K)$.
Wegen $(*)$ gilt daher unter Verwendung von Ax_1' und Ax_2' auch:
$I \models (\texttt{ERDF\_TRIPEL+ rdf:type}\;z\;K)$ und $I \models (\texttt{ERDF\_TRIPEL- rdf:type}\;z\;K)$.
Dies ist ein Widerspruch zu Axiom Ax_{21}.
Also gilt $rel_I\,(int_I^*\,((\texttt{pos}\;K))) \cap rel_I\,(int_I^*\,((\texttt{neg}\;K))) = \emptyset$
für alle $K \in V_{CL}$ mit $int_I^*\,(K) \in rel_I\,(int_I^*\,((\texttt{pos rdfs:Class})))$.

■

Nach Lemma 6.24 garantiert das Axiom Ax_{21} nicht nur die Kohärenz der Propertys, sondern es stellt diese Eigenschaft auch für Klassen sicher. Dies bedeutet auch, dass die negative Klassenextension der RDF URI-Referenz `rdfs:Resource` infolge der CL-Formel (6.22) keine einelementigen Tupel beinhaltet. Das heißt die beiden Axiome Ax_5 und Ax_{21} implizieren die folgende CL-Formel:

$$(\text{not}\,(\text{exists}\,(x)\,((\texttt{neg rdf:type})\;x\;\texttt{rdfs:Resource}))) \tag{6.39}$$

Da die Stelligkeit der Relationssymbole in Common Logic variabel ist, folgt aus der CL-Formel (6.38) keine generelle Disjunktheit der positiven und negativen Extension eines CL-Namens. Diese viel strengere Eigenschaft ist in Common Logic unter Verwendung eines CL-Sequenznamens $seq \in SEQ$ wie folgt formulierbar:

$$\begin{aligned}(\text{forall}\,(R)\,(\text{not}\,(\text{exists}\,(seq)\,(\text{and}\\ &((\texttt{pos}\;R)\;seq)\\ &((\texttt{neg}\;R)\;seq)\\ &))))\end{aligned} \tag{6.40}$$

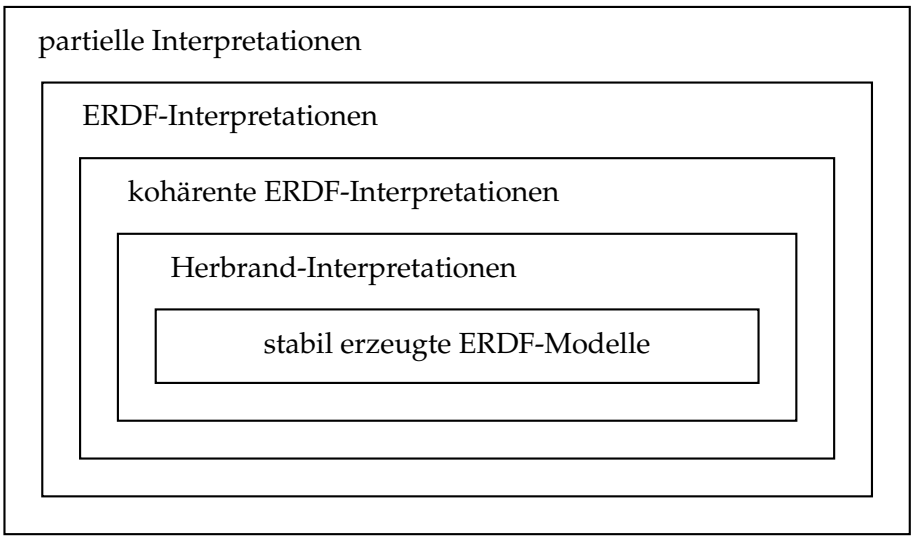

Abbildung 6.5: Verfeinerung der Interpretationen von ERDF

Abschließend sei angemerkt, dass dieser Sachverhalt in gleicher Weise auf die Eigenschaft der Totalität zutreffend ist. Die RDF URI-Referenzen `erdf:TotalClass` und `erdf:TotalProperty` sind in Common Logic nicht ausreichend um totale Relationen zu definieren, da diese beiden Klassenbezeichner lediglich Bedingungen für einstellige und zweistellige Relationen spezifizieren.

6.6 Herbrand-Interpretationen und stabil erzeugte ERDF-Modelle

Für die Definition der modelltheoretischen Semantik von logischen Programmen werden üblicherweise nicht alle Interpretationen betrachtet, sondern ausschließlich die sogenannten *Herbrand-Interpretationen*. Dieser Ansatz wird bei der Festlegung der intuitiven Modelle von ERDF ebenfalls angewendet. Da jedoch nicht jedes Herbrand-Modell einer ERDF-Ontologie als erwünscht gilt, wird die Menge der zulässigen Herbrand-Modelle weiter eingeschränkt. Dies führt zu den *stabil erzeugten ERDF-Modellen*. In Abbildung 6.5 ist diese Verfeinerung der modelltheoretischen Semantik für ERDF graphisch dargestellt. Wir führen nun die Herbrand-Interpretationen und daran anschließend die stabil erzeugten ERDF-Modelle formal ein.

6.6.1 Herbrand-Interpretationen

Die Definition der Herbrand-Interpretationen erfordert zunächst die Festlegung weiterer Begriffe. Als erstes führen wir die Skolemform eines ERDF-Graphen ein.

Definition 6.25 (Skolemfunktion und Skolemform eines ERDF-Graphen)
Sei $G \in Gr_{ERDF}(V_R, Var_{RDF})$ ein ERDF-Graph über V_R und Var_{RDF}.

(a) Die *Skolemfunktion* von G ist eine Abbildung $sk_G : Var_{RDF}(G) \rightarrow URI_{RDF}$, die jedem leeren RDF-Knoten von G eine RDF URI-Referenz zuordnet. Um die Eindeutigkeit dieser URI-Referenzen sicherzustellen, definieren wir sk_G wie folgt: $sk_G(x) = G{:}x$ für alle $x \in Var_{RDF}(G)$.

(b) Die *Skolemform* von G, welche wir mit $sk(G)$ bezeichnen, ist der grundinstanziierte ERDF-Graph, den man erhält, indem alle leeren RDF-Knoten $x \in Var_{RDF}(G)$ in G durch $sk_G(x)$ ersetzt werden.

Die Menge der RDF URI-Referenzen $sk_G(Var_{RDF}(G))$ nennen wir *Skolemvokabular* von G.

□

Also beinhaltet ein ERDF-Graph in Skolemform nach obiger Definition keine leeren RDF-Knoten, sondern lediglich RDF URI-Referenzen und RDF-Literale.
Der nächste Schritt legt das Vokabular einer ERDF-Ontologie fest.

Definition 6.26 (Vokabular einer ERDF-Ontologie)
Sei $O = \langle G, P \rangle$ eine ERDF-Ontologie über V_R und Var_{RDF}. Das *Vokabular* V_O der ERDF-Ontologie O ist definiert als: $V_O = V_R(sk(G)) \cup V_P \cup V_{RDF} \cup V_{RDFS} \cup V_{ERDF}$.
Die Menge V_P bezeichne dabei die Menge der RDF URI-Referenzen und RDF-Literale von P.

□

Nun sind wir in der Lage die Herbrand-Interpretationen formal einzuführen. Für die ERDF-Ontologie O bezeichne die Menge Res_O^H indes die Vereinigung von V_O ohne die wohlgeformten XML-Literale mit der Menge der XML-Werte der wohlgeformten XML-Literale aus V_O.

Definition 6.27 (Herbrand-Interpretation einer ERDF-Ontologie)
Es sei $O = \langle G, P \rangle$ eine ERDF-Ontologie über V_R und Var_{RDF}. Des Weiteren sei $I \in \mathcal{IB}^{ko}_{ERDF}(V_R, Var_{RDF})$ eine kohärente ERDF-Interpretation über V_R und Var_{RDF} mit $I = \langle \mathcal{S}, intU^P, intL^P, intV^P \rangle$ und der dazugehörigen semantischen ERDF-Struktur $\mathcal{S} = \langle \langle Res, Prop, LV, rel^+, rel^- \rangle, Kl, TKl, TProp, Kext^+, Kext^- \rangle$.
Die Interpretation I ist eine *Herbrand-Interpretation* von O, falls die folgenden Bedingungen erfüllt sind:

1. $Res = Res_O^H$
2. $intU^P(u) = u$ für alle RDF URI-Referenzen $u \in V_O$.
3. $intL^P(l) = l$ für alle getypten RDF-Literale $l \in V_O$ die keine wohlgeformten XML-Literale sind.

4. $intL^P(l)$ ist der XML-Wert von l, falls l ein wohlgeformtes XML-Literal aus V_O ist.

Die Menge aller Herbrand-Interpretationen von O bezeichnen wir mit $\mathcal{IB}^H_{ERDF}(O)$.
□

Definition 6.28 (Herbrand-Modell einer ERDF-Ontologie)
Sei $O = \langle G, P \rangle$ eine ERDF-Ontologie über V_R und Var_{RDF}.
Eine Herbrand-Interpretation $I \in \mathcal{IB}^H_{ERDF}(O)$ von O ist ein *Herbrand-Modell* von O genau dann, wenn $I \models \langle sk(G), P \rangle$.
Für die Menge aller Herbrand-Modelle von O vereinbaren wir die Bezeichnung $Mod^H_{ERDF}(O)$.
□

Nach Definition 6.27 gilt für eine Herbrand-Interpretation $I \in \mathcal{IB}^H_{ERDF}(O)$ von O stets $int^*(x) = x$ für alle $x \in V_O$ außer die wohlgeformten XML-Literale. Weiterhin wollen wir anmerken, dass die Menge Res^H_O für eine ERDF-Ontologie O stets unendlich ist, da $\{\texttt{rdf:\_i} \mid \texttt{i} \geq 1\} \subseteq V_{RDF} \subseteq V_O$. Aufgrund dieser Unendlichkeit ist sowohl das Erfüllbarkeitsproblem[21] als auch das Folgerungsproblem[22] der stabil erzeugten ERDF-Modellsemantik, welche wir in Abschnitt 6.6.2 einführen, unentscheidbar [6].

In [5] wird darauf hingewiesen, dass jede Herbrand-Interpretation einer ERDF-Ontologie O eindeutig durch die Menge der Propertys sowie die Abbildungen rel^+, rel^- und $intV^P$ festgelegt wird. Diesen Sachverhalt werden wir nun mithilfe des folgenden Lemmas beweisen.

Lemma 6.29
Es sei $O = \langle G, P \rangle$ eine ERDF-Ontologie über V_R und Var_{RDF}. Weiterhin seien $I, K \in \mathcal{IB}^H_{ERDF}(O)$ zwei Herbrand-Interpretationen von O mit $I = \langle \mathcal{S}_I, intU^P_I, intL^P_I, intV^P_I \rangle$ und $K = \langle \mathcal{S}_K, intU^P_K, intL^P_K, intV^P_K \rangle$. Die dazugehörigen semantischen ERDF-Strukturen seien definiert als $\mathcal{S}_I = \langle \langle Res_I, Prop_I, LV_I, rel^+_I, rel^-_I \rangle, Kl_I, TKl_I, TProp_I, Kext^+_I, Kext^-_I \rangle$ und $\mathcal{S}_K = \langle \langle Res_K, Prop_K, LV_K, rel^+_K, rel^-_K \rangle, Kl_K, TKl_K, TProp_K, Kext^+_K, Kext^-_K \rangle$.
Falls $Prop_I = Prop_K$, $rel^+_I(p) = rel^+_K(p)$ sowie $rel^-_I(p) = rel^-_K(p)$ für alle $p \in Prop_I$ und $intV^P_I(x) = intV^P_K(x)$ für alle $x \in Var_{RDF}$ gilt, dann sind die beiden Herbrand-Interpretationen I und K identisch.

Beweis:
Wir beweisen die Identität der beiden Herbrand-Interpretationen I und K, indem wir zeigen, dass die einzelnen Komponenten von I und K jeweils identisch sind.

[21] Das Erfüllbarkeitsproblem der stabil erzeugten ERDF-Modellsemantik ist ein Entscheidungsproblem mit der Fragestellung, ob eine gegebene ERDF-Ontologie O ein stabil erzeugtes ERDF-Modell besitzt.

[22] Auch dieses Problem ist ein Entscheidungsproblem. Es wird gefragt, ob die ERDF-Formel φ aus der ERDF-Ontologie O logisch folgt, also ob $O \models_{st} \varphi$ gilt. Dabei bezeichnet $\models_{st}$ die Folgerungsrelation, welche durch die stabil erzeugten ERDF-Modelle induziert wird.

Als erstes belegen wir die Behauptung für die Interpretationsabbildungen $intU^P$, $intL^P$ und $intV^P$.
1. Behauptung: $intU_I^P(u) = intU_K^P(u)$ für alle RDF URI-Referenzen $u \in V_O$
Es gilt $intU_I^P(u) = u = intU_K^P(u)$ für alle RDF URI-Referenzen $u \in V_O$ nach Definition 6.27.
2. Behauptung: $intL_I^P(l) = intL_K^P(l)$ für alle getypten RDF-Literale $l \in V_O$
1. Fall: Sei l ein getyptes RDF-Literal welches kein wohlgeformtes XML-Literal ist mit $l \in V_O$. Dann gilt nach Definition 6.27 $intL_I^P(l) = l = intL_K^P(l)$.
2. Fall: Sei l ein wohlgeformtes XML-Literal mit $l \in V_O$.
Dann gilt nach Definition 6.27 $intL_I^P(l) = XML(l) = intL_K^P(l)$, wobei $XML(l)$ den XML-Wert von l bezeichnet.
Damit gilt $intL_I^P(l) = intL_K^P(l)$ für alle getypten RDF-Literale $l \in V_O$.
3. Behauptung: $intV_I^P(x) = intV_K^P(x)$ für alle $x \in Var_{RDF}$
Dies gilt nach Voraussetzung.
Als nächstes folgen die Komponenten der semantischen P-Strukturen ohne die Mengen LV_I und LV_K.
4. Behauptung: $Res_I = Res_K$
Es gilt $Res_I = Res_O^H = Res_K$ nach Definition 6.27.
5. Behauptung: $Prop_I = Prop_K$
Dies gilt nach Voraussetzung.
6. Behauptung: $rel_I^+(p) = rel_K^+(p)$ für alle $p \in Prop_I$
Dies gilt nach Voraussetzung.
7. Behauptung: $rel_I^-(p) = rel_K^-(p)$ für alle $p \in Prop_I$
Dies gilt nach Voraussetzung.
Nun folgen die restlichen Elemente der Herbrand-Interpretationen.
8. Behauptung: $Kext_I^+(k) = Kext_K^+(k)$ für alle $k \in Kl_I$
Sei $k \in Kl_I$. Nach Definition 6.17 ist die Menge $Kext_I^+(k)$ wie folgt festgelegt:
$Kext_I^+(k) = \{x \mid \langle x,k\rangle \in rel_I^+(intU_I^P(\texttt{rdf:type}))\}$. Aufgrund der 1. Behauptung gilt $intU_I^P(\texttt{rdf:type}) = intU_K^P(\texttt{rdf:type})$ und damit folgt wegen der 6. Behauptung $rel_I^+(intU_I^P(\texttt{rdf:type})) = rel_K^+(intU_K^P(\texttt{rdf:type}))$. Also
$\{x \mid \langle x,k\rangle \in rel_I^+(intU_I^P(\texttt{rdf:type}))\} = \{x \mid \langle x,k\rangle \in rel_K^+(intU_K^P(\texttt{rdf:type}))\}$
und somit $Kext_I^+(k) = Kext_K^+(k)$.
Damit gilt $Kext_I^+(k) = Kext_K^+(k)$ für alle $k \in Kl_I$.
9. Behauptung: $Kext_I^-(k) = Kext_K^-(k)$ für alle $k \in Kl_I$
Sei $k \in Kl_I$. Nach Definition 6.17 ist die Menge $Kext_I^-(k)$ wie folgt definiert:
$Kext_I^-(k) = \{x \mid \langle x,k\rangle \in rel_I^-(intU_I^P(\texttt{rdf:type}))\}$. Aufgrund der 1. Behauptung gilt $intU_I^P(\texttt{rdf:type}) = intU_K^P(\texttt{rdf:type})$ und damit folgt wegen der 7. Behauptung $rel_I^-(intU_I^P(\texttt{rdf:type})) = rel_K^-(intU_K^P(\texttt{rdf:type}))$. Also
$\{x \mid \langle x,k\rangle \in rel_I^-(intU_I^P(\texttt{rdf:type}))\} = \{x \mid \langle x,k\rangle \in rel_K^-(intU_K^P(\texttt{rdf:type}))\}$
und somit $Kext_I^-(k) = Kext_K^-(k)$.
Damit gilt $Kext_I^-(k) = Kext_K^-(k)$ für alle $k \in Kl_I$.

10. Behauptung: $LV_I = LV_K$
Nach Definition 6.17 gilt $LV_I = Kext_I^+\left(intU_I^P\left(\texttt{rdfs:Literal}\right)\right)$. Des Weiteren gilt aufgrund der 1. Behauptung $intU_I^P\left(\texttt{rdfs:Literal}\right) = intU_K^P\left(\texttt{rdfs:Literal}\right)$ und damit folgt wegen der 8. Behauptung
$Kext_I^+\left(intU_I^P\left(\texttt{rdfs:Literal}\right)\right) = Kext_K^+\left(intU_K^P\left(\texttt{rdfs:Literal}\right)\right)$.
Also $LV_I = Kext_K^+\left(intU_K^P\left(\texttt{rdfs:Literal}\right)\right) = LV_K$.
11. Behauptung: $Kl_I = Kl_K$
Nach Definition 6.17 gilt $Kl_I = Kext_I^+\left(intU_I^P\left(\texttt{rdfs:Class}\right)\right)$. Des Weiteren gilt aufgrund der 1. Behauptung $intU_I^P\left(\texttt{rdfs:Class}\right) = intU_K^P\left(\texttt{rdfs:Class}\right)$ und damit folgt wegen der 8. Behauptung
$Kext_I^+\left(intU_I^P\left(\texttt{rdfs:Class}\right)\right) = Kext_K^+\left(intU_K^P\left(\texttt{rdfs:Class}\right)\right)$.
Also $Kl_I = Kext_K^+\left(intU_K^P\left(\texttt{rdfs:Class}\right)\right) = Kl_K$.
12. Behauptung: $TKl_I = TKl_K$
Nach Definition 6.17 gilt $TKl_I = Kext_I^+\left(intU_I^P\left(\texttt{erdf:TotalClass}\right)\right)$. Weiterhin gilt wegen der 1. Behauptung $intU_I^P\left(\texttt{erdf:TotalClass}\right) = intU_K^P\left(\texttt{erdf:TotalClass}\right)$ und damit folgt aufgrund der 8. Behauptung
$Kext_I^+\left(intU_I^P\left(\texttt{erdf:TotalClass}\right)\right) = Kext_K^+\left(intU_K^P\left(\texttt{erdf:TotalClass}\right)\right)$.
Also $TKl_I = Kext_K^+\left(intU_K^P\left(\texttt{erdf:TotalClass}\right)\right) = TKl_K$.
13. Behauptung: $TProp_I = TProp_K$
Nach Definition 6.17 gilt $TProp_I = Kext_I^+\left(intU_I^P\left(\texttt{erdf:TotalProperty}\right)\right)$. Weiterhin gilt aufgrund der 1. Behauptung
$intU_I^P\left(\texttt{erdf:TotalProperty}\right) = intU_K^P\left(\texttt{erdf:TotalProperty}\right)$ und damit folgt
$Kext_I^+\left(intU_I^P\left(\texttt{erdf:TotalProperty}\right)\right) = Kext_K^+\left(intU_K^P\left(\texttt{erdf:TotalProperty}\right)\right)$.
Also $TProp_I = Kext_K^+\left(intU_K^P\left(\texttt{erdf:TotalProperty}\right)\right) = TProp_K$.
Damit haben wir gezeigt, dass die beiden Herbrand-Interpretationen I und K identisch sind.

■

Um die Identität zweier Herbrand-Interpretationen I und K auszudrücken, vereinbaren wir die Notation $I = K$.

Angesichts der Tatsache, dass die Menge aller Herbrand-Modelle einer ERDF-Ontologie O auch unerwünschte Elemente enthält, schränken wir die Menge der zulässigen Modelle weiter ein. Wir betrachten zunächst ein kleines Beispiel.

Beispiel 6.30
Es seien $p, s, o \in URI_{RDF}$ und $O = \langle G, \emptyset \rangle$ eine ERDF-Ontologie über V_R und $Var_{RDF} = \emptyset$ mit $G = \{p\,(s, o)\}$.
Da G keine Aussage über das ERDF-Tripel $p\,(o, s)$ beinhaltet, existieren Herbrand-Modelle I von O mit $I \models p\,(o, s)$. Dies stimmt jedoch nicht mit der beabsichtigten Semantik überein, denn p ist weder eine totale Property noch kann das ERDF-Tripel $p\,(o, s)$ aus irgendeinem Sachverhalt abgeleitet und damit begründet werden. Also sollte die ERDF-Formel $\sim p\,(o, s)$ in jedem intuitiven Modell von O erfüllt sein.

□

Um derartige Herbrand-Modelle auszuschließen, führen wir die Menge der minimalen Herbrand-Modelle ein. Dazu benötigen wir eine binäre Relation, so dass die einzelnen Herbrand-Interpretationen untereinander geordnet werden können.

Definition 6.31 (Ordnung zwischen Herbrand-Interpretationen)
Es sei $O = \langle G, P \rangle$ eine ERDF-Ontologie über V_R und Var_{RDF}. Des Weiteren seien $I, K \in \mathcal{IB}^H_{ERDF}(O)$ zwei Herbrand-Interpretationen von O.
Die Interpretation K *erweitert* die Interpretation I genau dann, wenn:

1. $Prop_I \subseteq Prop_K$,
2. $rel^+_I(p) \subseteq rel^+_K(p)$ und $rel^-_I(p) \subseteq rel^-_K(p)$ für alle $p \in Prop_I$ und
3. $intV^P_I(x) = intV^P_K(x)$ für alle $x \in Var_{RDF}$.

Falls K die Interpretation I erweitert, dann schreiben wir dafür $I \preceq K$.

□

In [5] wird angemerkt, dass die Relation $\preceq$ eine Halbordnung auf der Menge $\mathcal{IB}^H_{ERDF}(O)$ ist. Auch dieses Resultat wollen wir durch Angabe des entsprechenden Beweises verdeutlichen.

Lemma 6.32
Es sei $O = \langle G, P \rangle$ eine ERDF-Ontologie über V_R und Var_{RDF}.
Die binäre Relation $\preceq$ ist eine Halbordnung auf der Menge $\mathcal{IB}^H_{ERDF}(O)$.

Beweis:
Nach Definition 2.3 ist die Reflexivität, die Transitivität und die Antisymmetrie für die Relation $\preceq$ zu zeigen.
Es seien nun $I_1, I_2, I_3 \in \mathcal{IB}^H_{ERDF}(O)$ Herbrand-Interpretationen von O mit
$I_1 = \left\langle \mathcal{S}_{I_1}, intU^P_{I_1}, intL^P_{I_1}, intV^P_{I_1} \right\rangle$, $I_2 = \left\langle \mathcal{S}_{I_2}, intU^P_{I_2}, intL^P_{I_2}, intV^P_{I_2} \right\rangle$ sowie
$I_3 = \left\langle \mathcal{S}_{I_3}, intU^P_{I_3}, intL^P_{I_3}, intV^P_{I_3} \right\rangle$ und den semantischen ERDF-Strukturen
$\mathcal{S}_{I_1} = \left\langle \left\langle Res_{I_1}, Prop_{I_1}, LV_{I_1}, rel^+_{I_1}, rel^-_{I_1} \right\rangle, Kl_{I_1}, TKl_{I_1}, TProp_{I_1}, Kext^+_{I_1}, Kext^-_{I_1} \right\rangle$,
$\mathcal{S}_{I_2} = \left\langle \left\langle Res_{I_2}, Prop_{I_2}, LV_{I_2}, rel^+_{I_2}, rel^-_{I_2} \right\rangle, Kl_{I_2}, TKl_{I_2}, TProp_{I_2}, Kext^+_{I_2}, Kext^-_{I_2} \right\rangle$ sowie
$\mathcal{S}_{I_3} = \left\langle \left\langle Res_{I_3}, Prop_{I_3}, LV_{I_3}, rel^+_{I_3}, rel^-_{I_3} \right\rangle, Kl_{I_3}, TKl_{I_3}, TProp_{I_3}, Kext^+_{I_3}, Kext^-_{I_3} \right\rangle$.
1. Behauptung: $\preceq$ ist reflexiv
Da $Prop_{I_1} \subseteq Prop_{I_1}$, $intV^P_{I_1}(x) = intV^P_{I_1}(x)$ für alle $x \in Var_{RDF}$ sowie für alle $p \in Prop_{I_1}$ gilt $rel^+_{I_1}(p) \subseteq rel^+_{I_1}(p)$ und $rel^-_{I_1}(p) \subseteq rel^-_{I_1}(p)$, folgt daher $I_1 \preceq I_1$ für alle $I_1 \in \mathcal{IB}^H_{ERDF}(O)$.
2. Behauptung: $\preceq$ ist transitiv
Es sei $I_1 \preceq I_2$ und $I_2 \preceq I_3$. Dann gilt $Prop_{I_1} \subseteq Prop_{I_2}$ und $Prop_{I_2} \subseteq Prop_{I_3}$ nach Voraussetzung. Da die Relation $\subseteq$ transitiv ist, folgt $Prop_{I_1} \subseteq Prop_{I_3}$.

Sei nun $p \in Prop_{I_1}$. Dann gilt $rel^+_{I_1}(p) \subseteq rel^+_{I_2}(p)$ und $rel^+_{I_2}(p) \subseteq rel^+_{I_3}(p)$ nach Voraussetzung. Also folgt aufgrund der Transitivität von $\subseteq$ auch $rel^+_{I_1}(p) \subseteq rel^+_{I_3}(p)$.
Analog gilt $rel^-_{I_1}(p) \subseteq rel^-_{I_2}(p)$ und $rel^-_{I_2}(p) \subseteq rel^-_{I_3}(p)$ nach Voraussetzung und damit $rel^-_{I_1}(p) \subseteq rel^-_{I_3}(p)$.
Da nach Voraussetzung $intV^P_{I_1}(x) = intV^P_{I_2}(x)$ und $intV^P_{I_2}(x) = intV^P_{I_3}(x)$ für alle $x \in Var_{RDF}$ gilt, folgt auch $intV^P_{I_1}(x) = intV^P_{I_3}(x)$ für alle $x \in Var_{RDF}$. Somit haben wir $I_1 \preceq I_3$ gezeigt.
3. Behauptung: $\preceq$ ist antisymmetrisch
Es sei $I_1 \preceq I_2$ und $I_2 \preceq I_1$. Nach Voraussetzung gilt $Prop_{I_1} \subseteq Prop_{I_2}$ und $Prop_{I_2} \subseteq Prop_{I_1}$ und damit $Prop_{I_1} = Prop_{I_2}$. Weiterhin gilt nach Voraussetzung $rel^+_{I_1}(p) \subseteq rel^+_{I_2}(p)$ und $rel^+_{I_2}(p) \subseteq rel^+_{I_1}(p)$ für alle $p \in Prop_{I_1}$. Also $rel^+_{I_1}(p) = rel^+_{I_2}(p)$ für alle $p \in Prop_{I_1}$. In gleicher Weise gilt $rel^-_{I_1}(p) \subseteq rel^-_{I_2}(p)$ und $rel^-_{I_2}(p) \subseteq rel^-_{I_1}(p)$ für alle $p \in Prop_{I_1}$. Also $rel^-_{I_1}(p) = rel^-_{I_2}(p)$ für alle $p \in Prop_{I_1}$.
Wir haben nun $Prop_{I_1} = Prop_{I_2}$ sowie $rel^+_{I_1}(p) = rel^+_{I_2}(p)$ und $rel^-_{I_1}(p) = rel^-_{I_2}(p)$ für alle $p \in Prop_{I_1}$ gezeigt. Da zusätzlich $intV^P_{I_1}(x) = intV^P_{I_2}(x)$ für alle $x \in Var_{RDF}$ nach Voraussetzung gilt, folgt unter Verwendung von Lemma 6.29 die Identität der beiden Herbrand-Interpretationen I_1 und I_2.
Damit haben wir die Reflexivität, Transitivität und Antisymmetrie für $\preceq$ gezeigt. Also ist die Relation $\preceq$ eine Halbordnung auf der Menge $\mathcal{IB}^H_{ERDF}(O)$.

■

Mithilfe der Relation $\preceq$ sind wir nun in der Lage, die minimalen Herbrand-Interpretationen und die minimalen Herbrand-Modelle einer ERDF-Ontologie einzuführen.

Definition 6.33 (minimale Herbrand-Interpretationen einer ERDF-Ontologie)
Es sei $O = \langle G, P \rangle$ eine ERDF-Ontologie über V_R und Var_{RDF}. Weiter sei $\mathcal{I} \subseteq \mathcal{IB}^H_{ERDF}(O)$ eine Menge von Herbrand-Interpretationen.
Die Menge der *minimalen Herbrand-Interpretationen* von $\mathcal{I}$ ist definiert als:
$Min(\mathcal{I}) = \{I \in \mathcal{I} \mid \nexists K \in \mathcal{I} : K \neq I \text{ und } K \preceq I\}$.

□

Als nächstes führen wir die minimalen Herbrand-Modelle einer ERDF-Ontologie formal ein.

Definition 6.34 (minimale Herbrand-Modelle einer ERDF-Ontologie)
Es sei $O = \langle G, P \rangle$ eine ERDF-Ontologie über V_R und Var_{RDF}.
Die Menge der *minimalen Herbrand-Modelle* von O definieren wir wie folgt:
$Mod^{min}_{ERDF}(O) = Min\left(Mod^H_{ERDF}(O)\right)$.

□

Betrachten wir nun das Beispiel 6.30 noch einmal. Dann besitzt die darin beschriebene ERDF-Ontologie O genau ein minimales Herbrand-Modell I mit $I \models p(s,o)$ und $I \models\!\sim p(o,s)$.

Es sei angemerkt, dass die Eindeutigkeit des minimalen Herbrand-Modells auch bei $Var_{RDF} = \emptyset$ nicht immer gegeben ist. Da die Grundmenge einer Herbrand-Interpretation stets unendlich ist, sind sogar unendlich viele minimale Herbrand-Modelle einer ERDF-Ontologie möglich.

Beispiel 6.35
Es sei $p \in URI_{RDF}$ und $O = \langle G, \emptyset \rangle$ eine ERDF-Ontologie über V_R und $Var_{RDF} = \emptyset$ mit $G = \{\texttt{rdf:type}\,(p, \texttt{erdf:TotalProperty})\}$.
Dann ist die Menge $Mod^{min}_{ERDF}(O)$ unendlich.

□

Abschließend wollen wir darauf hinweisen, dass die Herbrand-Interpretationen einer ERDF-Ontologie in Common Logic mithilfe der quoted Strings simuliert werden können. Dazu werden die Abbildungen $ERDFtoCL^+$ und $ERDFtoCL^-$ so modifiziert, dass jede RDF URI-Referenz und jedes getypte RDF-Literal, welches kein XML-Literal ist, auf einen quoted String abgebildet wird. Das heißt:

$$ERDFtoCL^+(x) = ERDFtoCL^-(x) = \begin{cases} \text{'}u\text{'} & \text{falls } u \text{ eine RDF} \\ & \text{URI-Referenz ist} \\ \text{'}l\text{'} & \text{falls } l \text{ ein getyptes RDF-Literal,} \\ & \text{das kein XML-Literal ist} \end{cases} \tag{6.41}$$

Für XML-Literale besteht keine Notwendigkeit die zwei Abbildungen anzupassen, denn jedes XML-Literal `"s"^^rdf:XMLLiteral` wird auf den CL-Funktionsterm (`rdf:XMLLiteral 's'`) abgebildet und die Funktion `rdf:XMLLiteral` ordnet jedem wohlgeformten Argument den entsprechenden XML-Wert zu. Nicht wohlgeformte XML-Literale werden im Gegensatz dazu durch `rdf:XMLLiteral` auf sich selbst abgebildet.
Die minimale Modellsemantik von ERDF ist jedoch in Common Logic unter Verwendung der klassischen Semantik nicht modellierbar.

6.6.2 Stabil erzeugte ERDF-Modelle

Die Menge der minimalen Herbrand-Modelle einer ERDF-Ontologie beinhaltet immer noch unerwünschte Elemente. Die Ursache dafür liegt in der Interpretation der ERDF-Regeln, die mit der Semantik von Implikationen übereinstimmt. Die ERDF-Regeln sind jedoch als Ableitungsregeln zu betrachten, um den konstruktiven Charakter der Erzeugung neuem Wissens aus bekannten Informationen möglichst natürlich wiederzugeben. Die minimale Modellsemantik leistet dies jedoch nicht, wie das nachfolgende Beispiel zeigt.

Beispiel 6.36
Es seien $p, q, s, o \in URI_{RDF}$ und $O = \langle \emptyset, P \rangle$ eine ERDF-Ontologie über V_R und $Var_{RDF} = \emptyset$ mit $P = \{p\,(s, o) \Leftarrow\sim q\,(s, o)\}$. Dann besitzt O genau zwei minimale

Herbrand-Modelle $Mod^{min}_{ERDF}(O) = \{I_1, I_2\}$ mit $I_1 \models\!\sim q(s,o)$ und $I_1 \models p(s,o)$ sowie $I_2 \models q(s,o)$ und $I_2 \models\!\sim p(s,o)$. Da keine ERDF-Regel vorhanden ist, welche die Ableitung von $q(s,o)$ erlaubt und dieses ERDF-Tripel nicht als Faktum gegeben ist, sollte für alle Herbrand-Modelle I von O die Eigenschaft $I \models\!\sim q(s,o)$ erfüllt sein.

□

Eine Modellmenge, welche diesen konstruktiven Charakter berücksichtigt, ist die von Heinrich Herre und Gerd Wagner vorgeschlagene Menge der *stabil erzeugten Modelle* [43]. Diese Semantik wurde bereits auf logische Programme mit zwei Arten von Negation erweitert [2], [24]. Die stabil erzeugten Modelle werden durch Interpretationsketten gebildet, deren aufeinanderfolgende Glieder durch eine zu spezifizierende Relation miteinander in Beziehung stehen. In [5] wurden die stabil erzeugten Modelle auf ERDF-Ontologien übertragen und als Verfeinerung der minimalen Herbrand-Modelle eingeführt.

Bevor wir uns diese Modellklasse genauer ansehen, definieren wir die Grundinstanziierung einer ERDF-Regel und eines ERDF-Programms.

Definition 6.37 (Grundinstanziierung einer ERDF-Regel und eines ERDF-Programms)
Sei V_R ein R-Vokabular und Var_{RDF} eine Menge von leeren RDF-Knoten.
Weiterhin seien r eine ERDF-Regel über V_R und Var_{RDF} sowie $P = \{r_1, r_2, \ldots, r_n\}$ ein ERDF-Programm über V_R und Var_{RDF}.
Außerdem bezeichne μ die Abbildung $\mu : Var^f_{RDF}(r) \to V_R$.

(a) Die Menge der ERDF-Regeln welche man erhält, indem jede freie Variable $x \in Var^f_{RDF}(r)$ von r durch $\mu(x)$ ersetzt wird, kennzeichnen wir mit $[r]^{\mu}_{V_R}$.

(b) Die *grundinstanziierte* Regelmenge von r ist definiert als $[r]_{V_R} = \bigcup_{\mu \in \mathcal{F}} [r]^{\mu}_{V_R}$, wobei $\mathcal{F}$ die Menge aller Abbildungen $\mu : Var^f_{RDF}(r) \to V_R$ bezeichne.

(c) Die *Grundinstanziierung* von P ist die folgende Regelmenge:
$[P]_{V_R} = \bigcup_{r_i \in P} [r_i]_{V_R}$ für $1 \leq i \leq n$.

□

Bei der Grundinstanziierung eines ERDF-Programms P werden also alle freien RDF-Knoten der ERDF-Regeln von P durch RDF URI-Referenzen oder RDF-Literale ersetzt. Da die freien RDF-Knoten einer ERDF-Regel jedoch wie allquantifizierte leere RDF-Knoten interpretiert werden, betrachten wir in diesem Fall alle Ersetzungsmöglichkeiten.
Nun führen wir die stabil erzeugten ERDF-Modelle einer ERDF-Ontologie ein.

Definition 6.38 (stabil erzeugtes ERDF-Modell einer ERDF-Ontologie)
Sei $O = \langle G, P \rangle$ eine ERDF-Ontologie über V_R und Var_{RDF}.
Weiterhin sei $I \in \mathcal{IB}^H_{ERDF}(O)$ eine Herbrand-Interpretation von O.

Die Interpretation I ist ein *stabil erzeugtes ERDF-Modell* von O genau dann, wenn eine Kette von Herbrand-Interpretationen $I_0 \preceq I_1 \preceq \ldots \preceq I_{k+1}$ mit $I_j \in \mathcal{IB}^H_{ERDF}(O)$ für $0 \leq j \leq k+1$ existiert, welche die folgenden Bedingungen erfüllt:

1. $I_k = I_{k+1} = I$
2. $I_0 \in Min\left(\left\{J \in \mathcal{IB}^H_{ERDF}(O) \;\middle|\; J \models sk(G)\right\}\right)$
3. Für jede Zahl n mit $0 < n \leq k+1$ gilt:
   $I_n \in Min\left(\left\{J \in \mathcal{IB}^H_{ERDF}(O) \;\middle|\; I_{n-1} \preceq J \text{ und } J \models Kopf(r), \forall r \in P[I_{n-1}, I]\right\}\right)$,
   wobei die Menge $P[I_{n-1}, I]$ wie folgt definiert ist:
   $P[I_{n-1}, I] = \left\{r \in [P]_{V_O} \;\middle|\; J \models Rumpf(r), \forall J \in \mathcal{IB}^H_{ERDF}(O) \text{ mit } I_{n-1} \preceq J \preceq I\right\}$. □

Für die Menge der stabil erzeugten ERDF-Modelle von O vereinbaren wir die Notation $Mod^{st}_{ERDF}(O)$.

Die Grundidee der stabil erzeugten ERDF-Modelle einer ERDF-Ontologie $O = \langle G, P\rangle$ besteht darin, ausgehend von den minimalen Herbrand-Modellen des in Skolemform befindlichen ERDF-Graphen $sk(G)$, durch iterative Anwendung der ERDF-Regeln, die gültigen Modelle bottom-up zu generieren. Dadurch werden die ERDF-Tripel von G wie Fakten behandelt, sind also in jedem stabil erzeugten Modell von O gültig. Zusätzlich wird durch die dritte Bedingung sichergestellt, dass jede eingesetzte ERDF-Regel auch anwendbar bleibt.

Sehen wir uns am Ende dieses Kapitels zur besseren Verständlichkeit der stabil erzeugten ERDF-Modelle das bekannte Tweety-Beispiel an:

Beispiel 6.39
Es sei `ex` der Namensraumpräfix der URI-Referenz `<http://www.example.org/>`, den wir für die Kennzeichnung eigener RDF URI-Referenzen verwenden.
Weiterhin sei $O = \langle G, P\rangle$ eine ERDF-Ontologie über V_R und $Var_{RDF} = \{x\}$, die wie folgt definiert ist:

$G = \{$ `rdf:type(ex:KannFliegen, erdf:TotalClass)`,
`rdf:type(ex:Pinguin, rdfs:Class)`,
`rdf:type(ex:Tweety, ex:Pinguin)` $\}$

$P = \{$ $\neg$`rdf:type(`x`, ex:KannFliegen)` $\Leftarrow$ `rdf:type(`x`, ex:Pinguin)` $\}$

□

Da V_O unendlich ist, existieren aufgrund der Totalität von `ex:KannFliegen` unendlich viele stabil erzeugte ERDF-Modelle von O. Für jedes $M \in Mod^{st}_{ERDF}(O)$ gilt jedoch $M \models \neg$`rdf:type(ex:Tweety, ex:KannFliegen)` und
$M \models \sim$ `rdf:type(ex:Tweety, ex:KannFliegen)`.
Begründung:
Für alle Interpretationen $I \in Min\left(\left\{J \in \mathcal{IB}^H_{ERDF}(O) \;\middle|\; J \models sk(G)\right\}\right)$ gilt entweder
$I \models$ `rdf:type(ex:Tweety, ex:KannFliegen)` oder

$I \models \neg$`rdf:type` $(\texttt{ex:Tweety}, \texttt{ex:KannFliegen})$.
Sei nun $I_0 \in Min\left(\left\{J \in \mathcal{IB}^H_{ERDF}(O) \;\middle|\; J \models sk(G)\right\}\right)$ mit
$I_0 \models \neg$`rdf:type` $(\texttt{ex:Tweety}, \texttt{ex:KannFliegen})$. Dann gilt $P[I_0, M] = P[M, M] = \{\neg\texttt{rdf:type}(\texttt{ex:Tweety}, \texttt{ex:KannFliegen}) \Leftarrow \texttt{rdf:type}(\texttt{ex:Tweety}, \texttt{ex:Pinguin})\}$.
Also wird M durch die Kette $I_0 \preceq M \preceq M$ erzeugt.
Anders verhält es sich bei den übrigen Herbrand-Interpretationen.
Sei $I_0 \in Min\left(\left\{J \in \mathcal{IB}^H_{ERDF}(O) \;\middle|\; J \models sk(G)\right\}\right)$ mit
$I_0 \models$ `rdf:type` $(\texttt{ex:Tweety}, \texttt{ex:KannFliegen})$. Dann gilt auch hier $P[I_0, I_0] = \{\neg\texttt{rdf:type}(\texttt{ex:Tweety}, \texttt{ex:KannFliegen}) \Leftarrow \texttt{rdf:type}(\texttt{ex:Tweety}, \texttt{ex:Pinguin})\}$.
Die Herbrand-Interpretation I_0 lässt sich jedoch aufgrund der Kohärenzbedingung nicht zu einem stabil erzeugten ERDF-Modell von O erweitern.

Abschließend führen wir die semantische Folgerungsrelation der stabil erzeugten ERDF-Modelle ein.

Definition 6.40 (Folgerungsrelation der stabil erzeugten ERDF-Modelle)
Sei V_R ein R-Vokabular und Var_{RDF} eine Menge von leeren RDF-Knoten. Weiterhin sei $O = \langle G, P\rangle$ eine ERDF-Ontologie über V_R und Var_{RDF} sowie F eine ERDF-Formel oder ein ERDF-Graph über V_R und Var_{RDF}.
Die *Folgerungsrelation* $\models_{st}$ ist definiert als:
$O \models_{st} F$ genau dann, wenn für alle $I \in Mod^{st}_{ERDF}(O)$ gilt $I \models F$.

□

7 Generalized Extended RDF

Eine große Herausforderung für das Semantic Web ist der Umgang mit widersprüchlichen Informationen, die nach [62] nicht vermeidbar sind. Die derzeitigen Ontologiebeschreibungssprachen des Semantic Web sind nicht in der Lage mit widersprüchlichen Informationen umzugehen, so dass eine inkonsistente Ontologie wertlos ist. Dies trifft in gleicher Weise auf ERDF zu, denn aufgrund der Kohärenzbedingung besitzt eine inkonsistente ERDF-Ontologie O kein stabil erzeugtes ERDF-Modell, so dass unter Verwendung der Relation $\models_{st}$ jede beliebige ERDF-Formel aus O logisch folgt.

Wir erweitern daher in diesem Kapitel Extended RDF unter Ausschluss der globalen Kohärenzbedingung zu *Generalized Extended RDF (GERDF)* und werden zeigen, dass mittels GERDF auch widersprüchliche Informationen modelliert werden können. Bevor wir die Semantik von GERDF formal einführen und deren Beziehung zu ERDF untersuchen, geben wir einen kurzen Überblick über Ursachen und Vorgehensmodelle bei inkonsistenten Ontologien.

7.1 Ursachen und Lösungsstrategien für inkonsistente Ontologien

Die offensichtlichste Ursache für die Inkonsistenz einer Ontologie ist ein Modellierungsfehler bei der Ontologieerstellung. Derartige Fehler treten bei sehr großen Ontologien häufig auf. Eine weitere Möglichkeit für Inkonsistenzen lässt sich mit der unzureichenden Ausdrucksfähigkeit der verwendeten Ontologiesprache begründen. So erlauben sehr viele formale Sprachen, darunter auch OWL, keine Ausnahmebehandlung. Dies führt sehr schnell zu Widersprüchen, wie beispielsweise in [46] anhand des bekannten Pinguinbeispiels gezeigt. Eine weitere inkonsistente Ontologie auf die in diesem Zusammenhang verwiesen werden sollte, ist die von Ian Horrocks entwickelte Ontologie *Mad Cows*[23]. Bei der Übersetzung einer Ontologie von einer formalen Sprache in eine andere können ebenfalls Inkonsistenzen entstehen. Als Beispiel sei hier die medizinische Ontologie *Diagnoses for Intensive Care Evaluation (DICE)* genannt. Die häufigste Ursache für inkonsistente Ontologien ist jedoch das sogenannte *Merging*, also das Zusammenfügen einzelner Ontologien zu einer gemeinsamen Ontologie. Aufgrund der dezentralen Struktur des Semantic

[23] http://www.daml.org/ontologies/399

Web sowie der unabhängigen Entwicklung der Teilontologien sind Widersprüche für derartige Fälle angesichts unterschiedlicher Ansichten und Standpunkte nicht zu vermeiden.

In der aktuellen Literatur lassen sich zwei Grundeinstellungen bezüglich widersprüchlichen Informationen erkennen. Einerseits vertreten viele Autoren den Standpunkt, dass Inkonsistenzen nicht erwünscht sind und entwickeln daher Strategien um Inkonsistenzen in Ontologien zu beseitigen, zu ignorieren oder diese erst gar nicht zuzulassen. So wäre beispielsweise die Verwendung einer Ontologiebeschreibungssprache wie RDFS denkbar, welche aufgrund der fehlenden Negation keine inkonsistenten Ontologien erlaubt. Derartige Sprachen besitzen jedoch eine sehr geringe Ausdruckskraft, wie bereits auf Seite 123 diskutiert. Eine andere Möglichkeit ist die Erweiterung der eingesetzten Ontologiebeschreibungssprache, so dass auch die Modellierung von Ausnahmen wie im Pinguin-Beispiel realisiert werden kann. Dies beseitigt jedoch nicht alle Inkonsistenzen. Die Berücksichtigung spezieller Methoden und Entwicklungspraktiken bei der Konstruktion neuer Ontologien soll ebenfalls helfen Inkonsistenzen zu vermeiden. Als Beispiel für derartige Vorschläge sei auf [25] verwiesen. In [63] wird eine Strategie entwickelt, um widersprüchliche Informationen mithilfe automatisierter Verfahren zu lokalisieren, mit dem Ziel diese anschließend manuell zu bereinigen. Dies setzt jedoch einen sehr hohen Sachverstand sowie Schreibrechte für die betreffende Ontologie voraus. Zusätzlich werden die Inkonsistenzen in der Regel durch Löschung der vermeintlich fehlerhaften Informationen beseitigt und sind deshalb mit einem Informationsverlust verbunden. Des Weiteren können Inkonsistenzen beim Merging vermieden werden, indem nicht die vollständigen Ontologien vereinigt werden, sondern lediglich zueinander konsistente Teilontologien. Dieser Ansatz wird zum Beispiel in [46] und [57] aufgegriffen.

Die zweite Grundeinstellung geht davon aus, dass Widersprüche generell unvermeidbar sind. Anstatt diese aufwändig zu beseitigen oder zu umgehen werden Strategien entwickelt, die robust gegenüber Inkonsistenzen sind. Eine Möglichkeit um diesen Ansatz umzusetzen ist die Verwendung einer logischen Sprache für die Beschreibung von Ontologien, die auch bei Inkonsistenzen keine beliebigen Schlussfolgerungen erlaubt. Diese logischen Sprachen werden als *parakonsistent* [59] bezeichnet. Die auf den nächsten Seiten dargestellte Erweiterung von ERDF gehört ebenfalls zur Menge der parakonsistenten Logiken.

7.2 GERDF-Interpretationen

Die Semantik von GERDF ist eine Erweiterung der ERDF-Semantik und basiert auf den abstrakten Syntaxkategorien von ERDF. Wir führen in diesem Abschnitt zunächst die GERDF-Interpretationen als Verfeinerung der ERDF-Interpretationen ein. Dabei orientieren wir uns an den Ideen von [3]. Anschließend werden die ko-

härenten GERDF-Interpretationen betrachtet, die eine echte Teilmenge der GERDF-Interpretationen bilden.

Um in GERDF Aussagen über die Kohärenz einzelner Klassen und Propertys formulieren zu können, führen wir zusätzliche RDF URI-Referenzen ein, die unter der Bezeichnung GERDF-Vokabular zusammengefasst werden.

Definition 7.1 (GERDF-Vokabular)
Es sei `gerdf` der Namensraumpräfix des GERDF-Vokabulars.
Das *GERDF-Vokabular* V_{GERDF} ist eine Menge von RDF URI-Referenzen, die wie folgt definiert ist:
$V_{GERDF} = \{$`gerdf:CoherentClass`, `gerdf:CoherentProperty`$\}$.
□

Als nächstes werden die semantischen GERDF-Strukturen als Erweiterung der semantischen ERDF-Strukturen festgelegt.

Definition 7.2 (semantische GERDF-Struktur)
Es sei V_R ein R-Vokabular.
Eine *semantische GERDF-Struktur* $\mathcal{S} = \langle \overline{\mathcal{S}}, CKl, CProp \rangle$ besteht aus:

- Einer semantischen ERDF-Struktur $\overline{\mathcal{S}}$ über V_R mit $\overline{\mathcal{S}} = \langle \langle Res, Prop, LV, rel^+, rel^- \rangle, Kl, TKl, TProp, Kext^+, Kext^- \rangle$.
- Einer Menge $CKl \subseteq Kl$, die als *Menge der kohärenten Klassen* von $\mathcal{S}$ bezeichnet wird.
- Einer Menge $CProp \subseteq Prop$, die *Menge der kohärenten Propertys* von $\mathcal{S}$ heißt.

□

Die semantischen ERDF-Strukturen werden also um zwei zusätzliche Mengen bereichert. Unter Verwendung der semantischen GERDF-Strukturen führen wir nun die GERDF-Interpretationen formal ein.

Definition 7.3 (GERDF-Interpretation)
Sei V_R ein R-Vokabular und Var_{RDF} eine Menge von leeren RDF-Knoten.
Eine *GERDF-Interpretation* $I = \langle \mathcal{S}, intU^P, intL^P, intV^P \rangle$ erfüllt die folgenden Bedingungen:

1. $\mathcal{S} = \langle \overline{\mathcal{S}}, CKl, CProp \rangle$ ist eine semantische GERDF-Struktur.
2. $\langle \overline{\mathcal{S}}, intU^P, intL^P, intV^P \rangle$ ist eine ERDF-Interpretation über $V_R \cup V_{GERDF}$ und Var_{RDF}.
3. $CKl = Kext^+ (int^* ($`gerdf:CoherentClass`$))$
4. $CProp = Kext^+ (int^* ($`gerdf:CoherentProperty`$))$
5. Falls $x \in CKl$, dann $Kext^+ (x) \cap Kext^- (x) = \emptyset$.

```
rdfs:subClassOf (gerdf:CoherentClass,rdfs:Class)
rdfs:subClassOf (gerdf:CoherentProperty,rdf:Property)
```

Abbildung 7.1: Die axiomatischen GERDF-Tripel

6. Falls $x \in CProp$, dann $rel^+(x) \cap rel^-(x) = \emptyset$.

7. I erfüllt die axiomatischen GERDF-Tripel von Abbildung 7.1.

□

Für die Menge der GERDF-Interpretationen über V_R und Var_{RDF} führen wir die Bezeichnung *GERDF-Interpretationsbereich* über V_R und Var_{RDF} ein und kennzeichnen diese mit $\mathcal{IB}_{GERDF}(V_R, Var_{RDF})$.

Nachstehend legen wir die GERDF-Folgerungsrelation zwischen zwei ERDF-Graphen fest.

Definition 7.4 (Modellmenge, Folgerungsrelation)
Sei V_R ein R-Vokabular und Var_{RDF} eine Menge von leeren RDF-Knoten. Weiterhin seien $G_1, G_2 \in Gr_{ERDF}(V_R, Var_{RDF})$ zwei ERDF-Graphen über V_R und Var_{RDF}.

(a) Die *GERDF-Modellmenge* eines ERDF-Graphen G_1 definieren wir wie folgt:
$Mod_{GERDF}(G_1) = \{I \in \mathcal{IB}_{GERDF}(V_R, Var_{RDF}) \mid I \models G_1\}$.

(b) Die *GERDF-Folgerungsrelation* $\models_{GERDF}$ ist definiert als:
$G_1 \models_{GERDF} G_2$ genau dann, wenn $Mod_{GERDF}(G_1) \subseteq Mod_{GERDF}(G_2)$.

□

Übereinstimmend mit ERDF erlauben die GERDF-Interpretationen die Darstellung von unvollständigen Informationen. Zusätzlich ist die Modellierung widersprüchlicher Aussagen realisierbar. Sei $I \in \mathcal{IB}_{GERDF}(V_R, Var_{RDF})$ eine GERDF-Interpretation und $p \in Prop$ eine Property. Dann ist für ein Paar $\langle x, y \rangle \in Res \times Res$ sowohl der Fall $\langle x, y \rangle \notin rel^+(p)$ und $\langle x, y \rangle \notin rel^-(p)$ als auch der Fall $\langle x, y \rangle \in rel^+(p)$ und $\langle x, y \rangle \in rel^-(p)$ denkbar. Mithilfe des ERDF-Vokabulars und des GERDF-Vokabulars ist es möglich, derartige Situationen für einzelne Propertys und Klassen auszuschließen.

Obwohl in GERDF widersprüchliche Informationen dargestellt werden können, tritt der sogenannte *XML-Clash*[24] auch bei GERDF-Interpretationen auf. Betrachten wir diesbezüglich das nachfolgende Beispiel:

[24] Als XML-Clash wird ein RDF-Tripel der Form `rdf:type` $(x$,`rdfs:Literal`$)$ bezeichnet, wobei x entweder ein nicht wohlgeformtes XML-Literal symbolisiert oder ein leerer RDF-Knoten ist, der einem nicht wohlgeformten XML-Literal zugewiesen wurde.

Beispiel 7.5
Es seien `"<notLegalXML"^^rdf:XMLLiteral` ein nicht wohlgeformtes XML-Literal, `ex:P`, `ex:a` $\in URI_{RDF}$ zwei RDF URI-Referenzen und G der folgende RDF-Graph:

$$G = \{ \begin{array}{l} \texttt{ex:P}\,(\texttt{ex:a}, \texttt{"<notLegalXML"^^rdf:XMLLiteral}), \\ \texttt{rdfs:range}\,(\texttt{ex:P}, \texttt{rdf:XMLLiteral}) \end{array} \} \tag{7.1}$$

□

Für dieses Beispiel gilt das folgende Lemma:

Lemma 7.6
Es existiert keine GERDF-Interpretation $I \in \mathcal{IB}_{GERDF}(V_R, Var_{RDF})$ *mit* $I \models G$.

Beweis:
Angenommen es existiert eine GERDF-Interpretation $I \in \mathcal{IB}_{GERDF}(V_R, Var_{RDF})$ mit $I \models G$, $I = \langle \mathcal{S}, intU^P, intL^P, intV^P \rangle$ und der semantischen GERDF-Struktur $\mathcal{S} = \langle \langle \langle Res, Prop, LV, rel^+, rel^- \rangle, Kl, TKl, TProp, Kext^+, Kext^- \rangle, CKl, CProp \rangle$.
Dann gilt $\langle int^*(\texttt{ex:P}), int^*(\texttt{rdf:XMLLiteral}) \rangle \in rel^+(int^*(\texttt{rdfs:range}))$ und $\langle int^*(\texttt{ex:a}), int^*(\texttt{"<notLegalXML"^^rdf:XMLLiteral}) \rangle \in rel^+(int^*(\texttt{ex:P}))$.
Aufgrund der elften Bedingung von Definition 6.17 folgt damit
$int^*(\texttt{"<notLegalXML"^^rdf:XMLLiteral}) \in Kext^+(int^*(\texttt{rdf:XMLLiteral}))$.
Somit gilt $intL^P(\texttt{"<notLegalXML"^^rdf:XMLLiteral}) \in LV$.
Dies ist ein Widerspruch zu Bedingung 22 von Definition 6.17.
Also existiert keine GERDF-Interpretation, welche den RDF-Graph G erfüllt.

■

Genau wie bei ERDF ist das Auftreten eines XML-Clashes nicht die einzige Ursache für die Unerfüllbarkeit eines ERDF-Graphen. Aufgrund der zusätzlichen Bedingungen für GERDF-Interpretationen existieren weitere ERDF-Graphen, die von keiner GERDF-Interpretation erfüllt werden.

Beispiel 7.7
Seien `ex:K`, `ex:a` $\in URI_{RDF}$ zwei RDF URI-Referenzen und $G \in Gr_{ERDF}(V_R, Var_{RDF})$ der folgende ERDF-Graph:

$$G = \{ \begin{array}{l} \texttt{rdf:type}\,(\texttt{ex:K}, \texttt{gerdf:CoherentClass}), \\ \texttt{rdf:type}\,(\texttt{ex:a}, \texttt{ex:K}), \\ \neg\texttt{rdf:type}\,(\texttt{ex:a}, \texttt{ex:K}) \end{array} \} \tag{7.2}$$

□

Dann existiert keine GERDF-Interpretation $I \in \mathcal{IB}_{GERDF}(V_R, Var_{RDF})$ mit $I \models G$.

7.3 Kohärente GERDF-Interpretationen

In diesem Abschnitt betrachten wir die kohärenten GERDF-Interpretationen, welche eine Teilmenge der GERDF-Interpretationen bilden. Zunächst führen wir die kohärenten GERDF-Interpretationen formal ein. Anschließend wird der Zusammenhang zwischen diesen Interpretationen und den in Definition 7.3 spezifizierten GERDF-Interpretationen näher untersucht.

Definition 7.8 (kohärente GERDF-Interpretation)
Sei $I = \langle \mathcal{S}, intU^P, intL^P, intV^P \rangle$ eine GERDF-Interpretation über V_R und Var_{RDF} mit $\mathcal{S} = \langle \langle \langle Res, Prop, LV, rel^+, rel^- \rangle, Kl, TKl, TProp, Kext^+, Kext^- \rangle, CKl, CProp \rangle$.
Die GERDF-Interpretation I ist eine *kohärente GERDF-Interpretation*, falls für alle $x \in Prop$ gilt: $rel^+(x) \cap rel^-(x) = \emptyset$.

□

Die Menge aller kohärenten GERDF-Interpretationen über V_R und Var_{RDF} nennen wir *kGERDF-Interpretationsbereich* über V_R und Var_{RDF}. Als Bezeichnung für den kGERDF-Interpretationsbereich legen wir die Notation $\mathcal{IB}^{ko}_{GERDF}(V_R, Var_{RDF})$ fest.

Für kohärente GERDF-Interpretationen wird die Folgerungsrelation für ERDF-Graphen nach dem Vorbild der GERDF-Folgerungsrelation definiert.

Definition 7.9 (Modellmenge, Folgerungsrelation)
Sei V_R ein R-Vokabular und Var_{RDF} eine Menge von leeren RDF-Knoten. Weiterhin seien $G_1, G_2 \in Gr_{ERDF}(V_R, Var_{RDF})$ zwei ERDF-Graphen über V_R und Var_{RDF}.

(a) Die *kGERDF-Modellmenge* eines ERDF-Graphen G_1 definieren wir wie folgt:
$Mod^{ko}_{GERDF}(G_1) = \left\{ I \in \mathcal{IB}^{ko}_{GERDF}(V_R, Var_{RDF}) \;\middle|\; I \models G_1 \right\}$.

(b) Die *kGERDF-Folgerungsrelation* $\models_{kGERDF}$ ist definiert als:
$G_1 \models_{kGERDF} G_2$ genau dann, wenn $Mod^{ko}_{GERDF}(G_1) \subseteq Mod^{ko}_{GERDF}(G_2)$.

□

Im Gegensatz zu ERDF ist es möglich, die global gültige Kohärenzbedingung für GERDF-Interpretationen unter Verwendung des RDF-Tripels

$$\texttt{rdfs:subClassOf}\,(\texttt{rdf:Property}, \texttt{gerdf:CoherentProperty}) \tag{7.3}$$

direkt zu fordern.

Im nun folgenden Teil dieses Kapitels wird die Beziehung zwischen den beiden Relationen $\models_{GERDF}$ und $\models_{kGERDF}$ genauer untersucht.

Lemma 7.10
Sei V_R ein R-Vokabular und Var_{RDF} eine Menge von leeren RDF-Knoten. Weiter seien $G_1, G_2 \in Gr_{RDF}(V_R, Var_{RDF})$ zwei RDF-Graphen.
Falls $G_1 \models_{kGERDF} G_2$, dann gilt $G_1 \models_{GERDF} G_2$.

Beweis:
Angenommen $G_1 \models_{kGERDF} G_2$. Es sei $I \in \mathcal{IB}_{GERDF}(V_R, Var_{RDF})$ eine GERDF-Interpretation mit $I = \langle \mathcal{S}_I, intU_I^P, intL_I^P, intV_I^P \rangle$, der semantischen GERDF-Struktur $\mathcal{S}_I = \langle \langle \langle Res_I, Prop_I, LV_I, rel_I^+, rel_I^- \rangle, Kl_I, TKl_I, TProp_I, Kext_I^+, Kext_I^- \rangle, CKl_I, CProp_I \rangle$ und $I \models G_1$. Es ist nun $I \models G_2$ zu zeigen.
Zunächst konstruieren wir aus der GERDF-Interpretation I eine kohärente GERDF-Interpretation K mit $K = \langle \mathcal{S}_K, intU_K^P, intL_K^P, intV_K^P \rangle$ und $\mathcal{S}_K = \langle \langle \langle Res_K, Prop_K, LV_K, rel_K^+, rel_K^- \rangle, Kl_K, TKl_K, TProp_K, Kext_K^+, Kext_K^- \rangle, CKl_K, CProp_K \rangle$ wie folgt:

1. $intU_K^P(u) := intU_I^P(u)$ für alle $u \in URI_{RDF}$
2. $intL_K^P(l) := intL_I^P(l)$ für alle $l \in tLit$
3. $intV_K^P(x) := intV_I^P(x)$ für alle $x \in Var_{RDF}$
4. $Res_K := Res_I$
5. $Prop_K := Prop_I$
6. $LV_K := LV_I$
7. $rel_K^+(p) := rel_I^+(p)$ für alle $p \in Prop_K$
8. $rel_K^-(p) := \{\langle x, y \rangle \in Res_K \times Res_K \mid \langle x, y \rangle \in rel_I^-(p) \text{ und } \langle x, y \rangle \notin rel_K^+(p)\}$
9. $Kl_K := Kl_I$
10. $TKl_K := TKl_I$
11. $TProp_K := TProp_I$
12. $Kext_K^+(\kappa) := Kext_I^+(\kappa)$ für alle $\kappa \in Kl_K$
13. $Kext_K^-(\kappa) := \{x \in Res_K \mid \langle x, \kappa \rangle \in rel_K^-(int_K^*(\texttt{rdf:type}))\}$
14. $CKl_K := CKl_I$
15. $CProp_K := CProp_I$

Als nächstes zeigen wir, dass K eine ERDF-Interpretation ist. Da sich die beiden Interpretationen I und K nach Konstruktionsvorschrift für K lediglich in den Abbildungen rel_I^- und rel_K^- sowie $Kext_I^-$ und $Kext_K^-$ unterscheiden, ist es ausreichend die Gültigkeit der Bedingungen 3, 13, 15, 19 und 20 von Definition 6.17 nachzuweisen. Bevor diese Eigenschaften gezeigt werden, weisen wir auf die folgenden Sachverhalte hin:
Beobachtung 1: Nach dem Konstruktionsverfahren für K gilt $rel_K^-(p) \subseteq rel_I^-(p)$ für alle $p \in Prop_K$.

Beobachtung 2: Aus Beobachtung 1 folgt direkt $Kext_K^-(\kappa) \subseteq Kext_I^-(\kappa)$ für alle $\kappa \in Kl_K$.
Beobachtung 3: Aufgrund der Konstruktion von K gilt ebenfalls $int_K^*(a) = int_I^*(a)$ für alle $a \in V_R \cup Var_{RDF}$.

(3) Behauptung: $x \in Kext_K^-(y)$ genau dann, wenn $\langle x, y\rangle \in rel_K^-(int_K^*(\texttt{rdf:type}))$
Beweis: dies gilt nach Konstruktion der Abbildung $Kext_K^-$

(13) Behauptung: falls $\langle x, y\rangle \in rel_K^+(int_K^*(\texttt{rdfs:subClassOf}))$, dann $Kext_K^-(y) \subseteq Kext_K^-(x)$
Beweis: Angenommen $\langle x, y\rangle \in rel_K^+(int_K^*(\texttt{rdfs:subClassOf}))$ und $Kext_K^-(y) \nsubseteq Kext_K^-(x)$. Dann existiert ein $z \in Res_K$ mit $z \in Kext_K^-(y)$ und $z \notin Kext_K^-(x)$. Damit folgt $\langle z, y\rangle \in rel_K^-(int_K^*(\texttt{rdf:type}))$ und $\langle z, x\rangle \notin rel_K^-(int_K^*(\texttt{rdf:type}))$. Da I eine GERDF-Interpretation ist, gilt jedoch $Kext_I^-(y) \subseteq Kext_I^-(x)$. Nach Beobachtung 2 gilt $Kext_K^-(y) \subseteq Kext_I^-(y)$ und $Kext_K^-(x) \subseteq Kext_I^-(x)$. Also $z \in Kext_I^-(y)$ und damit auch $z \in Kext_I^-(x)$. Da aber $z \notin Kext_K^-(x)$ muss aufgrund der Konstruktion von K die Bedingung $z \in Kext_K^+(x)$ erfüllt sein. Somit gilt nach Annahme auch $z \in Kext_K^+(y)$. Also $z \in Kext_K^+(y) \cap Kext_K^-(y)$ und damit $\langle z, y\rangle \in rel_K^+(int_K^*(\texttt{rdf:type})) \cap rel_K^-(int_K^*(\texttt{rdf:type}))$. Dies ist ein Widerspruch zu der Konstruktion von $rel_K^-(int_K^*(\texttt{rdf:type}))$. Also ist die Annahme falsch.

(15) Behauptung: falls $\langle x, y\rangle \in rel_K^+(int_K^*(\texttt{rdfs:subPropertyOf}))$, dann $rel_K^-(y) \subseteq rel_K^-(x)$
Beweis: Sei nun $\langle x, y\rangle \in rel_K^+(int_K^*(\texttt{rdfs:subPropertyOf}))$ und $rel_K^-(y) \nsubseteq rel_K^-(x)$. Dann existiert ein Paar $\langle z_1, z_2\rangle \in Res_K \times Res_K$ mit $\langle z_1, z_2\rangle \in rel_K^-(y)$ und $\langle z_1, z_2\rangle \notin rel_K^-(x)$. Da I eine GERDF-Interpretation ist, gilt $rel_I^-(y) \subseteq rel_I^-(x)$. Nach Beobachtung 1 gilt $rel_K^-(y) \subseteq rel_I^-(y)$ und $rel_K^-(x) \subseteq rel_I^-(x)$. Also $\langle z_1, z_2\rangle \in rel_I^-(y)$ und damit auch $\langle z_1, z_2\rangle \in rel_I^-(x)$. Da aber $\langle z_1, z_2\rangle \notin rel_K^-(x)$ muss aufgrund der Konstruktion von K die Bedingung $\langle z_1, z_2\rangle \in rel_K^+(x)$ erfüllt sein. Somit gilt nach Annahme auch $\langle z_1, z_2\rangle \in rel_K^+(y)$. Also $\langle z_1, z_2\rangle \in rel_K^+(y) \cap rel_K^-(y)$. Dies ist ein Widerspruch zu der Konstruktion von $rel_K^-(y)$. Also ist die Annahme falsch.

(19) Behauptung: falls $x \in TKl_K$, dann $Kext_K^+(x) \cup Kext_K^-(x) = Res_K$
Beweis: Angenommen $x \in TKl_K$ und $Kext_K^+(x) \cup Kext_K^-(x) \subsetneq Res_K$. Dann existiert ein $z \in Res_K$ mit $z \notin Kext_K^+(x)$ und $z \notin Kext_K^-(x)$. Da $Kext_K^+(x) = Kext_I^+(x)$ folgt damit $z \notin Kext_I^+(x)$. Aufgrund der Tatsache, dass I eine GERDF-Interpretation ist, gilt somit $z \in Kext_I^-(x)$. Dies ist ein Widerspruch zur Konstruktion von $rel_K^-(int_K^*(\texttt{rdf:type}))$, denn $z \notin Kext_K^-(x)$ und $z \notin Kext_K^+(x)$. Also ist die Annahme falsch.

(20) Behauptung: falls $x \in TProp_K$, dann $rel_K^+(x) \cup rel_K^-(x) = Res_K \times Res_K$
Beweis: Angenommen $x \in TProp_K$ und $rel_K^+(x) \cup rel_K^-(x) \subsetneq Res_K \times Res_K$.

Dann existiert ein Paar $\langle z_1, z_2 \rangle \in Res_K \times Res_K$ mit $\langle z_1, z_2 \rangle \notin rel_K^+(x)$ und $\langle z_1, z_2 \rangle \notin rel_K^-(x)$. Nach Konstruktion von K gilt $rel_K^+(x) = rel_I^+(x)$. Somit folgt $\langle z_1, z_2 \rangle \notin rel_I^+(x)$. Da I eine GERDF-Interpretation ist, gilt jedoch $rel_I^+(x) \cup rel_I^-(x) = Res_I \times Res_I$. Also $\langle z_1, z_2 \rangle \in rel_I^-(x)$, aber $\langle z_1, z_2 \rangle \notin rel_K^-(x)$. Dies ist ein Widerspruch zu der Konstruktion von $rel_K^-(x)$. Also ist die Annahme falsch.

Damit haben wir gezeigt, dass K eine ERDF-Interpretation ist.
Zusätzlich erfüllt K aufgrund des obigen Konstruktionsverfahrens und der Tatsache, dass I eine GERDF-Interpretation ist, alle Bedingungen von Definition 7.3. Also ist K eine GERDF-Interpretation. Nach Konstruktion der Abbildungen rel_K^+ und rel_K^- erfüllt K außerdem die Kohärenzbedingung für alle $x \in Prop_K$, so dass K eine kohärente GERDF-Interpretation ist.
Nun zeigen wir $K \models G_1$. Nach Annahme gilt $I \models G_1$. Also existiert eine ERDF-Modifikation $\widehat{I} \in Modi_{ERDF}^{VAR}$ von I bezüglich der Menge $VAR = Var_{RDF}(G_1)$ mit $\widehat{I} = \left\langle \mathcal{S}_{\widehat{I}}, intU_{\widehat{I}}^P, intL_{\widehat{I}}^P, intV_{\widehat{I}}^P \right\rangle$ und $\widehat{I} \models t$ für alle $t \in G_1$. Nach Definition 6.12 folgt $\widehat{K} \models t$ für alle $t \in G_1$ mit $\widehat{K} = \left\langle \mathcal{S}_K, intU_K^P, intL_K^P, intV_{\widehat{K}}^P \right\rangle$, wobei die Abbildung $intV_{\widehat{K}}^P : Var_{RDF} \to Res_K$ wie folgt definiert ist:

$$intV_{\widehat{K}}^P(x) = \begin{cases} intV_K^P(x) & \text{falls } x \in Var_{RDF} \setminus VAR \\ intV_{\widehat{I}}^P(x) & \text{falls } x \in VAR \end{cases}$$

Da $\widehat{K}$ nach Definition 6.10 eine ERDF-Modifikation von K bezüglich der Menge $Var_{RDF}(G_1)$ ist, gilt $K \models G_1$. Nach Voraussetzung folgt dadurch $K \models G_2$. Also existiert eine ERDF-Modifikation $\widetilde{K} \in Modi_{ERDF}^{VAR}(K)$ von K bezüglich der Menge $VAR = Var_{RDF}(G_2)$ mit $\widetilde{K} \models t$ für alle $t \in G_2$. Analog zu obiger Argumentation existiert eine GERDF-Interpretation $\widetilde{I} = \left\langle \mathcal{S}_I, intU_I^P, intL_I^P, intV_{\widetilde{I}}^P \right\rangle$ mit der Abbildung $intV_{\widetilde{I}}^P : Var_{RDF} \to Res_I$ mit

$$intV_{\widetilde{I}}^P(x) = \begin{cases} intV_I^P(x) & \text{falls } x \in Var_{RDF} \setminus VAR \\ intV_{\widetilde{K}}^P(x) & \text{falls } x \in VAR \end{cases}$$

so dass $\widetilde{I} \models t$ für alle $t \in G_2$. Damit folgt $I \models G_2$. Da I beliebig gewählt ist, gilt also $Mod_{GERDF}(G_1) \subseteq Mod_{GERDF}(G_2)$ und damit $G_1 \models_{GERDF} G_2$.

■

Die Umkehrung von Lemma 7.10 ist ebenfalls gültig, so dass wir das folgende Resultat erhalten:

Satz 7.11
Sei V_R ein R-Vokabular und Var_{RDF} eine Menge von leeren RDF-Knoten. Weiter seien $G_1, G_2 \in Gr_{RDF}(V_R, Var_{RDF})$ zwei RDF-Graphen.
Dann gilt $G_1 \models_{kGERDF} G_2$ genau dann, wenn $G_1 \models_{GERDF} G_2$.

Beweis:
Nach Lemma 7.10 gilt bereits: falls $G_1 \models_{kGERDF} G_2$, dann gilt auch $G_1 \models_{GERDF} G_2$. Es muss also nur die Umkehrung dieses Lemmas gezeigt werden.
Seien $G_1, G_2 \in Gr_{RDF}(V_R, Var_{RDF})$ zwei RDF-Graphen mit $G_1 \models_{GERDF} G_2$. Weiter sei $I \in \mathcal{IB}^{ko}_{GERDF}(V_R, Var_{RDF})$ eine kohärente GERDF-Interpretation mit $I \models G_1$. Wir zeigen nun $I \models G_2$.
Nach Definition 7.8 gilt $\mathcal{IB}^{ko}_{GERDF}(V_R, Var_{RDF}) \subseteq \mathcal{IB}_{GERDF}(V_R, Var_{RDF})$. Also ist I eine GERDF-Interpretation. Da $I \models G_1$ folgt damit nach Voraussetzung $I \models G_2$.
■

Damit haben wir gezeigt, dass die beiden Folgerungsrelationen $\models_{GERDF}$ und $\models_{kGERDF}$ für RDF-Graphen äquivalent sind. Für ERDF-Graphen trifft dies jedoch nicht zu, wie das nachfolgende Beispiel zeigt.

Beispiel 7.12
Es seien `ex:p`, `ex:s`, `ex:o` drei RDF URI-Referenzen und $G \in Gr_{ERDF}(V_R, Var_{RDF})$ der folgende ERDF-Graph:

$$G = \{\ \texttt{ex:p}(\texttt{ex:s}, \texttt{ex:o}), \neg\texttt{ex:p}(\texttt{ex:s}, \texttt{ex:o})\ \} \tag{7.4}$$

□

Es existiert keine kohärente GERDF-Interpretation $I \in \mathcal{IB}^{ko}_{GERDF}(V_R, Var_{RDF})$ mit $I \models G$, so dass $G \models_{kGERDF} \widehat{G}$ für alle $\widehat{G} \in Gr_{ERDF}(V_R, Var_{RDF})$ gilt. Für GERDF-Interpretationen trifft dies jedoch nicht zu.

7.4 Beziehung zu ERDF

Nachdem im vorangegangenen Abschnitt die Beziehung zwischen den beiden Folgerungsrelationen $\models_{kGERDF}$ und $\models_{GERDF}$ untersucht wurde, analysieren wir nun den Zusammenhang zwischen der kERDF-Folgerungsrelation und der kGERDF-Folgerungsrelation.
Zu Beginn wird die Gültigkeit des folgenden Lemmas gezeigt:

Lemma 7.13
Sei V_R ein R-Vokabular und Var_{RDF} eine Menge von leeren RDF-Knoten. Weiter seien $G_1, G_2 \in Gr_{RDF}(V_R, Var_{RDF})$ zwei RDF-Graphen mit $V_R(G_1) \cap V_{GERDF} = \emptyset$ und $V_R(G_2) \cap V_{GERDF} = \emptyset$.
Falls $G_1 \models_{kGERDF} G_2$, dann gilt $G_1 \models_{kERDF} G_2$.

Beweis:
Angenommen $G_1 \models_{kGERDF} G_2$. Es sei $I \in \mathcal{IB}^{ko}_{ERDF}(V_R, Var_{RDF})$ eine kohärente ERDF-Interpretation mit $I = \langle \mathcal{S}_I, intU^P_I, intL^P_I, intV^P_I \rangle$, der semantischen ERDF-Struktur $\mathcal{S}_I = \langle \langle Res_I, Prop_I, LV_I, rel^+_I, rel^-_I \rangle, Kl_I, TKl_I, TProp_I, Kext^+_I, Kext^-_I \rangle$ und $I \models G_1$. Es ist nun zu zeigen, dass $I \models G_2$.

Wir konstruieren aus der Interpretation I eine kohärente GERDF-Interpretation K mit $K = \langle \mathcal{S}_K, intU_K^P, intL_K^P, intV_K^P \rangle$ und der semantischen GERDF-Struktur $\mathcal{S}_K = \langle \langle \langle Res_K, Prop_K, LV_K, rel_K^+, rel_K^- \rangle, Kl_K, TKl_K, TProp_K, Kext_K^+, Kext_K^- \rangle, CKl_K, CProp_K \rangle$. Diese Konstruktion ist wesentlich aufwändiger als in Lemma 7.10, denn das Universum von K enthält zusätzliche Elemente. Wir führen zunächst die Abbildung $res : V_{GERDF} \to R$ mit $R \cap Res_I = \emptyset$ ein, welche das GERDF-Vokabular nach R abbildet. Basierend auf der kohärenten ERDF-Interpretation I und der Abbildung res, konstruieren wir die Interpretation K wie folgt:

- $Res_K := Res_I \cup res\,(V_{GERDF})$
- $intU_K^P\,(u) := intU_I^P\,(u)$ für alle $u \in URI_{RDF} \setminus V_{GERDF}$ und
  $intU_K^P\,(u) := res\,(u)$ für alle $u \in V_{GERDF}$
- $intL_K^P\,(l) := intL_I^P\,(l)$ für alle $l \in tLit$
- $intV_K^P\,(x) := intV_I^P\,(x)$ für alle $x \in Var_{RDF}$
- die Abbildung $\widehat{rel}_K^+ : Res_K \to \mathcal{P}\,(Res_K \times Res_K)$ sei folgendermaßen definiert:

  (A1) Falls $x, y, z \in Res_I$ und $\langle x, y \rangle \in rel_I^+\,(z)$, dann $\langle x, y \rangle \in \widehat{rel}_K^+\,(z)$.

  (A2) $\langle res\,(\texttt{gerdf:CoherentClass}), int_K^*\,(\texttt{rdfs:Class}) \rangle \in \widehat{rel}_K^+\,(int_K^*\,(\texttt{rdfs:subClassOf}))$

  (A3) $\langle res\,(\texttt{gerdf:CoherentProperty}), int_K^*\,(\texttt{rdf:Property}) \rangle \in \widehat{rel}_K^+\,(int_K^*\,(\texttt{rdfs:subClassOf}))$

  Ausgehend von diesen Bedingungen werden nun die nachfolgenden Regeln so lange angewendet bis ein Fixpunkt erreicht wird.

  (B1) Falls $\langle x, y \rangle \in \widehat{rel}_K^+\,(int_K^*\,(\texttt{rdfs:domain}))$ und $\langle z, w \rangle \in \widehat{rel}_K^+\,(x)$, dann $\langle z, y \rangle \in \widehat{rel}_K^+\,(int_K^*\,(\texttt{rdf:type}))$.

  (B2) Falls $\langle x, y \rangle \in \widehat{rel}_K^+\,(int_K^*\,(\texttt{rdfs:range}))$ und $\langle z, w \rangle \in \widehat{rel}_K^+\,(x)$, dann $\langle w, y \rangle \in \widehat{rel}_K^+\,(int_K^*\,(\texttt{rdf:type}))$.

  (B3) Falls $\langle x, int_K^*\,(\texttt{rdfs:Class}) \rangle \in \widehat{rel}_K^+\,(int_K^*\,(\texttt{rdf:type}))$, dann $\langle x, int_K^*\,(\texttt{rdfs:Resource}) \rangle \in \widehat{rel}_K^+\,(int_K^*\,(\texttt{rdfs:subClassOf}))$.

  (B4) Falls $\langle x, y \rangle \in \widehat{rel}_K^+\,(int_K^*\,(\texttt{rdfs:subClassOf}))$, dann $\langle x, int_K^*\,(\texttt{rdfs:Class}) \rangle \in \widehat{rel}_K^+\,(int_K^*\,(\texttt{rdf:type}))$.

  (B5) Falls $\langle x, y \rangle \in \widehat{rel}_K^+\,(int_K^*\,(\texttt{rdfs:subClassOf}))$, dann $\langle y, int_K^*\,(\texttt{rdfs:Class}) \rangle \in \widehat{rel}_K^+\,(int_K^*\,(\texttt{rdf:type}))$.

  (B6) Falls $\langle x, y \rangle \in \widehat{rel}_K^+\,(int_K^*\,(\texttt{rdfs:subClassOf}))$ und $\langle z, x \rangle \in \widehat{rel}_K^+\,(int_K^*\,(\texttt{rdf:type}))$, dann $\langle z, y \rangle \in \widehat{rel}_K^+\,(int_K^*\,(\texttt{rdf:type}))$.

(B7) Falls $\langle x, int_K^*(\texttt{rdfs:Class})\rangle \in \widehat{rel}_K^+(int_K^*(\texttt{rdf:type}))$, dann $\langle x, x\rangle \in \widehat{rel}_K^+(int_K^*(\texttt{rdfs:subClassOf}))$.

(B8) Falls $\langle x, y\rangle \in \widehat{rel}_K^+(int_K^*(\texttt{rdfs:subClassOf}))$ und $\langle y, z\rangle \in \widehat{rel}_K^+(int_K^*(\texttt{rdfs:subClassOf}))$, dann $\langle x, z\rangle \in \widehat{rel}_K^+(int_K^*(\texttt{rdfs:subClassOf}))$.

(B9) Falls $\langle x, y\rangle \in \widehat{rel}_K^+(int_K^*(\texttt{rdfs:subPropertyOf}))$, dann $\langle x, int_K^*(\texttt{rdf:Property})\rangle \in \widehat{rel}_K^+(int_K^*(\texttt{rdf:type}))$.

(B10) Falls $\langle x, y\rangle \in \widehat{rel}_K^+(int_K^*(\texttt{rdfs:subPropertyOf}))$, dann $\langle y, int_K^*(\texttt{rdf:Property})\rangle \in \widehat{rel}_K^+(int_K^*(\texttt{rdf:type}))$.

(B11) Falls $\langle x, y\rangle \in \widehat{rel}_K^+(int_K^*(\texttt{rdfs:subPropertyOf}))$ und $\langle z, w\rangle \in \widehat{rel}_K^+(x)$, dann $\langle z, w\rangle \in \widehat{rel}_K^+(y)$.

(B12) Falls $\langle x, int_K^*(\texttt{rdf:Property})\rangle \in \widehat{rel}_K^+(int_K^*(\texttt{rdf:type}))$, dann $\langle x, x\rangle \in \widehat{rel}_K^+(int_K^*(\texttt{rdfs:subPropertyOf}))$.

(B13) Falls $\langle x, y\rangle \in \widehat{rel}_K^+(int_K^*(\texttt{rdfs:subPropertyOf}))$ und $\langle y, z\rangle \in \widehat{rel}_K^+(int_K^*(\texttt{rdfs:subPropertyOf}))$, dann $\langle x, z\rangle \in \widehat{rel}_K^+(int_K^*(\texttt{rdfs:subPropertyOf}))$.

(B14) Falls $\langle x, int_K^*(\texttt{rdfs:Datatype})\rangle \in \widehat{rel}_K^+(int_K^*(\texttt{rdf:type}))$, dann $\langle x, int_K^*(\texttt{rdfs:Literal})\rangle \in \widehat{rel}_K^+(int_K^*(\texttt{rdfs:subClassOf}))$.

(B15) Falls $\langle x, int_K^*(\texttt{rdfs:ContainerMembershipProperty})\rangle \in \widehat{rel}_K^+(int_K^*(\texttt{rdf:type}))$, dann $\langle x, int_K^*(\texttt{rdfs:member})\rangle \in \widehat{rel}_K^+(int_K^*(\texttt{rdfs:subPropertyOf}))$.

Damit ist die Abbildung $\widehat{rel}_K^+$ vollständig definiert.

- $Prop_K := \left\{ x \in Res_K \;\middle|\; \langle x, int_K^*(\texttt{rdf:Property})\rangle \in \widehat{rel}_K^+(int_K^*(\texttt{rdf:type})) \right\}$
- $rel_K^+(x) := \widehat{rel}_K^+(x)$ für alle $x \in Prop_K$
- $LV_K := \left\{ x \in Res_K \mid \langle x, int_K^*(\texttt{rdfs:Literal})\rangle \in rel_K^+(int_K^*(\texttt{rdf:type})) \right\}$
- die Abbildung $rel_K^- : Prop_K \to \mathcal{P}(Res_K \times Res_K)$ sei folgenderweise definiert:

  (C1) Falls $x, y, z \in Res_I$ und $\langle x, y\rangle \in rel_I^-(z)$, dann $\langle x, y\rangle \in rel_K^-(z)$.

  (C2) Falls $\langle x, int_K^*(\texttt{erdf:TotalClass})\rangle \in rel_K^+(int_K^*(\texttt{rdf:type}))$ und $\langle y, x\rangle \notin rel_K^+(int_K^*(\texttt{rdf:type}))$, dann $\langle y, x\rangle \in rel_K^-(int_K^*(\texttt{rdf:type}))$.

  (C3) Falls $\langle x, int_K^*(\texttt{erdf:TotalProperty})\rangle \in rel_K^+(int_K^*(\texttt{rdf:type}))$ und $\langle y, z\rangle \notin rel_K^+(x)$, dann $\langle y, z\rangle \in rel_K^-(x)$.

Ausgehend von diesen Bedingungen werden nun die nachfolgenden Regeln so lange angewendet bis ein Fixpunkt erreicht wird.

(D1) Falls $\langle x, y\rangle \in rel_K^+ (int_K^* (\texttt{rdfs:subClassOf}))$ und $\langle z, y\rangle \in rel_K^- (int_K^* (\texttt{rdf:type}))$, dann $\langle z, x\rangle \in rel_K^- (int_K^* (\texttt{rdf:type}))$.

(D2) Falls $\langle x, y\rangle \in rel_K^+ (int_K^* (\texttt{rdfs:subPropertyOf}))$ und $\langle z, w\rangle \in rel_K^- (y)$, dann $\langle z, w\rangle \in rel_K^- (x)$.

Damit ist die Abbildung rel_K^- vollständig definiert.

- $Kl_K := \{x \in Res_K \mid \langle x, int_K^* (\texttt{rdfs:Class})\rangle \in rel_K^+ (int_K^* (\texttt{rdf:type}))\}$
- $TKl_K := \{x \in Res_K \mid \langle x, int_K^* (\texttt{erdf:TotalClass})\rangle \in rel_K^+ (int_K^* (\texttt{rdf:type}))\}$
- $TProp_K := \{x \in Res_K \mid \langle x, int_K^* (\texttt{erdf:TotalProperty})\rangle \in rel_K^+ (int_K^* (\texttt{rdf:type}))\}$
- die Abbildung $Kext_K^+ : Kl_K \to \mathcal{P}(Res_K)$ ist wie folgt definiert: $x \in Kext_K^+ (y)$ genau dann, wenn $\langle x, y\rangle \in rel_K^+ (int_K^* (\texttt{rdf:type}))$
- die Abbildung $Kext_K^- : Kl_K \to \mathcal{P}(Res_K)$ ist wie folgt definiert: $x \in Kext_K^- (y)$ genau dann, wenn $\langle x, y\rangle \in rel_K^- (int_K^* (\texttt{rdf:type}))$
- $CKl_K := \{x \in Res_K \mid \langle x, int_K^* (\texttt{gerdf:CoherentClass})\rangle \in rel_K^+ (int_K^* (\texttt{rdf:type}))\}$
- $CProp_K := \{x \in Res_K \mid \langle x, int_K^* (\texttt{gerdf:CoherentProperty})\rangle \in rel_K^+ (int_K^* (\texttt{rdf:type}))\}$

Damit ist die Interpretation K vollständig definiert. Nun ist zu zeigen, dass K eine kohärente GERDF-Interpretation ist.
Wir beweisen zunächst das folgende Hilfslemma:
Lemma 1:
Seien $x, y, z \in Res_K$, dann gilt: $\langle x, y\rangle \in \widehat{rel}_K^+ (z)$ genau dann, wenn $\langle x, y\rangle \in rel_K^+ (z)$.
Beweis:
Sei nun $\langle x, y\rangle \in \widehat{rel}_K^+ (z)$. Dann gilt aufgrund der Definition von $\widehat{rel}_K^+$ entweder
(i) $z \in Prop_I$ oder (ii) es existiert ein $w \in Res_K$ mit
$\langle w, z\rangle \in \widehat{rel}_K^+ (int_K^* (\texttt{rdfs:subPropertyOf}))$.

(i) Angenommen $z \in Prop_I$. Dann gilt $\langle z, int_I^* (\texttt{rdf:Property})\rangle \in rel_I^+ (int_I^* (\texttt{rdf:type}))$ und damit gilt auch $\langle z, int_K^* (\texttt{rdf:Property})\rangle \in rel_I^+ (int_K^* (\texttt{rdf:type}))$. Nach (A1) folgt $\langle z, int_K^* (\texttt{rdf:Property})\rangle \in \widehat{rel}_K^+ (int_K^* (\texttt{rdf:type}))$.
Also $z \in Prop_K$ und somit $\langle x, y\rangle \in rel_K^+ (z)$.

(ii) Angenommen es existiert ein $w \in Res_K$ mit $\langle w, z\rangle \in \widehat{rel}_K^+ (int_K^* (\texttt{rdfs:subPropertyOf}))$. Aufgrund von (B10) folgt damit $\langle z, int_K^* (\texttt{rdf:Property})\rangle \in \widehat{rel}_K^+ (int_K^* (\texttt{rdf:type}))$.
Also $z \in Prop_K$ und somit $\langle x, y\rangle \in rel_K^+ (z)$.

Sei nun $\langle x, y\rangle \in rel_K^+(z)$. Dann folgt nach Definition von rel_K^+ sofort $\langle x, y\rangle \in \widehat{rel}_K^+(z)$. Damit ist das Lemma 1 bewiesen.
Als nächstes zeigen wir, dass $\left\langle \widehat{\mathcal{S}_K}, intU_K^P, intL_K^P, intV_K^P \right\rangle$ eine ERDF-Interpretation über $V_R \cup V_{GERDF}$ und Var_{RDF} ist mit
$\widehat{\mathcal{S}_K} = \left\langle \left\langle Res_K, Prop_K, LV_K, rel_K^+, rel_K^- \right\rangle, Kl_K, TKl_K, TProp_K, Kext_K^+, Kext_K^- \right\rangle$.

- Die Bedingungen 1. und 2. von Definition 6.17 sind aufgrund der Konstruktion erfüllt.

- Die Bedingung 3. gilt nach Definition der beiden Abbildungen $Kext_K^+$ und $Kext_K^-$.

- Behauptung: $Prop_K = Kext_K^+(int_K^*(\texttt{rdf:Property}))$
  Beweis:
  "$\subseteq$" Sei $x \in Prop_K$, dann gilt $\langle x, int_K^*(\texttt{rdf:Property})\rangle \in \widehat{rel}_K^+(int_K^*(\texttt{rdf:type}))$ nach Konstruktion von $Prop_K$. Aufgrund von Lemma 1 folgt damit $\langle x, int_K^*(\texttt{rdf:Property})\rangle \in rel_K^+(int_K^*(\texttt{rdf:type}))$ und somit gilt auch $x \in Kext_K^+(int_K^*(\texttt{rdf:Property}))$.
  "$\supseteq$" Sei $x \in Kext_K^+(int_K^*(\texttt{rdf:Property}))$, dann gilt $\langle x, int_K^*(\texttt{rdf:Property})\rangle \in rel_K^+(int_K^*(\texttt{rdf:type}))$. Aufgrund von Lemma 1 folgt damit $\langle x, int_K^*(\texttt{rdf:Property})\rangle \in \widehat{rel}_K^+(int_K^*(\texttt{rdf:type}))$ und somit gilt auch $x \in Prop_K$.

- Behauptung: $Kl_K = Kext_K^+(int_K^*(\texttt{rdfs:Class}))$
  Beweis: Sei $x \in Kl_K$
  gdw. $\langle x, int_K^*(\texttt{rdfs:Class})\rangle \in rel_K^+(int_K^*(\texttt{rdf:type}))$ nach Definition von Kl_K
  gdw. $x \in Kext_K^+(int_K^*(\texttt{rdfs:Class}))$ nach Definition von $Kext_K^+$.

- Behauptung: $Res_K = Kext_K^+(int_K^*(\texttt{rdfs:Resource}))$
  Beweis:
  "$\supseteq$" gilt nach Definition von $Kext_K^+$
  "$\subseteq$" Sei $x \in Res_K$. Dann unterscheiden wir zwei Fälle.
  Fall 1: Sei $x \in Res_I$. Dann gilt $x \in Kext_I^+(int_I^*(\texttt{rdfs:Resource}))$, da I eine ERDF-Interpretation ist. Damit gilt nach Definition von $Kext_I^+$ auch $\langle x, int_I^*(\texttt{rdfs:Resource})\rangle \in rel_I^+(int_I^*(\texttt{rdf:type}))$.
  Daraus folgt $\langle x, int_K^*(\texttt{rdfs:Resource})\rangle \in \widehat{rel}_K^+(int_K^*(\texttt{rdf:type}))$. Nach Lemma 1 gilt $\langle x, int_K^*(\texttt{rdfs:Resource})\rangle \in rel_K^+(int_K^*(\texttt{rdf:type}))$. Damit gilt $x \in Kext_K^+(int_K^*(\texttt{rdfs:Resource}))$, nach Definition von $Kext_K^+$.
  Fall 2: Sei $x \in res(V_{GERDF})$. Dann gilt aufgrund von (A2) und (A3) entweder $\langle x, int_K^*(\texttt{rdfs:Class})\rangle \in \widehat{rel}_K^+(int_K^*(\texttt{rdfs:subClassOf}))$ oder $\langle x, int_K^*(\texttt{rdf:Property})\rangle \in \widehat{rel}_K^+(int_K^*(\texttt{rdfs:subClassOf}))$. Damit folgt $\langle x, int_K^*(\texttt{rdfs:Class})\rangle \in \widehat{rel}_K^+(int_K^*(\texttt{rdf:type}))$ nach (B4). Weiterhin gilt

$\langle int_K^*(\texttt{rdf:type}), int_K^*(\texttt{rdfs:Resource})\rangle \in \widehat{rel}_K^+(int_K^*(\texttt{rdfs:domain}))$, da I die axiomatischen RDFS-Tripel erfüllt und demzufolge auch das RDF-Tripel $\langle int_I^*(\texttt{rdf:type}), int_I^*(\texttt{rdfs:Resource})\rangle \in rel_I^+(int_I^*(\texttt{rdfs:domain}))$. Also $\langle x, int_K^*(\texttt{rdfs:Resource})\rangle \in \widehat{rel}_K^+(int_K^*(\texttt{rdf:type}))$ nach (B1). Somit folgt aufgrund von Lemma 1 $\langle x, int_K^*(\texttt{rdfs:Resource})\rangle \in rel_K^+(int_K^*(\texttt{rdf:type}))$ und damit $x \in Kext_K^+(int_K^*(\texttt{rdfs:Resource}))$.

- Behauptung: $LV_K = Kext_K^+(int_K^*(\texttt{rdfs:Literal}))$
  Beweis: Sei $x \in LV_K$
  gdw. $\langle x, int_K^*(\texttt{rdfs:Literal})\rangle \in rel_K^+(int_K^*(\texttt{rdf:type}))$ nach Definition von LV_K
  gdw. $x \in Kext_K^+(int_K^*(\texttt{rdfs:Literal}))$ nach Definition von $Kext_K^+$.

- Behauptung: $TKl_K = Kext_K^+(int_K^*(\texttt{erdf:TotalClass}))$
  Beweis: Sei $x \in TKl_K$
  gdw. $\langle x, int_K^*(\texttt{erdf:TotalClass})\rangle \in rel_K^+(int_K^*(\texttt{rdf:type}))$
  nach Definition von TKl_K
  gdw. $x \in Kext_K^+(int_K^*(\texttt{erdf:TotalClass}))$ nach Definition von $Kext_K^+$.

- Behauptung: $TProp_K = Kext_K^+(int_K^*(\texttt{erdf:TotalProperty}))$
  Beweis: Sei $x \in TProp_K$
  gdw. $\langle x, int_K^*(\texttt{erdf:TotalProperty})\rangle \in rel_K^+(int_K^*(\texttt{rdf:type}))$
  nach Definition von $TProp_K$
  gdw. $x \in Kext_K^+(int_K^*(\texttt{erdf:TotalProperty}))$ nach Definition von $Kext_K^+$.

- Behauptung: Falls $\langle x, y\rangle \in rel_K^+(int_K^*(\texttt{rdfs:domain}))$ und $\langle z, w\rangle \in rel_K^+(x)$, dann $z \in Kext_K^+(y)$.
  Beweis: Seien $\langle x, y\rangle \in rel_K^+(int_K^*(\texttt{rdfs:domain}))$ und $\langle z, w\rangle \in rel_K^+(x)$. Dann gilt $\langle x, y\rangle \in \widehat{rel}_K^+(int_K^*(\texttt{rdfs:domain}))$ und $\langle z, w\rangle \in \widehat{rel}_K^+(x)$, nach Lemma 1. Aufgrund von (B1) folgt damit $\langle z, y\rangle \in \widehat{rel}_K^+(int_K^*(\texttt{rdf:type}))$.
  Also $\langle z, y\rangle \in rel_K^+(int_K^*(\texttt{rdf:type}))$ und somit $z \in Kext_K^+(y)$.

- Behauptung: Falls $\langle x, y\rangle \in rel_K^+(int_K^*(\texttt{rdfs:range}))$ und $\langle z, w\rangle \in rel_K^+(x)$, dann $w \in Kext_K^+(y)$.
  Beweis: Seien $\langle x, y\rangle \in rel_K^+(int_K^*(\texttt{rdfs:range}))$ und $\langle z, w\rangle \in rel_K^+(x)$. Dann gilt $\langle x, y\rangle \in \widehat{rel}_K^+(int_K^*(\texttt{rdfs:range}))$ und $\langle z, w\rangle \in \widehat{rel}_K^+(x)$, nach Lemma 1. Aufgrund von (B2) folgt damit $\langle w, y\rangle \in \widehat{rel}_K^+(int_K^*(\texttt{rdf:type}))$.
  Also $\langle w, y\rangle \in rel_K^+(int_K^*(\texttt{rdf:type}))$ und somit $w \in Kext_K^+(y)$.

- Behauptung: Falls $x \in Kl_K$, dann
  $\langle x, int_K^*(\texttt{rdfs:Resource})\rangle \in rel_K^+(int_K^*(\texttt{rdfs:subClassOf}))$.
  Beweis: Sei $x \in Kl_K$. Dann gilt $\langle x, int_K^*(\texttt{rdfs:Class})\rangle \in rel_K^+(int_K^*(\texttt{rdf:type}))$, nach Definition von Kl_K. Daraus folgt

$\langle x, int_K^*(\texttt{rdfs:Class})\rangle \in \widehat{rel}_K^+(int_K^*(\texttt{rdf:type}))$, nach Lemma 1. Aufgrund von (B3) gilt nun $\langle x, int_K^*(\texttt{rdfs:Resource})\rangle \in \widehat{rel}_K^+(int_K^*(\texttt{rdfs:subClassOf}))$ und somit $\langle x, int_K^*(\texttt{rdfs:Resource})\rangle \in rel_K^+(int_K^*(\texttt{rdfs:subClassOf}))$.

- Behauptung: Falls $\langle x, y\rangle \in rel_K^+(int_K^*(\texttt{rdfs:subClassOf}))$, dann $x, y \in Kl_K$, $Kext_K^+(x) \subseteq Kext_K^+(y)$ und $Kext_K^-(y) \subseteq Kext_K^-(x)$.
  Beweis:
  Sei $\langle x, y\rangle \in rel_K^+(int_K^*(\texttt{rdfs:subClassOf}))$. Dann gilt $\langle x, y\rangle \in \widehat{rel}_K^+(int_K^*(\texttt{rdfs:subClassOf}))$. Aufgrund von (B4) folgt damit $\langle x, int_K^*(\texttt{rdfs:Class})\rangle \in \widehat{rel}_K^+(int_K^*(\texttt{rdf:type}))$.
  Also $\langle x, int_K^*(\texttt{rdfs:Class})\rangle \in rel_K^+(int_K^*(\texttt{rdf:type}))$ und somit $x \in Kl_K$.
  Sei $\langle x, y\rangle \in rel_K^+(int_K^*(\texttt{rdfs:subClassOf}))$. Dann gilt $\langle x, y\rangle \in \widehat{rel}_K^+(int_K^*(\texttt{rdfs:subClassOf}))$. Aufgrund von (B5) folgt damit $\langle y, int_K^*(\texttt{rdfs:Class})\rangle \in \widehat{rel}_K^+(int_K^*(\texttt{rdf:type}))$.
  Also $\langle y, int_K^*(\texttt{rdfs:Class})\rangle \in rel_K^+(int_K^*(\texttt{rdf:type}))$ und somit $y \in Kl_K$.
  Sei $\langle x, y\rangle \in rel_K^+(int_K^*(\texttt{rdfs:subClassOf}))$ und $z \in Kext_K^+(x)$. Dann gilt $\langle x, y\rangle \in \widehat{rel}_K^+(int_K^*(\texttt{rdfs:subClassOf}))$ und $\langle z, x\rangle \in \widehat{rel}_K^+(int_K^*(\texttt{rdf:type}))$. Aufgrund von (B6) folgt damit $\langle z, y\rangle \in \widehat{rel}_K^+(int_K^*(\texttt{rdf:type}))$.
  Also $\langle z, y\rangle \in rel_K^+(int_K^*(\texttt{rdf:type}))$ und somit $z \in Kext_K^+(y)$.
  Sei $\langle x, y\rangle \in rel_K^+(int_K^*(\texttt{rdfs:subClassOf}))$ und $z \in Kext_K^-(y)$. Dann gilt $\langle x, y\rangle \in rel_K^+(int_K^*(\texttt{rdfs:subClassOf}))$ und $\langle z, y\rangle \in rel_K^-(int_K^*(\texttt{rdf:type}))$. Aufgrund von (D1) folgt damit $\langle z, x\rangle \in rel_K^-(int_K^*(\texttt{rdf:type}))$.
  Also $z \in Kext_K^-(x)$.

- Behauptung: $rel_K^+(int_K^*(\texttt{rdfs:subClassOf}))$ ist reflexiv auf der Menge Kl_K.
  Beweis:
  Es ist zu zeigen: Falls $x \in Kl_K$, dann $\langle x, x\rangle \in rel_K^+(int_K^*(\texttt{rdfs:subClassOf}))$.
  Sei $x \in Kl_K$. Dann gilt $\langle x, int_K^*(\texttt{rdfs:Class})\rangle \in rel_K^+(int_K^*(\texttt{rdf:type}))$ und damit $\langle x, int_K^*(\texttt{rdfs:Class})\rangle \in \widehat{rel}_K^+(int_K^*(\texttt{rdf:type}))$. Aufgrund von (B7) folgt damit $\langle x, x\rangle \in \widehat{rel}_K^+(int_K^*(\texttt{rdfs:subClassOf}))$.
  Also $\langle x, x\rangle \in rel_K^+(int_K^*(\texttt{rdfs:subClassOf}))$.

- Behauptung: $rel_K^+(int_K^*(\texttt{rdfs:subClassOf}))$ ist transitiv auf der Menge Kl_K.
  Beweis:
  Es ist zu zeigen: Falls $x, y, z \in Kl_K$, $\langle x, y\rangle \in rel_K^+(int_K^*(\texttt{rdfs:subClassOf}))$ und $\langle y, z\rangle \in rel_K^+(int_K^*(\texttt{rdfs:subClassOf}))$, dann auch $\langle x, z\rangle \in rel_K^+(int_K^*(\texttt{rdfs:subClassOf}))$.
  Seien nun $x, y, z \in Kl_K$, $\langle x, y\rangle \in rel_K^+(int_K^*(\texttt{rdfs:subClassOf}))$ und $\langle y, z\rangle \in rel_K^+(int_K^*(\texttt{rdfs:subClassOf}))$. Dann gilt $\langle x, y\rangle \in \widehat{rel}_K^+(int_K^*(\texttt{rdfs:subClassOf}))$ und $\langle y, z\rangle \in \widehat{rel}_K^+(int_K^*(\texttt{rdfs:subClassOf}))$. Aufgrund von (B8) folgt damit

$\langle x, z\rangle \in \widehat{rel}_K^+ (int_K^* (\texttt{rdfs:subClassOf}))$.
Also $\langle x, z\rangle \in rel_K^+ (int_K^* (\texttt{rdfs:subClassOf}))$.

- Die Bedingungen 15. bis 18. von Definition 6.17 werden auf analoge Weise unter Verwendung von (B9) bis (B15) sowie (D2) gezeigt.

- Behauptung: Falls $x \in TKl_K$, dann $Kext_K^+ (x) \cup Kext_K^- (x) = Res_K$.
Beweis: Sei $x \in TKl_K$.
Es ist nun zu zeigen, dass für alle $y \in Res_K$ gilt $y \in Kext_K^+ (x) \cup Kext_K^- (x)$.
Angenommen es existiert ein $y \in Res_K$ mit $y \notin Kext_K^+ (x)$.
Da $x \in TKl_K$ gilt $\langle x, int_K^* (\texttt{erdf:TotalClass})\rangle \in rel_K^+ (int_K^* (\texttt{rdf:type}))$ und $\langle y, x\rangle \notin rel_K^+ (int_K^* (\texttt{rdf:type}))$. Aufgrund von (C2) folgt damit $\langle y, x\rangle \in rel_K^- (int_K^* (\texttt{rdf:type}))$. Also $y \in Kext_K^- (x)$.

- Behauptung: Falls $x \in TProp_K$, dann $rel_K^+ (x) \cup rel_K^- (x) = Res_K \times Res_K$.
Beweis: Sei $x \in TProp_K$.
Es ist nun zu zeigen, dass für alle Paare $\langle y, z\rangle \in Res_K \times Res_K$ gilt $\langle y, z\rangle \in rel_K^+ (x) \cup rel_K^- (x)$.
Angenommen es existiert ein Paar $\langle y, z\rangle \in Res_K \times Res_K$ mit $\langle y, z\rangle \notin rel_K^+ (x)$.
Da $x \in TProp_K$ gilt $\langle x, int_K^* (\texttt{erdf:TotalProperty})\rangle \in rel_K^+ (int_K^* (\texttt{rdf:type}))$ und $\langle y, z\rangle \notin rel_K^+ (x)$. Aufgrund von (C3) folgt damit $\langle y, z\rangle \in rel_K^- (x)$.

- Behauptung: Falls $\texttt{"s"\^\^rdf:XMLLiteral} \in V_R$ und s ein wohlgeformtes XML-Literal ist, dann ist $intL_K^P (\texttt{"s"\^\^rdf:XMLLiteral})$ der XML-Wert von s.
Beweis:
Sei s ein wohlgeformtes XML-Literal und $\texttt{"s"\^\^rdf:XMLLiteral} \in V_R$.
Nach Konstruktion von K gilt $intL_K^P (\texttt{"s"\^\^rdf:XMLLiteral}) = intL_I^P (\texttt{"s"\^\^rdf:XMLLiteral})$. Da I eine ERDF-Interpretation ist, folgt $intL_I^P (\texttt{"s"\^\^rdf:XMLLiteral})$ ist der XML-Wert von s.
Also $intL_K^P (\texttt{"s"\^\^rdf:XMLLiteral})$ ist der XML-Wert von s.

- Behauptung: Sei $\texttt{"s"\^\^rdf:XMLLiteral} \in V_R$ ein wohlgeformtes XML-Literal, dann gilt $intL_K^P (\texttt{"s"\^\^rdf:XMLLiteral}) \in Kext_K^+ (int_K^* (\texttt{rdf:XMLLiteral}))$.
Beweis:
Sei s ein wohlgeformtes XML-Literal und $\texttt{"s"\^\^rdf:XMLLiteral} \in V_R$.
Dann gilt $intL_I^P (\texttt{"s"\^\^rdf:XMLLiteral}) \in Kext_I^+ (int_I^* (\texttt{rdf:XMLLiteral}))$, da I eine ERDF-Interpretation ist. Somit gilt auch $\langle intL_I^P (\texttt{"s"\^\^rdf:XMLLiteral}), int_I^* (\texttt{rdf:XMLLiteral})\rangle \in rel_I^+ (int_I^* (\texttt{rdf:type}))$. Aufgrund von (A1) folgt $\langle intL_K^P (\texttt{"s"\^\^rdf:XMLLiteral}), int_K^* (\texttt{rdf:XMLLiteral})\rangle \in \widehat{rel}_K^+ (int_K^* (\texttt{rdf:type}))$. Nach Lemma 1 gilt $\langle intL_K^P (\texttt{"s"\^\^rdf:XMLLiteral}), int_K^* (\texttt{rdf:XMLLiteral})\rangle \in$

$rel_K^+(int_K^*(\texttt{rdf:type}))$.
Also $intL_K^P(\texttt{"s"\^\^rdf:XMLLiteral}) \in Kext_K^+(int_K^*(\texttt{rdf:XMLLiteral}))$.

Bevor wir fortfahren, wollen wir auf die nachfolgende Beobachtung hinweisen.
Beobachtung 1: Seien $x, y, z \in Res_I$ und $\langle x, y\rangle \in rel_K^+(z)$, dann gilt $\langle x, y\rangle \in rel_I^+(z)$.
Diese Beobachtung lässt sich per Induktion über die Konstruktion von $\widehat{rel}_K^+$ ausgehend von (A1) zeigen.

- Behauptung: Falls $\texttt{"s"\^\^rdf:XMLLiteral} \in V_R$ und s ein nicht wohlgeformtes XML-Literal ist, dann gilt $intL_K^P(\texttt{"s"\^\^rdf:XMLLiteral}) \notin LV_K$.
  Beweis:
  Sei s ein nicht wohlgeformtes XML-Literal und $\texttt{"s"\^\^rdf:XMLLiteral} \in V_R$.
  Angenommen $intL_K^P(\texttt{"s"\^\^rdf:XMLLiteral}) \in LV_K$. Dann gilt
  $\langle intL_K^P(\texttt{"s"\^\^rdf:XMLLiteral}), int_K^*(\texttt{rdfs:Literal})\rangle \in rel_K^+(int_K^*(\texttt{rdf:type}))$.
  Da $intL_K^P(\texttt{"s"\^\^rdf:XMLLiteral}) = intL_I^P(\texttt{"s"\^\^rdf:XMLLiteral}) \in Res_I$, $int_K^*(\texttt{rdfs:Literal}) = int_I^*(\texttt{rdfs:Literal}) \in Res_I$ und $int_K^*(\texttt{rdf:type}) = int_I^*(\texttt{rdf:type}) \in Res_I$, folgt nach Beobachtung 1
  $\langle intL_I^P(\texttt{"s"\^\^rdf:XMLLiteral}), int_I^*(\texttt{rdfs:Literal})\rangle \in rel_I^+(int_I^*(\texttt{rdf:type}))$.
  Somit gilt $intL_I^P(\texttt{"s"\^\^rdf:XMLLiteral}) \in Kext_I^+(int_I^*(\texttt{rdfs:Literal}))$ und damit $intL_I^P(\texttt{"s"\^\^rdf:XMLLiteral}) \in LV_I$. Dies ist ein Widerspruch zu Definition 6.17. Also $intL_K^P(\texttt{"s"\^\^rdf:XMLLiteral}) \notin LV_K$.

- Falls $\texttt{"s"\^\^rdf:XMLLiteral} \in V_R$ und s ein nicht wohlgeformtes XML-Literal ist, dann gilt $intL_K^P(\texttt{"s"\^\^rdf:XMLLiteral}) \in Kext_K^-(int_K^*(\texttt{rdfs:Literal}))$.
  Beweis:
  Sei s ein nicht wohlgeformtes XML-Literal und $\texttt{"s"\^\^rdf:XMLLiteral} \in V_R$.
  Da I eine kohärente ERDF-Interpretation ist, gilt
  $intL_I^P(\texttt{"s"\^\^rdf:XMLLiteral}) \in Kext_I^-(int_I^*(\texttt{rdfs:Literal}))$. Damit gilt
  $\langle intL_I^P(\texttt{"s"\^\^rdf:XMLLiteral}), int_I^*(\texttt{rdfs:Literal})\rangle \in rel_I^-(int_I^*(\texttt{rdf:type}))$.
  Aufgrund von (C1) folgt
  $\langle intL_K^P(\texttt{"s"\^\^rdf:XMLLiteral}), int_K^*(\texttt{rdfs:Literal})\rangle \in rel_K^-(int_K^*(\texttt{rdf:type}))$.
  Also $intL_K^P(\texttt{"s"\^\^rdf:XMLLiteral}) \in Kext_K^-(int_K^*(\texttt{rdfs:Literal}))$.

- Da I eine kohärente ERDF-Interpretation ist, erfüllt K aufgrund von (A1) die axiomatischen RDF-Tripel, RDFS-Tripel und ERDF-Tripel.

Damit haben wir gezeigt, dass $\left\langle \widehat{\mathcal{S}_K}, intU_K^P, intL_K^P, intV_K^P \right\rangle$ eine ERDF-Interpretation über $V_R \cup V_{GERDF}$ und Var_{RDF} ist.
Nachfolgend wird nun bewiesen, dass K eine GERDF-Interpretation ist.

- Die Bedingung 1. von Definition 7.3 ist aufgrund der Konstruktion erfüllt.

- Die Bedingung 2. ist nach obigen Beweis ebenfalls gültig.

- Behauptung: $CKl_K = Kext_K^+ \left(int_K^* \left(\texttt{gerdf:CoherentClass}\right)\right)$
  Beweis: Sei $x \in CKl_K$
  gdw. $\langle x, int_K^* \left(\texttt{gerdf:CoherentClass}\right)\rangle \in rel_K^+ \left(int_K^* \left(\texttt{rdf:type}\right)\right)$
  nach Definition von CKl_K
  gdw. $x \in Kext_K^+ \left(int_K^* \left(\texttt{gerdf:CoherentClass}\right)\right)$ nach Definition von $Kext_K^+$.

- Behauptung: $CProp_K = Kext_K^+ \left(int_K^* \left(\texttt{gerdf:CoherentProperty}\right)\right)$
  Beweis: Sei $x \in CProp_K$
  gdw. $\langle x, int_K^* \left(\texttt{gerdf:CoherentProperty}\right)\rangle \in rel_K^+ \left(int_K^* \left(\texttt{rdf:type}\right)\right)$
  nach Definition von $CProp_K$
  gdw. $x \in Kext_K^+ \left(int_K^* \left(\texttt{gerdf:CoherentProperty}\right)\right)$ nach Definition von $Kext_K^+$.

- Behauptung: Falls $x \in CKl_K$, dann $Kext_K^+ (x) \cap Kext_K^- (x) = \emptyset$
  Beweis: Nach Konstruktion von K gilt: $CKl_K = \emptyset$.

- Behauptung: Falls $x \in CProp_K$, dann $rel_K^+ (x) \cap rel_K^- (x) = \emptyset$
  Beweis: Nach Konstruktion von K gilt: $CProp_K = \emptyset$.

- Aufgrund von (A2) und (A3) erfüllt K die axiomatischen GERDF-Tripel

Also ist K eine GERDF-Interpretation.
Aufgrund der Tatsache, dass I eine kohärente ERDF-Interpretation ist, erfüllt auch K nach obiger Konstruktion, insbesondere wegen (C2) und (C3), die Kohärenzbedingung. Also ist K eine kohärente GERDF-Interpretation.
Jetzt zeigen wir $K \models G_1$.
Nach Annahme gilt $I \models G_1$. Also existiert eine ERDF-Modifikation $\widehat{I} \in Modi_{ERDF}^{VAR}$ von I bezüglich der Menge $VAR = Var_{RDF} (G_1)$ mit $\widehat{I} = \left\langle \mathcal{S}_{\widehat{I}}, intU_{\widehat{I}}^P, intL_{\widehat{I}}^P, intV_{\widehat{I}}^P \right\rangle$ und $\widehat{I} \models t$ für alle $t \in G_1$. Somit gilt $p \in URI_{RDF} \setminus V_{GERDF}$,
$s, o \in (URI_{RDF} \cup Var_{RDF} \cup Lit) \setminus V_{GERDF}$, $int_{\widehat{I}}^*(p) \in Prop_{\widehat{I}}$ und
$\left\langle int_{\widehat{I}}^*(s), int_{\widehat{I}}^*(o) \right\rangle \in rel_{\widehat{I}}^+ \left(int_{\widehat{I}}^* (p)\right)$ für alle $t \in G_1$ mit $t = p\,(s, o)$.
Es existiert eine kohärente GERDF-Interpretation $\widehat{K} = \left\langle \mathcal{S}_K, intU_K^P, intL_K^P, intV_{\widehat{K}}^P \right\rangle$ mit

$$intV_{\widehat{K}}^P (x) = \begin{cases} intV_K^P (x) & \text{falls } x \in Var_{RDF} \setminus VAR \\ intV_{\widehat{I}}^P (x) & \text{falls } x \in VAR \end{cases}$$

für die nach obigen Konstruktionsverfahren die folgenden Eigenschaften gelten:

- $int_{\widehat{K}}^*(p) = int_{\widehat{I}}^*(p) \in Prop_{\widehat{I}} \subseteq Prop_K$

- $int_{\widehat{K}}^*(s) = int_{\widehat{I}}^*(s)$, $int_{\widehat{K}}^*(o) = int_{\widehat{I}}^*(o)$ und damit
  $\left\langle int_{\widehat{K}}^*(s), int_{\widehat{K}}^*(o) \right\rangle \in rel_K^+ \left(int_{\widehat{K}}^* (p)\right)$ aufgrund von (A1) und Lemma 1

Also $\widehat{K} \models t$ für alle $t \in G_1$ und damit $\widehat{K} \models G_1$. Da $\widehat{K}$ eine ERDF-Modifikation von K bezüglich der Menge $Var_{RDF}(G_1)$ ist, gilt $K \models G_1$. Weil K eine kohärente GERDF-Interpretation ist, folgt nach der Annahme $G_1 \models_{kGERDF} G_2$ auch $K \models G_2$.
Bevor wir nun $I \models G_2$ zeigen, wollen wir auf den folgenden Sachverhalt hinweisen:
Sei $f : Res_K \to Res_I$ eine Abbildung, die wie folgt definiert ist:

$$f(x) = \begin{cases} x & \text{falls } x \in Res_I \\ int_I^*(\texttt{rdfs:Class}) & \text{falls } x = res(\texttt{gerdf:CoherentClass}) \\ int_I^*(\texttt{rdf:Property}) & \text{falls } x = res(\texttt{gerdf:CoherentProperty}) \end{cases}$$

Dann gelten die folgenden Beobachtungen:
Beobachtung 2:
Falls $\langle x, y \rangle \in rel_K^+(z)$ und $y \in res(V_{GERDF})$, dann gilt $x = y$.
Beobachtung 3:
Falls $\langle x, y \rangle \in rel_K^+(z)$, dann $\langle f(x), f(y) \rangle \in rel_I^+(f(z))$.
Analog zu Beobachtung 1, werden auch diese Beobachtungen per Induktion über die Konstruktion von $\widehat{rel}_K^+$ gezeigt.
Nach obigen Überlegungen gilt $K \models G_2$.
Also existiert eine ERDF-Modifikation $\widehat{K} \in Modi_{ERDF}^{VAR}$ von K bezüglich der Menge $VAR = Var_{RDF}(G_2)$ mit $\widehat{K} = \left\langle \mathcal{S}_{\widehat{K}}, intU_{\widehat{K}}^P, intL_{\widehat{K}}^P, intV_{\widehat{K}}^P \right\rangle$ und $\widehat{K} \models t$ für alle $t \in G_2$.
Sei nun $\widehat{I} = \left\langle \mathcal{S}_I, intU_I^P, intL_I^P, intV_{\widehat{I}}^P \right\rangle$ mit

$$intV_{\widehat{I}}^P(x) = \begin{cases} intV_{\widehat{K}}^P(x) & \text{falls } x \in Res_I \\ int_I^*(\texttt{rdfs:Class}) & \text{falls } intV_{\widehat{K}}^P(x) = res(\texttt{gerdf:CoherentClass}) \\ int_I^*(\texttt{rdf:Property}) & \text{falls } intV_{\widehat{K}}^P(x) = \\ & res(\texttt{gerdf:CoherentProperty}) \end{cases}$$

Wir zeigen nun, dass $\widehat{I} \models t$ für alle $t \in G_2$.
Sei $t \in G_2$ mit $t = p(s, o)$. Da $\widehat{K} \models t$, gilt $int_{\widehat{K}}^*(p) \in Prop_{\widehat{K}}$.
Daraus folgt $\left\langle int_{\widehat{K}}^*(p), int_{\widehat{K}}^*(\texttt{rdf:Property}) \right\rangle \in rel_K^+\left(int_{\widehat{K}}^*(\texttt{rdf:type})\right)$.
Da $p \notin V_{GERDF}$ gilt $int_{\widehat{K}}^*(p) = int_{\widehat{I}}^*(p)$, $int_{\widehat{K}}^*(\texttt{rdf:Property}) = int_{\widehat{I}}^*(\texttt{rdf:Property})$ und $int_{\widehat{K}}^*(\texttt{rdf:type}) = int_{\widehat{I}}^*(\texttt{rdf:type})$. Nach Beobachtung 1 folgt somit $\left\langle int_{\widehat{I}}^*(p), int_{\widehat{I}}^*(\texttt{rdf:Property}) \right\rangle \in rel_I^+\left(int_{\widehat{I}}^*(\texttt{rdf:type})\right)$. Also $int_{\widehat{I}}^*(p) \in Prop_{\widehat{I}}$.
Zusätzlich gilt $\left\langle int_{\widehat{K}}^*(s), int_{\widehat{K}}^*(o) \right\rangle \in rel_K^+\left(int_{\widehat{K}}^*(p)\right)$.
Um $\left\langle int_{\widehat{I}}^*(s), int_{\widehat{I}}^*(o) \right\rangle \in rel_I^+\left(int_{\widehat{I}}^*(p)\right)$ zu zeigen, betrachten wir die folgende Fallunterscheidung:
1. Fall:
Seien $s, o \in Var_{RDF}(G_2)$, $intV_{\widehat{K}}^P(s) \notin res(V_{GERDF})$ und $intV_{\widehat{K}}^P(o) \notin res(V_{GERDF})$.

Dann gilt $int^*_{\widehat{K}}(s) = int^*_{\widehat{I}}(s) \in Res_I$, $int^*_{\widehat{K}}(o) = int^*_{\widehat{I}}(o) \in Res_I$ und $int^*_{\widehat{K}}(p) = int^*_{\widehat{I}}(p) \in Res_I$. Da $\left\langle int^*_{\widehat{K}}(s), int^*_{\widehat{K}}(o)\right\rangle \in rel^+_K\left(int^*_{\widehat{K}}(p)\right)$ folgt nach Beobachtung 1 auch $\left\langle int^*_{\widehat{I}}(s), int^*_{\widehat{I}}(o)\right\rangle \in rel^+_I\left(int^*_{\widehat{I}}(p)\right)$.
2. Fall:
Seien $s, o \in Var_{RDF}(G_2)$, $intV^P_{\widehat{K}}(s) \in res(V_{GERDF})$ und $intV^P_{\widehat{K}}(o) \notin res(V_{GERDF})$.
Dann gilt $int^*_{\widehat{K}}(o) = int^*_{\widehat{I}}(o) \in Res_I$ und $int^*_{\widehat{K}}(p) = int^*_{\widehat{I}}(p) \in Res_I$.
Da $\left\langle int^*_{\widehat{K}}(s), int^*_{\widehat{K}}(o)\right\rangle \in rel^+_K\left(int^*_{\widehat{K}}(p)\right)$ folgt nach Beobachtung 3 auch $\left\langle int^*_{\widehat{I}}(s), int^*_{\widehat{I}}(o)\right\rangle \in rel^+_I\left(int^*_{\widehat{I}}(p)\right)$.
3. Fall:
Sei $o \in Var_{RDF}(G_2)$ mit $intV^P_{\widehat{K}}(o) \in res(V_{GERDF})$.
Aufgrund von Beobachtung 2 gilt $s \in Var_{RDF}(G_2)$ und $intV^P_{\widehat{K}}(s) = intV^P_{\widehat{K}}(o)$.
Da $\left\langle int^*_{\widehat{K}}(s), int^*_{\widehat{K}}(o)\right\rangle \in rel^+_K\left(int^*_{\widehat{K}}(p)\right)$ und $int^*_{\widehat{K}}(p) = int^*_{\widehat{I}}(p) \in Res_I$ folgt nach Beobachtung 3 auch $\left\langle int^*_{\widehat{I}}(s), int^*_{\widehat{I}}(o)\right\rangle \in rel^+_I\left(int^*_{\widehat{I}}(p)\right)$.
4. Fall:
Seien $s, o \notin Var_{RDF}(G_2)$.
Dann gilt $int^*_{\widehat{K}}(s) = int^*_{\widehat{I}}(s) \in Res_I$, $int^*_{\widehat{K}}(o) = int^*_{\widehat{I}}(o) \in Res_I$ und $int^*_{\widehat{K}}(p) = int^*_{\widehat{I}}(p) \in Res_I$. Da $\left\langle int^*_{\widehat{K}}(s), int^*_{\widehat{K}}(o)\right\rangle \in rel^+_K\left(int^*_{\widehat{K}}(p)\right)$ folgt nach Beobachtung 1 auch $\left\langle int^*_{\widehat{I}}(s), int^*_{\widehat{I}}(o)\right\rangle \in rel^+_I\left(int^*_{\widehat{I}}(p)\right)$.
Damit gilt für jeden der obigen Fälle $\left\langle int^*_{\widehat{I}}(s), int^*_{\widehat{I}}(o)\right\rangle \in rel^+_I\left(int^*_{\widehat{I}}(p)\right)$.
Also $\widehat{I} \models t$ für alle $t \in G_2$. Da $\widehat{I}$ nach obiger Konstruktion eine ERDF-Modifikation von I bezüglich der Menge $VAR = Var_{RDF}(G_2)$ ist, gilt $I \models G_2$.

■

Mithilfe von Lemma 7.13 werden wir nun den nachfolgenden Satz beweisen.

Satz 7.14
Sei V_R ein R-Vokabular und Var_{RDF} eine Menge von leeren RDF-Knoten. Weiter seien $G_1, G_2 \in Gr_{RDF}(V_R, Var_{RDF})$ zwei RDF-Graphen mit $V_R(G_1) \cap V_{GERDF} = \emptyset$ und $V_R(G_2) \cap V_{GERDF} = \emptyset$.
Dann gilt $G_1 \models_{kGERDF} G_2$ genau dann, wenn $G_1 \models_{kERDF} G_2$.

Beweis:
Nach Lemma 7.13 gilt bereits: falls $G_1 \models_{kGERDF} G_2$, dann gilt auch $G_1 \models_{kERDF} G_2$. Es muss also die Umkehrung dieses Lemmas gezeigt werden.
Seien $G_1, G_2 \in Gr_{RDF}(V_R, Var_{RDF})$ zwei RDF-Graphen mit $V_R(G_1) \cap V_{GERDF} = \emptyset$, $V_R(G_2) \cap V_{GERDF} = \emptyset$ und $G_1 \models_{kERDF} G_2$. Weiter sei $I \in \mathcal{IB}^{ko}_{GERDF}(V_R, Var_{RDF})$ eine kohärente GERDF-Interpretation mit $I \models G_1$, $I = \langle \mathcal{S}_I, intU^P_I, intL^P_I, intV^P_I\rangle$ und $\mathcal{S}_I = \langle\langle\langle Res_I, Prop_I, LV_I, rel^+_I, rel^-_I\rangle, Kl_I, TKl_I, TProp_I, Kext^+_I, Kext^-_I\rangle, CKl_I, CProp_I\rangle$.
Wir zeigen nun $I \models G_2$.

Nach Definition 7.3 ist $K = \langle \mathcal{S}_K, intU_I^P, intL_I^P, intV_I^P \rangle$ mit der semantischen ERDF-Struktur $\mathcal{S}_K = \langle \langle Res_I, Prop_I, LV_I, rel_I^+, rel_I^- \rangle, Kl_I, TKl_I, TProp_I, Kext_I^+, Kext_I^- \rangle$ eine ERDF-Interpretation. Da I eine kohärente GERDF-Interpretation ist, erfüllt auch K die Kohärenzbedingung. Nach Voraussetzung gilt $I \models G_1$. Somit gilt auch $K \models G_1$ und damit $K \models G_2$. Also $I \models G_2$.

■

Korollar 7.15
Sei V_R ein R-Vokabular und Var_{RDF} eine Menge von leeren RDF-Knoten. Weiter seien $G_1, G_2 \in Gr_{RDF}(V_R, Var_{RDF})$ zwei RDF-Graphen mit $V_R(G_1) \cap (V_{GERDF} \cup V_{ERDF}) = \emptyset$ und $V_R(G_2) \cap (V_{GERDF} \cup V_{ERDF}) = \emptyset$.
Dann gilt $G_1 \models_{RDFS} G_2$ genau dann, wenn $G_1 \models_{GERDF} G_2$.

Beweis:
Das Korollar ist eine Konsequenz der Sätze 6.21, 7.14 und 7.11.
Seien $G_1, G_2 \in Gr_{RDF}(V_R, Var_{RDF})$ zwei RDF-Graphen mit
$V_R(G_1) \cap (V_{GERDF} \cup V_{ERDF}) = \emptyset$ und $V_R(G_2) \cap (V_{GERDF} \cup V_{ERDF}) = \emptyset$.
Es gilt $G_1 \models_{RDFS} G_2$ genau dann, wenn $G_1 \models_{kERDF} G_2$, nach Satz 6.21.
Aufgrund von Satz 7.14 folgt $G_1 \models_{RDFS} G_2$ genau dann, wenn $G_1 \models_{kGERDF} G_2$.
Somit gilt $G_1 \models_{RDFS} G_2$ genau dann, wenn $G_1 \models_{GERDF} G_2$, nach Satz 7.11.

■

Die in diesem Kapitel eingeführte Folgerungsrelation $\models_{GERDF}$ ist also eine konservative Erweiterung der RDFS-Folgerungsrelation.

7.5 Erweiterungen der GERDF-Interpretationen

Übereinstimmend mit ERDF wird auch die Menge der GERDF-Interpretationen weiter eingeschränkt. In Analogie zu Kapitel 6.6 führen wir zunächst die Herbrand-Interpretationen für GERDF-Interpretationen ein. Anschließend werden die stabil erzeugten ERDF-Modelle auf die GERDF-Interpretationen übertragen.

7.5.1 GERDF-Herbrand-Interpretationen

Um in der Lage zu sein, die GERDF-Herbrand-Interpretationen formal definieren zu können, führen wir die in Abschnitt 6.6.1 spezifizierten Begriffe auch für GERDF-Interpretationen ein. Dazu erweitern wir nun das Vokabular einer ERDF-Ontologie wie folgt:

Definition 7.16 (G-Vokabular einer ERDF-Ontologie)
Sei $O = \langle G, P \rangle$ eine ERDF-Ontologie über V_R und Var_{RDF}. Das *G-Vokabular* V_O^G der ERDF-Ontologie O ist definiert als:
$V_O^G = V_R(sk(G)) \cup V_P \cup V_{RDF} \cup V_{RDFS} \cup V_{ERDF} \cup V_{GERDF}$.

Die Menge V_P bezeichne dabei die Menge der RDF URI-Referenzen und RDF-Literale von P.

□

Das G-Vokabular einer ERDF-Ontologie fügt also dem Vokabular einer ERDF-Ontologie das GERDF-Vokabular hinzu.

Des Weiteren bezeichne die Menge $\widehat{Res_O^H}$ die Vereinigung von V_O^G ohne die wohlgeformten XML-Literale mit der Menge der XML-Werte der wohlgeformten XML-Literale aus V_O^G.

Definition 7.17 (GERDF-Herbrand-Interpretation einer ERDF-Ontologie)
Es sei $O = \langle G, P \rangle$ eine ERDF-Ontologie über V_R und Var_{RDF}. Des Weiteren sei $I \in \mathcal{IB}_{GERDF}(V_R, Var_{RDF})$ eine GERDF-Interpretation über V_R und Var_{RDF} mit $I = \langle \mathcal{S}, intU^P, intL^P, intV^P \rangle$ und der zugehörigen semantischen GERDF-Struktur $\mathcal{S} = \langle \langle \langle Res, Prop, LV, rel^+, rel^- \rangle, Kl, TKl, TProp, Kext^+, Kext^- \rangle, CKl, CProp \rangle$.
Die Interpretation I ist eine *GERDF-Herbrand-Interpretation* von O, falls die folgenden Bedingungen erfüllt sind:

1. $Res = \widehat{Res_O^H}$
2. $intU^P(u) = u$ für alle RDF URI-Referenzen $u \in V_O^G$.
3. $intL^P(l) = l$ für alle getypten RDF-Literale $l \in V_O^G$ die keine wohlgeformten XML-Literale sind.
4. $intL^P(l)$ ist der XML-Wert von l, falls l ein wohlgeformtes XML-Literal aus V_O^G ist.

Die Menge aller GERDF-Herbrand-Interpretationen der ERDF-Ontologie O kennzeichnen wir mit $\mathcal{IB}_{GERDF}^H(O)$.

□

Definition 7.18 (GERDF-Herbrand-Modell einer ERDF-Ontologie)
Sei $O = \langle G, P \rangle$ eine ERDF-Ontologie über V_R und Var_{RDF}.
Eine GERDF-Herbrand-Interpretation $I \in \mathcal{IB}_{GERDF}^H(O)$ der ERDF-Ontologie O ist ein *GERDF-Herbrand-Modell* von O genau dann, wenn $I \models \langle sk(G), P \rangle$.
Für die Menge aller GERDF-Herbrand-Modelle von O vereinbaren wir die Bezeichnung $Mod_{GERDF}^H(O)$.

□

Da die beiden Mengen CKl_I und $CProp_I$ einer GERDF-Interpretation I eindeutig durch die positiven Klassenextensionen von int_I^*(`gerdf:CoherentClass`) und int_I^*(`gerdf:CoherentProperty`) festgelegt werden, ist das Lemma 6.29 auch für GERDF-Herbrand-Interpretationen gültig.

Als nächstes übertragen wir die Ordnungsrelation $\preceq$ auf die Menge der GERDF-Herbrand-Interpretationen.

Definition 7.19 (Ordnung zwischen GERDF-Herbrand-Interpretationen)
Es sei $O = \langle G, P \rangle$ eine ERDF-Ontologie über V_R und Var_{RDF}. Des Weiteren seien $I, K \in \mathcal{IB}^H_{GERDF}(O)$ zwei GERDF-Herbrand-Interpretationen von O.
Die Interpretation K *erweitert* die Interpretation I genau dann, wenn:

1. $Prop_I \subseteq Prop_K$,
2. $rel^+_I(p) \subseteq rel^+_K(p)$ und $rel^-_I(p) \subseteq rel^-_K(p)$ für alle $p \in Prop_I$ und
3. $intV^P_I(x) = intV^P_K(x)$ für alle $x \in Var_{RDF}$.

Falls K die Interpretation I erweitert, dann schreiben wir dafür $I \preceq_G K$.
□

Das Lemma 6.32 ist auf die Relation $\preceq_G$ übertragbar, so dass $\preceq_G$ eine Halbordnung auf der Menge $\mathcal{IB}^H_{GERDF}(O)$ ist.

Unter Verwendung der Relation $\preceq_G$ werden nachfolgend die minimalen GERDF-Herbrand-Interpretationen sowie die minimalen GERDF-Herbrand-Modelle einer ERDF-Ontologie definiert.

Definition 7.20 (minimale GERDF-Herbrand-Interpretationen einer ERDF-Ontologie)
Es sei $O = \langle G, P \rangle$ eine ERDF-Ontologie über V_R und Var_{RDF}. Weiter sei $\mathcal{I} \subseteq \mathcal{IB}^H_{GERDF}(O)$ eine Menge von GERDF-Herbrand-Interpretationen.
Die Menge der *minimalen GERDF-Herbrand-Interpretationen* von $\mathcal{I}$ ist definiert als:
$Min(\mathcal{I}) = \{I \in \mathcal{I} \mid \nexists K \in \mathcal{I} : K \neq I \text{ und } K \preceq_G I\}$.
□

Definition 7.21 (minimale GERDF-Herbrand-Modelle einer ERDF-Ontologie)
Es sei $O = \langle G, P \rangle$ eine ERDF-Ontologie über V_R und Var_{RDF}.
Die Menge der *minimalen GERDF-Herbrand-Modelle* von O definieren wir wie folgt:
$Mod^{min}_{GERDF}(O) = Min\left(Mod^H_{GERDF}(O)\right)$.
□

7.5.2 Stabil erzeugte GERDF-Modelle

Nachdem wir die Herbrand-Interpretationen für GERDF-Interpretationen eingeführt haben, werden in diesem Abschnitt die stabil erzeugten ERDF-Modelle zu stabil erzeugten GERDF-Modellen erweitert.

Definition 7.22 (stabil erzeugtes GERDF-Modell einer ERDF-Ontologie)
Sei $O = \langle G, P \rangle$ eine ERDF-Ontologie über V_R und Var_{RDF}.
Weiterhin sei $I \in \mathcal{IB}^H_{GERDF}(O)$ eine GERDF-Herbrand-Interpretation von O.
Die Interpretation I ist ein *stabil erzeugtes GERDF-Modell* von O genau dann, wenn eine Kette von GERDF-Herbrand-Interpretationen $I_0 \preceq_G I_1 \preceq_G \ldots \preceq_G I_{k+1}$ mit $I_j \in \mathcal{IB}^H_{GERDF}(O)$ für $0 \leq j \leq k+1$ existiert, welche die folgenden Bedingungen erfüllt:

1. $I_k = I_{k+1} = I$
2. $I_0 \in Min\left(\left\{J \in \mathcal{IB}^H_{GERDF}(O) \;\middle|\; J \models sk(G)\right\}\right)$
3. Für jede Zahl n mit $0 < n \leq k+1$ gilt:
   $I_n \in Min\left(\left\{J \in \mathcal{IB}^H_{GERDF}(O) \;\middle|\; I_{n-1} \preceq_G J \text{ und } J \models Kopf(r), \forall r \in P[I_{n-1}, I]\right\}\right)$,
   wobei die Menge $P[I_{n-1}, I]$ wie folgt definiert ist:
   $P[I_{n-1}, I] = \left\{r \in [P]_{V^G_O} \middle| J \models Rumpf(r), \forall J \in \mathcal{IB}^H_{GERDF}(O) \text{ mit } I_{n-1} \preceq_G J \preceq_G I\right\}$. □

Für die Menge der stabil erzeugten GERDF-Modelle von O vereinbaren wir die Notation $Mod^{st}_{GERDF}(O)$.

Sehen wir uns nun die ERDF-Ontologie O aus Beispiel 6.39 noch einmal an. Dann ist die Beobachtung $M \models \sim$ `rdf:type` (`ex:Tweety`, `ex:KannFliegen`) und $M \models \neg$`rdf:type`(`ex:Tweety`,`ex:KannFliegen`) nicht mehr für alle $M \in Mod^{st}_{GERDF}(O)$ gültig. Infolge der fehlenden Kohärenz existieren stabil erzeugte GERDF-Modelle $M \in Mod^{st}_{GERDF}(O)$ von O mit $M \models$ `rdf:type` (`ex:Tweety`, `ex:KannFliegen`) und $M \models \neg$`rdf:type` (`ex:Tweety`, `ex:KannFliegen`).

Begründung:

Für alle Interpretationen $I \in Min\left(\left\{J \in \mathcal{IB}^H_{GERDF}(O) \;\middle|\; J \models sk(G)\right\}\right)$ gilt entweder
$I \models$ `rdf:type` (`ex:Tweety`, `ex:KannFliegen`) oder
$I \models \neg$`rdf:type` (`ex:Tweety`, `ex:KannFliegen`).
Sei nun $I_0 \in Min\left(\left\{J \in \mathcal{IB}^H_{GERDF}(O) \;\middle|\; J \models sk(G)\right\}\right)$ mit
$I_0 \models \neg$`rdf:type` (`ex:Tweety`, `ex:KannFliegen`). Dann gilt $P[I_0, M] = P[M, M] =$ {¬`rdf:type`(`ex:Tweety`, `ex:KannFliegen`) ⇐ `rdf:type`(`ex:Tweety`, `ex:Pinguin`)}.
Also wird M durch die Kette $I_0 \preceq_G M \preceq_G M$ erzeugt.
Sei nun $I_0 \in Min\left(\left\{J \in \mathcal{IB}^H_{GERDF}(O) \;\middle|\; J \models sk(G)\right\}\right)$ mit
$I_0 \models$ `rdf:type` (`ex:Tweety`, `ex:KannFliegen`). Dann gilt $P[I_0, M] = P[M, M] =$ {¬`rdf:type`(`ex:Tweety`, `ex:KannFliegen`) ⇐ `rdf:type`(`ex:Tweety`, `ex:Pinguin`)}.
Also wird M durch die Kette $I_0 \preceq_G M \preceq_G M$ erzeugt, wobei für M gilt:
$M \models \neg$`rdf:type` (`ex:Tweety`, `ex:KannFliegen`) und
$M \models$ `rdf:type` (`ex:Tweety`, `ex:KannFliegen`).
Dieses Beispiel zeigt, dass in stabil erzeugten GERDF-Modellen auch widersprüchliche Informationen möglich sind.

7.5.3 Minimal inkonsistente stabil erzeugte GERDF-Modelle

Im vorangegangenen Abschnitt haben wir gesehen, dass ERDF-Ontologien inkonsistente stabil erzeugte GERDF-Modelle besitzen. Diese treten auch dann auf, wenn die betreffenden ERDF-Ontologien wie in Beispiel 6.39 keine Widersprüche beinhalten und somit auch konsistente stabil erzeugte GERDF-Modelle besitzen. Da

wir Inkonsistenzen durch Aufhebung der Kohärenzbedingung ausdrücklich erlauben, aber die Anzahl der widersprüchlichen Informationen möglichst gering halten wollen, schränken wir die stabil erzeugten GERDF-Modelle einer ERDF-Ontologie nach dem Vorbild von Graham Priest [58] weiter ein. Die so erzeugten Modelle werden als minimal inkonsistente stabil erzeugte GERDF-Modelle bezeichnet.

Als erstes definieren wir den Begriff der Inkonsistenz für GERDF-Herbrand-Interpretationen.

Definition 7.23 (Inkonsistenz einer GERDF-Herbrand-Interpretation)
Sei $O = \langle G, P \rangle$ eine ERDF-Ontologie über V_R und Var_{RDF}.
Weiterhin sei $I \in \mathcal{IB}^H_{GERDF}(O)$ eine GERDF-Herbrand-Interpretation von O.
Die *Inkonsistenz* $Ink(I)$ der Interpretation I wird folgendermaßen definiert:
$Ink(I) = \{p(s,o) \in Tri_{ERDF}(V_R, Var_{RDF}) \mid I \models p(s,o) \text{ und } I \models \neg p(s,o)\}$.
□

Falls I eine kohärente GERDF-Herbrand-Interpretation ist, dann gilt $Ink(I) = \emptyset$.

Der nächste Schritt besteht darin, die GERDF-Herbrand-Interpretationen einer ERDF-Ontologie anhand ihrem Widersprüchlichkeitsgrad zu ordnen.

Definition 7.24 (Inkonsistenzordnung zwischen GERDF-Herbrand-Interpretationen)
Sei $O = \langle G, P \rangle$ eine ERDF-Ontologie über V_R und Var_{RDF}. Weiterhin seien $I, K \in \mathcal{IB}^H_{GERDF}(O)$ zwei GERDF-Herbrand-Interpretationen von O.
Die Interpretation K ist *widersprüchlicher* als die Interpretation I genau dann, wenn $Ink(I) \subseteq Ink(K)$.
Derartige Fälle werden mit der Notation $I \preceq_{ink} K$ gekennzeichnet.
□

Unter Verwendung der Relation $\preceq_{ink}$ sind wir nun in der Lage die minimal inkonsistenten GERDF-Herbrand-Interpretationen sowie die minimal inkonsistenten GERDF-Herbrand-Modelle einer ERDF-Ontologie einzuführen.

Definition 7.25 (minimal inkonsistente GERDF-Herbrand-Interpretation)
Es sei $O = \langle G, P \rangle$ eine ERDF-Ontologie über V_R und Var_{RDF}. Weiterhin sei $\mathcal{I} \subseteq \mathcal{IB}^H_{GERDF}(O)$ eine Menge von GERDF-Herbrand-Interpretationen.
Die Menge der *minimal inkonsistenten GERDF-Herbrand-Interpretationen* von $\mathcal{I}$ ist definiert als:
$Min_{ink}(\mathcal{I}) = \{I \in \mathcal{I} \mid \nexists K \in \mathcal{I} : Ink(K) \neq Ink(I) \text{ und } K \preceq_{ink} I\}$.
□

Definition 7.26 (minimal inkonsistentes GERDF-Herbrand-Modell)
Es sei $O = \langle G, P \rangle$ eine ERDF-Ontologie über V_R und Var_{RDF}.
Die Menge der *minimal inkonsistenten GERDF-Herbrand-Modelle* von O definieren wir wie folgt:
$Mod^{INKmin}_{GERDF}(O) = Min_{ink}\left(Mod^H_{GERDF}(O)\right)$.
□

Nachfolgend werden nun die minimal inkonsistenten stabil erzeugten GERDF-Modelle einer ERDF-Ontologie festgelegt.

Definition 7.27 (minimal inkonsistentes stabil erzeugtes GERDF-Modell)
Es sei $O = \langle G, P \rangle$ eine ERDF-Ontologie über V_R und Var_{RDF}. Weiterhin sei $I \in \mathcal{IB}^H_{GERDF}(O)$ eine GERDF-Herbrand-Interpretation von O.
Die Interpretation I ist ein *minimal inkonsistentes stabil erzeugtes GERDF-Modell* von O genau dann, wenn eine Kette $I_0 \preceq_G I_1 \preceq_G \ldots \preceq_G I_{k+1}$ von GERDF-Herbrand-Interpretationen von O mit $I_j \in \mathcal{IB}^H_{GERDF}(O)$ für $0 \leq j \leq k+1$ existiert und die folgenden Bedingungen erfüllt werden:

1. $I \in Min_{ink}\left(Mod^{st}_{GERDF}\right)$
2. $I_k = I_{k+1} = I$
3. $I_0 \in Min\left(\left\{ I \in \mathcal{IB}^H_{GERDF}(O) \;\middle|\; I \models sk(G) \right\}\right)$
4. Für jede Zahl n mit $0 < n \leq k+1$ gilt:

$$I_n \in Min\left(\left\{ J \in \mathcal{IB}^H_{GERDF}(O) \;\middle|\; I_{n-1} \preceq_G J, Ink(J) \subseteq Ink(I) \text{ und } J \models Kopf(r)\, \forall r \in P_{[I_{n-1},I]} \right\}\right),$$

wobei die Menge $P[I_{n-1}, I]$ wie folgt definiert ist:
$P[I_{n-1}, I] = \left\{ r \in [P]_{V^G_O} \;\middle|\; J \models Rumpf(r), \forall J \in \mathcal{IB}^H_{GERDF}(O) \text{ mit } I_{n-1} \preceq_G J \preceq_G I \right\}$. □

Die Menge der minimal inkonsistenten stabil erzeugten GERDF-Modelle von O bezeichnen wir mit $Mod^{mInk\ st}_{GERDF}(O)$.

Beobachtung 7.28
Sei O die ERDF-Ontologie aus Beispiel 6.39. Dann gilt für alle $M \in Mod^{mInk\ st}_{GERDF}(O)$:
$M \models \sim$`rdf:type`$($`ex:Tweety`,`ex:KannFliegen`$)$ *und*
$M \models \neg$`rdf:type`$($`ex:Tweety`,`ex:KannFliegen`$)$.

Beweis:
Die stabil erzeugten GERDF-Modelle $\widehat{M} \in Mod^{st}_{GERDF}(O)$ mit
$\widehat{M} \models$ `rdf:type`$($`ex:Tweety`,`ex:KannFliegen`$)$ und
$\widehat{M} \models \neg$`rdf:type`$($`ex:Tweety`,`ex:KannFliegen`$)$
erfüllen nicht die Eigenschaft der minimalen Inkonsistenz.

■

Dieses Resultat lässt sich für beliebige ERDF-Ontologien verallgemeinern. Sei O eine ERDF-Ontologie über V_R und Var_{RDF}. Dann gilt offensichtlich $Ink(M) = \emptyset$ für alle $M \in Mod^{mInk\ st}_{GERDF}(O)$, falls ein $\widehat{M} \in Mod^{st}_{GERDF}(O)$ existiert mit $Ink\left(\widehat{M}\right) = \emptyset$.

Für ERDF-Ontologien die keine ERDF-Regeln beinhalten, sondern ausschließlich aus ERDF-Tripeln bestehen, sind alle stabil erzeugten GERDF-Modelle auch minimal inkonsistente stabil erzeugte GERDF-Modelle.

Lemma 7.29
Sei $O = \langle G, P \rangle$ eine ERDF-Ontologie über V_R und Var_{RDF} mit $P = \emptyset$.
Dann gilt $Mod^{mInk\,st}_{GERDF}(O) = Mod^{st}_{GERDF}(O)$.

Beweis:
" $\subseteq$ " Diese Inklusion gilt nach Definition 7.27.
" $\supseteq$ " Sei $M \in Mod^{st}_{GERDF}(O)$. Dann ist $M \in Mod^{mInk\,st}_{GERDF}(O)$ zu zeigen. Angenommen es gilt $M \notin Mod^{mInk\,st}_{GERDF}(O)$. Da $M \in Mod^{st}_{GERDF}(O)$ existiert ein $\widehat{M} \in Mod^{mInk\,st}_{GERDF}(O)$ mit $\widehat{M} \preceq_{ink} M$ und $Ink\left(\widehat{M}\right) \neq Ink(M)$. Also $Ink\left(\widehat{M}\right) \subsetneq Ink(M)$. Demzufolge existiert eine Property $p \in Prop_M$ für die entweder (i) $p \notin Prop_{\widehat{M}}$ oder (ii) $rel^+_{\widehat{M}}(p) \subsetneq rel^+_M(p)$ oder (iii) $rel^-_{\widehat{M}}(p) \subsetneq rel^-_M(p)$ gilt. Also $\widehat{M} \neq M$ und $\widehat{M} \preceq_G M$. Damit folgt $M \notin Min\left(\left\{I \in \mathcal{IB}^H_{GERDF}(O) \;\middle|\; I \models sk(G)\right\}\right)$. Also $M \notin Mod^{st}_{GERDF}(O)$. Dies ist ein Widerspruch. Also ist die Annahme falsch und damit gilt $Mod^{mInk\,st}_{GERDF}(O) \supseteq Mod^{st}_{GERDF}(O)$.

■

Abschließend betrachten wir die Folgerungsrelationen, welche durch die stabil erzeugten GERDF-Modelle und die minimal inkonsistenten stabil erzeugten GERDF-Modelle impliziert werden.

Definition 7.30 (Folgerungsrelationen)
Sei $O = \langle G, P \rangle$ eine ERDF-Ontologie über V_R und Var_{RDF}. Weiterhin sei F ein ERDF-Graph oder eine ERDF-Formel über V_R und Var_{RDF}.

(a) Es gilt $O \models^{st}_{GERDF} F$ genau dann, wenn $I \models F$ für alle $I \in Mod^{st}_{GERDF}(O)$.

(b) Es gilt $O \models^{mInk\,st}_{GERDF} F$ genau dann, wenn $I \models F$ für alle $I \in Mod^{mInk\,st}_{GERDF}(O)$. □

Lemma 7.31
Die Folgerungsrelationen $\models^{st}_{GERDF}$ und $\models^{mInk\,st}_{GERDF}$ sind weder monoton noch kumulativ.

Beweis:
Der Beweis erfolgt mithilfe eines Gegenbeispiels, welches auf der Beweisidee von Observation 18 [42] basiert.
Seien $\texttt{ex:s}, \texttt{ex:o}, \texttt{ex:a}, \texttt{ex:b}, \texttt{ex:c} \in URI_{RDF}$ RDF URI-Referenzen.
Weiterhin sei $O = \langle G, P \rangle$ eine ERDF-Ontologie mit $G = \emptyset$ und

$$\begin{aligned} P = \{\ & \texttt{ex:b}(\texttt{ex:s},\texttt{ex:o}) \Leftarrow {\sim}\texttt{ex:c}(\texttt{ex:s},\texttt{ex:o}), \\ & \texttt{ex:c}(\texttt{ex:s},\texttt{ex:o}) \Leftarrow {\sim}\texttt{ex:b}(\texttt{ex:s},\texttt{ex:o}), \\ & \texttt{ex:a}(\texttt{ex:s},\texttt{ex:o}) \Leftarrow {\sim}\texttt{ex:a}(\texttt{ex:s},\texttt{ex:o}), \\ & \texttt{ex:a}(\texttt{ex:s},\texttt{ex:o}) \Leftarrow {\sim}\texttt{ex:c}(\texttt{ex:s},\texttt{ex:o})\ \}. \end{aligned}$$

Dann gilt $Mod^{st}_{GERDF}(O) = Mod^{mInk\,st}_{GERDF}(O) = \{M\}$ mit
$M \models \texttt{ex:a}(\texttt{ex:s},\texttt{ex:o}) \wedge\ \texttt{ex:b}(\texttt{ex:s},\texttt{ex:o}) \wedge {\sim}\texttt{ex:c}(\texttt{ex:s},\texttt{ex:o})$.

Also $O \models^{st}_{GERDF} \texttt{ex:a}\,(\texttt{ex:s}, \texttt{ex:o})$ und $O \models^{st}_{GERDF} \texttt{ex:b}\,(\texttt{ex:s}, \texttt{ex:o})$.
Analog dazu gilt $O \models^{mInk\ st}_{GERDF} \texttt{ex:a}\,(\texttt{ex:s}, \texttt{ex:o})$ und $O \models^{mInk\ st}_{GERDF} \texttt{ex:b}\,(\texttt{ex:s}, \texttt{ex:o})$.
Sei nun $\widehat{O} = \left\langle \widehat{G}, P \right\rangle$ mit $\widehat{G} = \{\texttt{ex:a}\,(\texttt{ex:s}, \texttt{ex:o})\}$.
Dann gilt $Mod^{st}_{GERDF}\left(\widehat{O}\right) = Mod^{mInk\ st}_{GERDF}\left(\widehat{O}\right) = \{M_1, M_2\}$ mit
$M_1 \models \texttt{ex:a}\,(\texttt{ex:s}, \texttt{ex:o}) \wedge \texttt{ex:b}\,(\texttt{ex:s}, \texttt{ex:o}) \wedge \sim \texttt{ex:c}\,(\texttt{ex:s}, \texttt{ex:o})$ und
$M_2 \models \texttt{ex:a}\,(\texttt{ex:s}, \texttt{ex:o}) \wedge \sim \texttt{ex:b}\,(\texttt{ex:s}, \texttt{ex:o}) \wedge \texttt{ex:c}\,(\texttt{ex:s}, \texttt{ex:o})$.
Also $\widehat{O} \not\models^{st}_{GERDF} \texttt{ex:b}\,(\texttt{ex:s}, \texttt{ex:o})$ und $\widehat{O} \not\models^{mInk\ st}_{GERDF} \texttt{ex:b}\,(\texttt{ex:s}, \texttt{ex:o})$.

■

Dieses Lemma besagt, dass durch Hinzunahme von zusätzlichen Informationen die bisherigen Folgerungen einer ERDF-Ontologie ungültig werden können. Ein derartiger Fall ist selbst dann möglich, wenn die ERDF-Ontologie mit Informationen der Folgerungsmenge erweitert wird.

8 Zusammenfassung und Ausblick

Dieses Kapitel gibt zunächst einen Überblick über die Resultate der Diplomarbeit. Anschließend stellen wir einige interessante Aspekte vor, die attraktive Thematiken für zukünftige Arbeiten darstellen, um auf die hier gewonnenen Ergebnisse aufzubauen und diese zu erweitern.

8.1 Zusammenfassung der Ergebnisse

Die wichtigsten Resultate der Diplomarbeit werden in diesem Abschnitt noch einmal überblicksartig zusammengefasst.

Die ersten Ergebnisse werden in Kapitel vier erzielt. Nachdem wir die Syntax von Common Logic eingeführt und mit der prädikatenlogischen Syntax in Beziehung gesetzt haben, wurde die offizielle Semantik von Common Logic leicht modifiziert und gezeigt, dass diese Abwandlung die Wohldefiniertheit einer Retraktion und damit der Semantik eines CL-Moduls garantiert. In der offiziellen CL-Semantik ist dies nicht der Fall.

Anschließend wurden die Eigenschaften von Common Logic genau untersucht. Dabei haben wir festgestellt, dass sich segregierte CL-Dialekte und unsegregierte CL-Dialekte nicht nur syntaktisch, sondern auch semantisch voneinander unterscheiden. Anhand eines Beispiels wurde gezeigt, dass CL-Formeln existieren, die in keinem unsegregierten CL-Dialekt erfüllbar sind, aber Modelle in segregierten CL-Dialekten besitzen.

Die Semantik der CL-Sequenznamen wurde ebenfalls in dieser Diplomarbeit untersucht. Wir haben dabei bewiesen, dass mithilfe von allquantifizierten CL-Sequenznamen unendliche Konjunktionen möglich sind. Analog dazu erlauben existenzquantifizierte CL-Sequenznamen die Formulierung von unendlichen Disjunktionen.

Unter Verwendung dieser Resultate haben wir gezeigt, dass die Endlichkeit des Gegenstandsbereichs einer CL-Interpretation mithilfe einer CL-Formel gefordert werden kann. Eine Konsequenz dieses Ergebnisses ist die Tatsache, dass der Endlichkeitssatz in Common Logic nicht gültig ist und die CL-Folgerungsrelation damit nicht kompakt ist.

Die syntaktischen Freiheiten von Common Logic wurden ebenfalls untersucht und mittels zahlreicher Beispiele illustriert. Es wurde festgestellt, dass Common Logic die Selbstreferenz erlaubt und dadurch die Formulierung von Antinomi-

en zulässt. Als Beispiel haben wir die Russellsche Klasse in Common Logic definiert und deren Semantik untersucht. Es wurde gezeigt, dass diese Charakterisierung der Russellschen Klasse in allen unsegregierten CL-Dialekten unerfüllbar ist. Unter bestimmten Bedingungen existieren jedoch auch hier segregierte CL-Interpretationen, die diese CL-Formel erfüllen.

Zusätzlich wurden in Kapitel vier weitere Eigenschaften von Common Logic wie die Ungültigkeit des Extensionalitätsaxioms sowie die Existenz zusätzlicher Tautologien diskutiert[25].

Im fünften Kapitel haben wir die Ontologiesprachen RDF und RDFS behandelt. Nachdem die Eigenschaft der Reifikation ausführlich in Abschnitt 5.4 thematisiert wurde, haben wir gezeigt, dass sich RDF-Dokumente und RDFS-Dokumente unter Erhaltung ihrer Semantik nach Common Logic übertragen lassen. Dabei haben wir auf die Arbeiten von Patrick Hayes [36] und Christopher Menzel [53] zurückgegriffen.

Im nächsten Kapitel wurde diese Einbettung auf die ERDF-Interpretationen erweitert, so dass auch ERDF-Dokumente nach Common Logic übertragen werden können. Im letzten Teil des sechsten Kapitels haben wir die in [5] entwickelte Semantik der stabil erzeugten ERDF-Modelle vorgestellt, die eine Verfeinerung der ERDF-Interpretationen darstellt. Dabei wurden einige Resultate bewiesen, die in [5] nur angedeutet werden.

Aufgrund der globalen Kohärenzbedingung lässt ERDF keine widersprüchlichen Informationen zu. Daher schlagen wir in Kapitel sieben eine Erweiterung vor, die auch die Modellierung von inkonsistenten Ontologien erlaubt. Zunächst wurden die GERDF-Interpretationen eingeführt, die auf den ERDF-Interpretationen basieren. Im Anschluss daran haben wir gezeigt, dass die Folgerungsrelation, welche durch die GERDF-Interpretationen induziert wird, zu der in Definition 6.20 eingeführten kERDF-Folgerungsrelation für RDF-Graphen äquivalent ist, falls die betrachteten RDF-Graphen kein GERDF-Vokabular enthalten. Dadurch stellt die GERDF-Folgerungsrelation eine konservative Erweiterung der RDFS-Folgerungsrelation dar.

Abschließend wurden die stabil erzeugten ERDF-Modelle auf GERDF übertragen und zu den minimal inkonsistenten stabil erzeugten GERDF-Modellen verfeinert. In diesem Zusammenhang haben wir bewiesen, dass die Folgerungsrelation der stabil erzeugten GERDF-Modelle sowie die Folgerungsrelation der minimal inkonsistenten stabil erzeugten GERDF-Modelle weder monoton noch kumulativ sind. Darüber hinaus wurde gezeigt, dass diese beiden Semantiken zusammenfallen, falls ERDF-Ontologien lediglich aus ERDF-Graphen bestehen und keine ERDF-Regeln beinhalten.

[25] Das Wort zusätzlich bedeute an dieser Stelle, dass vergleichbare Tautologien in der Prädikatenlogik der ersten Stufe aufgrund der syntaktischen Einschränkungen nicht formulierbar sind.

8.2 Ausblick

Aufgrund der Tatsache, dass sowohl Common Logic als auch semantisch ähnliche Sprachen wie IKL [40] oder SKIF [38] noch sehr jung sind, existieren derzeit zahlreiche offene Problemstellungen, von denen einige bereits in Kapitel 4.9 angesprochen wurden. Wir wollen in diesem Abschnitt auf weitere offene Fragen eingehen, welche Themen für zukünftige Untersuchungen bereitstellen.

Abgesehen von dem offiziellen ISO-Standard [48] existieren nur sehr wenige Publikationen zu Common Logic. So wurde bisher kein Handbuch zu Common Logic veröffentlicht, in dem beispielsweise die Gültigkeit wichtiger Axiome, wie dem in Kapitel 4.9 angesprochenem Extensionalitätsaxiom (4.45), diskutiert oder konkrete Anwendungsfälle modelliert werden. Ein Teil dieser Thematik wird im IKL Guide [39] behandelt, so dass dieses Dokument einen vielversprechenden Ausgangspunkt für eine umfangreiche Common Logic Dokumentation darstellt.

Eine weitere Aufgabe ist die Erarbeitung einer Axiomatisierung für Common Logic bzw. für Teilsprachen von Common Logic. Ein erster Schritt in diese Richtung beinhaltet die Überprüfung der prädikatenlogischen Axiome[26] um herauszufinden, ob diese auch in Common Logic allgemeingültig sind.

Darüber hinaus existiert derzeit keine Beweistheorie für Common Logic. Effiziente Theorembeweiser sind bisher ebenfalls nicht verfügbar, so dass im Regelfall prädikatenlogische Theorembeweiser verwendet werden, die jedoch eine Übertragung der CL-Dokumente in die Prädikatenlogik erfordern.

Das ursprüngliche Ziel der Common Logic Initiative war die Entwicklung eines Standards, der als Austausch- und Übertragungsschicht zwischen verschiedenen logischen Sprachen fungieren und die semantische Interoperabilität dieser Systeme gewährleisten soll [20]. Aufgrund dieses Vorhabens wurde Common Logic mit einer sehr hohen Ausdruckskraft ausgestattet, um möglichst viele logische Systeme nach Common Logic übersetzen zu können. Da Common Logic zusätzlich die Verwendung von URIs erlaubt, lassen sich sogar die komplexen Semantic Web Sprachen RDFS und OWL nach Common Logic übertragen[27]. Daher ist Common Logic ein möglicher Kandidat für die Realisierung der logischen Schicht des Semantic Web. Allerdings besitzt Common Logic eine monotone Semantik, so dass die Repräsentation nichtmonotoner Regelformalismen, wie für RIF geplant, derzeit nicht möglich ist. Die Reifikation wird in Common Logic ebenfalls nicht unterstützt. Die Entwicklung eines CL-Dialekts, welcher die Standardsemantik von Common Logic erweitert, um diese Defizite zu beseitigen, ist demzufolge eine lohnenswerte Aufgabe für die nahe Zukunft.

Selbstverständlich bietet auch Generalized Extended RDF zahlreiche Thematiken für zukünftige Arbeiten. Einerseits existieren Ansatzpunkte für theoretische

[26] Dieses Axiomensystem ist beispielsweise in [29, Seite 69] angegeben.
[27] Eine Übersetzung von OWL nach Common Logic wurde von Patrick Hayes [36] ausgearbeitet.

Untersuchungen wie beispielsweise die Analyse der Beziehung zwischen den minimal inkonsistenten stabil erzeugten GERDF-Modellen und den stabil erzeugten ERDF-Modellen oder der Vergleich mit anderen parakonsistenten Semantiken. Andererseits sind auch praktische Arbeiten von Interesse. So ist zum Beispiel die Implementierung einer Inferenzmaschine nach dem Vorbild des ERDF-Prototypen[28] von Bedeutung. Die Erweiterung des GERDF-Vokabulars ist ebenfalls denkbar, um zusätzliche Funktionen wie Kommentare oder eine Importmöglichkeit für andere Ontologien bereitzustellen.

[28] http://oxygen.informatik.tu-cottbus.de/JenaRulesWeb/

Literaturverzeichnis

[1] *Common Logic Standardization Meeting*. Verfügbar unter: http://cl.tamu.edu/minutes/stanford-minutes.html, März 2002.
Zuletzt besucht am: 23. März 2009.

[2] ALFERES, JOSÉ JÚLIO, HEINRICH HERRE und LUÍS MONIZ PEREIRA: *Partial Models of Extended Generalized Logic Programs*. In: LLOYD, JOHN W., VERÓNICA DAHL, ULRICH FURBACH, MANFRED KERBER, KUNG-KIU LAU, CATUSCIA PALAMIDESSI, LUÍS MONIZ PEREIRA, YEHOSHUA SAGIV und PETER J. STUCKEY (Herausgeber): *CL '00: Proceedings of the First International Conference on Computational Logic, London, UK, July 24-28, 2000*, Band 1861 der Reihe *Lecture Notes in Computer Science*, Seiten 149–163. Springer-Verlag, 2000.

[3] ANALYTI, ANASTASIA, GRIGORIS ANTONIOU, CARLOS VIEGAS DAMÁSIO und GERD WAGNER: *Negation and Negative Information in the W3C Resource Description Framework*. Annals of Mathematics, Computing & Teleinformatics, 1(2):25–34, 2004.

[4] ANALYTI, ANASTASIA, GRIGORIS ANTONIOU, CARLOS VIEGAS DAMÁSIO und GERD WAGNER: *Extended RDF as a Semantic Foundation of Rule Markup Languages*. Technischer Bericht I1-D3-2, REWERSE, 2005.

[5] ANALYTI, ANASTASIA, GRIGORIS ANTONIOU, CARLOS VIEGAS DAMÁSIO und GERD WAGNER: *Extended RDF as a Semantic Foundation of Rule Markup Languages*. Journal of Artificial Intelligence Research, 32:37–94, 2008.

[6] ANALYTI, ANASTASIA, GRIGORIS ANTONIOU, CARLOS VIEGAS DAMÁSIO und GERD WAGNER: *On the Computability and Complexity Issues of Extended RDF*. In: HO, TU BAO und ZHI-HUA ZHOU (Herausgeber): *PRICAI 2008: Trends in Artificial Intelligence, Proceedings of the 10th Pacific Rim International Conference on Artificial Intelligence, Hanoi, Vietnam, December 15-19, 2008*, Band 5351 der Reihe *Lecture Notes in Computer Science*, Seiten 5–16. Springer-Verlag, 2008.

[7] ANTONIOU, GRIGORIS, CARLOS VIEGAS DAMÁSIO, BENJAMIN GROSOF, IAN HORROCKS, MICHAEL KIFER, JAN MALUSZYNSKI und PETER F. PATEL-SCHNEIDER: *Combining Rules and Ontologies. A survey.* Technischer Bericht I3-D3, REWERSE, März 2005.

[8] BARAL, CHITTA und MICHAEL GELFOND: *Logic Programming and Knowledge Representation*. Journal of Logic Programming, 19/20:73–148, 1994.

[9] BECKETT, DAVE: *RDF/XML Syntax Specification (Revised)*. Technischer Bericht, World Wide Web Consortium (W3C), Februar 2004. W3C Recommendation. Verfügbar unter: http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/.

[10] BECKETT, DAVID: *Turtle - Terse RDF Triple Language*. Technischer Bericht, November 2007. Verfügbar unter http://www.dajobe.org/2004/01/turtle/. Zuletzt besucht am: 18. April 2009.

[11] BELL, JOHN L.: *Infinitary Logic*. In: ZALTA, EDWARD N. (Herausgeber): *The Stanford Encyclopedia of Philosophy*. Stanford University, Spring 2009 Edition, März 2009. Verfügbar unter: http://plato.stanford.edu/archives/spr2009/entries/logic-infinitary/.

[12] BERNERS-LEE, T., R. FIELDING und L. MASINTER: *RFC3986 Uniform Resource Identifier (URI): Generic Syntax*. Technischer Bericht, The Internet Society (ISOC), Januar 2005. Verfügbar unter: http://tools.ietf.org/html/rfc3986.

[13] BERNERS-LEE, TIM: *The Semantic Web*, 2002. Verfügbar unter: http://www.w3.org/2002/Talks/04-sweb/Overview.html
Zuletzt besucht am: 13. März 2009.

[14] BERNERS-LEE, TIM, JAMES HENDLER und ORA LASSILA: *The Semantic Web*. Scientific American, 284(5):34–43, Mai 2001.

[15] BOLANDER, THOMAS: *Self-Reference*. In: ZALTA, EDWARD N. (Herausgeber): *The Stanford Encyclopedia of Philosophy*. Stanford University, Spring 2009 Edition, März 2009. Verfügbar unter: http://plato.stanford.edu/archives/spr2009/entries/self-reference/.

[16] BOLEY, HAROLD und MICHAEL KIFER: *RIF Basic Logic Dialect*. Technischer Bericht, World Wide Web Consortium (W3C), Juli 2008. W3C Working Draft. Verfügbar unter: http://www.w3.org/TR/2008/WD-rif-bld-20080730/.

[17] BRAY, TIM, DAVE HOLLANDER, ANDREW LAYMAN und RICHARD TOBIN: *Namespaces in XML 1.0 (Second Edition)*. Technischer Bericht, World Wide Web Consortium (W3C), August 2006. W3C Recommendation. Verfügbar unter: http://www.w3.org/TR/2006/REC-xml-names-20060816/.

[18] BRAY, TIM, JEAN PAOLI, C. M. SPERBERG-MCQUEEN, EVE MALER und FRANÇOIS YERGEAU: *Extensible Markup Language (XML) 1.0 (Fifth Edi-*

tion). Technischer Bericht, World Wide Web Consortium (W3C), November 2008. W3C Recommendation. Verfügbar unter: http://www.w3.org/TR/2008/REC-xml-20081126/.

[19] BRICKLEY, DAN und R.V. GUHA: *RDF Vocabulary Description Language 1.0: RDF Schema*. Technischer Bericht, World Wide Web Consortium (W3C), Februar 2004. W3C Recommendation. Verfügbar unter: http://www.w3.org/TR/2004/REC-rdf-schema-20040210/.

[20] DELUGACH, HARRY S.: *Towards Conceptual Structures Interoperability Using Common Logic*. In: CROITORU, MADALINA, ROBERT JÄSCHKE und SEBASTIAN RUDOLPH (Herausgeber): *Conceptual Structures Tools and the Web 2008: Proceedings of the Third Conceptual Structures Tool Interoperability Workshop (CS-TIW-2008), Toulouse, France, July 7, 2008*, Band 352, Seiten 13–21, 2008.

[21] DIETRICH, JENS und HEINRICH HERRE: *Contributions to the Theory of Nonmonotonic Inference Systems*. Technischer Bericht Report Nr. 97-05, Institut für Informatik, Universität Leipzig, 1997.

[22] DUERST, M. und M. SUIGNARD: *RFC3987 Internationalized Resource Identifiers (IRIs)*. Technischer Bericht, The Internet Society (ISOC), Januar 2005. Verfügbar unter: http://tools.ietf.org/html/rfc3987.

[23] ENDERTON, HERBERT B.: *Second-order and Higher-order Logic*. In: ZALTA, EDWARD N. (Herausgeber): *The Stanford Encyclopedia of Philosophy*. Stanford University, Spring 2009 Edition, März 2009. Verfügbar unter: http://plato.stanford.edu/archives/spr2009/entries/logic-higher-order/.

[24] ENGELFRIET, JOERI und HEINRICH HERRE: *Stable Generated Models, Partial Temporal Logic and Disjunctive Defaults*. The Journal of Logic and Algebraic Programming, 41(1):1–25, 1999.

[25] FAHAD, MUHAMMAD, MUHAMMAD ABDUL QADIR und MUHAMMAD WAJAHAAT NOSHAIRWAN: *Semantic Inconsistency Errors in Ontology*. In: *Proceedings of the 2007 IEEE International Conference on Granular Computing, GrC 2007, San Jose, California, USA, November 2-4, 2007*, Seiten 283–286. IEEE, 2007.

[26] FIKES, RICHARD und DEBORAH MCGUINNESS: *An Axiomatic Semantics for RDF, RDF-S, and DAML+OIL (March 2001)*. Technischer Bericht, World Wide Web Consortium (W3C), Dezember 2001. W3C Note. Verfügbar unter: http://www.w3.org/TR/2001/NOTE-daml+oil-axioms-20011218.

[27] GENESERETH, MICHAEL R.: *Knowledge Interchange Format*. Technischer Bericht NCITS.T2/98-004, Stanford University, 1998. draft proposed American

National Standard (dpANS). Verfügbar unter http://logic.stanford.edu/kif/dpans.html.

[28] GOLBREICH, CHRISTINE, OLIVIER DAMERON, BERNARD GIBAUD und ANITA BURGUN: *Web Ontology Language Requirements w.r.t Expressiveness of Taxonomy and Axioms in Medicine*. In: FENSEL, DIETER, KATIA P. SYCARA und JOHN MYLOPOULOS (Herausgeber): *Proceedings of the Second International Semantic Web Conference (ISWC 2003), Sanibel Island, FL, USA, October 20-23, 2003*, Band 2870 der Reihe *Lecture Notes in Computer Science*, Seiten 180–194. Springer-Verlag, 2003.

[29] GOLTZ, HANS-JOACHIM und HEINRICH HERRE: *Grundlagen der logischen Programmierung*, Band 34 der Reihe *Informatik - Kybernetik - Rechentechnik*. Akademie-Verlag Berlin, 1990.

[30] GÓMEZ-PÉREZ, ASUNCIÓN, MARIANO FERNÁNDEZ-LÓPEZ und OSCAR CORCHO: *Ontological Engineering*, Kapitel 1: *Theoretical Foundations of Ontologies*, Seiten 1–45. Advanced Information and Knowledge Processing. Springer-Verlag, 1. Auflage, 2004.

[31] GÓMEZ-PÉREZ, ASUNCIÓN, MARIANO FERNÁNDEZ-LÓPEZ und OSCAR CORCHO: *Ontological Engineering*, Kapitel 2: *The Most Outstanding Ontologies*, Seiten 47–106. Advanced Information and Knowledge Processing. Springer-Verlag, 1. Auflage, 2004.

[32] GRUBER, THOMAS R.: *A translation approach to portable ontology specifications*. Knowledge Acquisition, 5(2):199–220, 1993.

[33] GRUNINGER, MICHAEL und JINTAE LEE: *Ontology applications and design*. Communications of the ACM, 45(2):39–41, 2002.

[34] GUARINO, NICOLA: *Formal Ontology and Information Systems*. In: GUARINO, NICOLA (Herausgeber): *Proceedings of the International Conference on Formal Ontology in Information Systems (FOIS '98), Trento, Italy, June 6-8, 1998*, Seiten 3–15, Amsterdam, 1998. IOS Press.

[35] HAYES, PATRICK: *RDF Semantics*. Technischer Bericht, World Wide Web Consortium (W3C), Februar 2004. W3C Recommendation. Verfügbar unter: http://www.w3.org/TR/2004/REC-rdf-mt-20040210/.

[36] HAYES, PATRICK: *Translating Semantic Web languages into Common Logic*. Unveröffentlichtes Dokument, verfügbar unter: http://www.ihmc.us/users/phayes/cl/sw2scl.html, Juli 2005.
Zuletzt besucht am: 26. April 2009.

[37] HAYES, PATRICK: *[CL] question about some statements in the Common Logic standard document*. Nachricht der Common Logic Mailingliste, verfügbar unter: http://philebus.tamu.edu/pipermail/cl/2008-September/001801.html, September 2008.
Zuletzt besucht am: 30. März 2009.

[38] HAYES, PATRICK und CHRISTOPHER MENZEL: *A Semantics for the Knowledge Interchange Format*. In: *Proceedings of the Workshop on the IEEE Standard Upper Ontology at IJCAI, Aug 6, 2001*, 2001.

[39] HAYES, PATRICK J.: *IKL Guide*. Unveröffentlichtes IKRIS Memorandum, verfügbar unter: http://www.ihmc.us/users/phayes/IKL/GUIDE/GUIDE.html, 2006.
Zuletzt besucht am: 03. Juni 2009.

[40] HAYES, PATRICK J. und CHRISTOPHER MENZEL: *IKL Specification Document*. Unveröffentlichtes IKRIS Memorandum, verfügbar unter: http://www.ihmc.us/users/phayes/IKL/SPEC/SPEC.html, 2006.
Zuletzt besucht am: 23. März 2009.

[41] HERRE, HEINRICH, BARBARA HELLER, PATRYK BUREK, ROBERT HOEHNDORF, FRANK LOEBE und HANNES MICHALEK: *General Formal Ontology (GFO): A Foundational Ontology Integrating Objects and Processes. Part I: Basic Principles (Version 1.0)*. Onto-Med Report 8, Research Group Ontologies in Medicine (Onto-Med), University of Leipzig, 2006.

[42] HERRE, HEINRICH, JAN JASPARS und GERD WAGNER: *Partial Logics with Two Kinds of Negation as a Foundation for Knowledge-Based Reasoning*. In: GABBAY, DOV M. und HEINRICH WANSING (Herausgeber): *What is Negation?*, Band 13 der Reihe *Applied Logic Series*, Seiten 121–159. Kluwer Academic Publishers, März 1999.

[43] HERRE, HEINRICH und GERD WAGNER: *Stable Models Are Generated by a Stable Chain*. Journal of Logic Programming, 30(2):165–177, 1997.

[44] HITZLER, PASCAL, MARKUS KRÖTZSCH, SEBASTIAN RUDOLPH und YORK SURE: *Semantic Web - Grundlagen*. eXamen.press. Springer-Verlag, Erste Auflage, 2008.

[45] HORROCKS, IAN und PETER F. PATEL-SCHNEIDER: *Three Theses of Representation in the Semantic Web*. In: *Proceedings of the 12th International World Wide Web Conference (WWW2003), Budapest, Hungary, May 20-24, 2003*, Seiten 39–47. ACM Press, 2003.

[46] HUANG, ZHISHENG, FRANK VAN HARMELEN und ANNETTE TEN TEIJE: *Reasoning With Inconsistent Ontologies: Framework, Prototype, and Experiment*. In: DAVIES, JOHN, RUDI STUDER und PAUL WARREN (Herausgeber): *Semantic Web Technologies: Trends and Research in Ontology-based Systems*, Kapitel 5, Seiten 71–93. John Wiley & Sons, Ltd, 2006.

[47] IRVINE, A. D.: *Russell's Paradox*. In: ZALTA, EDWARD N. (Herausgeber): *The Stanford Encyclopedia of Philosophy*. Stanford University, Fall 2008 Edition, September 2008. Verfügbar unter: http://plato.stanford.edu/archives/fall2008/entries/russell-paradox/.

[48] ISO/IEC: *Information technology - Common Logic (CL): a framework for a family of logic-based languages*. Technischer Bericht, International Organization for Standardization (ISO), Oktober 2007.

[49] KLYNE, GRAHAM und JEREMY J. CARROLL: *Resource Description Framework (RDF): Concepts and Abstract Syntax*. Technischer Bericht, World Wide Web Consortium (W3C), Februar 2004. W3C Recommendation. Verfügbar unter: http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/.

[50] LASSILA, ORA und DEBORAH MCGUINNESS: *The Role of Frame-Based Representation on the Semantic Web*. Linköping Electronic Articles in Computer and Information Science, 6(5), 2001.

[51] MANOLA, FRANK und ERIC MILLER: *RDF Primer*. Technischer Bericht, World Wide Web Consortium (W3C), Februar 2004. W3C Recommendation. Verfügbar unter: http://www.w3.org/TR/2004/REC-rdf-primer-20040210/.

[52] MENZEL, CHRISTOPHER: *The Common Logic Standard*. Verfügbar unter: http://metadata-standards.org/OpenForum2003/Presentations/Meeting%20Rooms%20Saved/Ballroom%20North/CL-ISO.ppt, 2003.
Zuletzt besucht am: 12. April 2009.

[53] MENZEL, CHRISTOPHER und PATRICK HAYES: *A New Axiomatic Semantics for Semantic Web Languages*. Unveröffentlichtes Dokument, verfügbar unter: http://cmenzel.org/Papers/AxiomaticSemantics.pdf, 2006.
Zuletzt besucht am: 28. Januar 2009.

[54] OMG: *OMG Unified Modeling Language (OMG UML), Infrastructure, V2.1.2*. Technischer Bericht, Object Management Group (OMG), November 2007.

[55] OMG: *OMG Unified Modeling Language (OMG UML), Superstructure, V2.1.2*. Technischer Bericht, Object Management Group (OMG), November 2007.

[56] OMG: *Semantics of Business Vocabulary and Business Rules (SBVR), Version 1.0*. Technischer Bericht, Object Management Group (OMG), Januar 2008.

[57] PORTO, FABIO: *Reasoning on Dynamically Built Reasoning Space with Ontology Modules*. In: MEERSMAN, ROBERT, ZAHIR TARI, MOHAND-SAID HACID, JOHN MYLOPOULOS, BARBARA PERNICI, ÖZALP BABAOGLU, HANS-ARNO JACOBSEN, JOSEPH P. LOYALL, MICHAEL KIFER und STEFANO SPACCAPIETRA (Herausgeber): *On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE, Proceedings of the OTM Confederated International Conferences, CoopIS, DOA, and ODBASE 2005, Agia Napa, Cyprus, October 31 - November 4, 2005,Part II*, Band 3761 der Reihe *Lecture Notes in Computer Science*, Seiten 1623–1628. Springer, 2005.

[58] PRIEST, GRAHAM: *Minimally inconsistent LP*. Studia Logica, 50(2):321–331, 1991.

[59] PRIEST, GRAHAM und KOJI TANAKA: *Paraconsistent Logic*. In: ZALTA, EDWARD N. (Herausgeber): *The Stanford Encyclopedia of Philosophy*. Stanford University, Summer 2009 Edition, Juni 2009. In Kürze verfügbar unter: http://plato.stanford.edu/archives/sum2009/entries/logic-paraconsistent/.

[60] PRUD'HOMMEAUX, ERIC und ANDY SEABORNE: *SPARQL Query Language for RDF*. Technischer Bericht, World Wide Web Consortium (W3C), Januar 2008. W3C Recommendation. Verfügbar unter: http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/.

[61] RIAZANOV, ALEXANDRE und ANDREI VORONKOV: *Vampire 1.1 (System Description)*. In: GORÉ, RAJEEV, ALEXANDER LEITSCH und TOBIAS NIPKOW (Herausgeber): *Automated Reasoning, Proceedings of the first International Joint Conference, IJCAR 2001, Siena, Italy, June 18-23, 2001*, Band 2083 der Reihe *Lecture Notes in Computer Science*, Seiten 376–380. Springer-Verlag, 2001.

[62] SCHAFFERT, SEBASTIAN, FRANÇOIS BRY, PHILIPPE BESNARD, HENDRIK DECKER, STEFAN DECKER, CARLOS F. ENGUIX und ANDREAS HERZIG: *Position Paper: Paraconsistent Reasoning for the Semantic Web*. In: COSTA, PAULO C. G. DA, KATHRYN B. LASKEY, KENNETH J. LASKEY und MICHAEL POOL (Herausgeber): *Proceedings of the ISWC Workshop on Uncertainty Reasoning for the Semantic Web (URSW), Galway, Ireland, November 7, 2005*, Band 173, 2005.

[63] SCHLOBACH, STEFAN, ZHISHENG HUANG, RONALD CORNET und FRANK VAN HARMELEN: *Debugging Incoherent Terminologies*. Journal of Automated Reasoning, 39(3):317–349, 2007.

[64] SHAPIRO, STEWART: *Foundations without Foundationalism - A Case for Second-Order Logic*. Oxford University Press, 2000. Verfügbar unter: http://www.oxfordscholarship.com/oso/public/content/philosophy/

9780198250296/toc.html, Oxford Scholarship Online.
Zuletzt besucht am: 11. April 2009.

[65] SHAPIRO, STEWART: *Classical Logic*. In: ZALTA, EDWARD N. (Herausgeber): *The Stanford Encyclopedia of Philosophy*. Stanford University, Fall 2008 Edition, September 2008. Verfügbar unter: http://plato.stanford.edu/archives/fall2008/entries/logic-classical/.

[66] SOWA, JOHN F.: *Common Logic Standard*, April 2002. Verfügbar unter: http://philebus.tamu.edu/pipermail/kif/2002-April/001069.html.
Zuletzt besucht am: 23. März 2009.

[67] SOWA, JOHN F.: *Conceptual Graphs*. In: HARMELEN, FRANK VAN, VLADIMIR LIFSCHITZ und BRUCE PORTER (Herausgeber): *Handbook of Knowledge Representation*, Band 3 der Reihe *Foundations of Artificial Intelligence*, Kapitel 5, Seiten 213–237. Elsevier, 2008.

[68] STAAB, STEFFEN: *Wissensmanagement mit Ontologien und Metadaten*. Informatik-Spektrum, 25(3):194–209, 2002.

[69] THE COMMON LOGIC WORKING GROUP: *Common Logic: Abstract Syntax and Semantics*. Ehemaliges Arbeitspapier, verfügbar unter: http://cl.tamu.edu/docs/cl/cl-latest.html, 2003.
Zuletzt besucht am: 10. April 2009.

[70] THE UNICODE CONSORTIUM: *The Unicode Standard, Version 5.0*. Addison-Wesley, 5. Auflage, November 2006.

[71] USCHOLD, MIKE und MICHAEL GRUNINGER: *Ontologies: Principles, Methods and Applications*. Knowledge Engineering Review, 11(2):93–155, 1996.

[72] W3C: *The Semantic Web Layer Cake*. World Wide Web Consortium (W3C). Verfügbar unter: http://www.w3.org/2007/03/layerCake.png.
Zuletzt besucht am: 11. März 2009.

[73] WAGNER, GERD, ADRIAN GIURCA, MIRCEA DIACONESCU, GRIGORIS ANTONIOU und CARLOS VIEGAS DAMÁSIO: *ERDF Implementation and Evaluation*. Technischer Bericht I1-D14, REWERSE, March 2008.

[74] WEIDENBACH, CHRISTOPH, RENATE A. SCHMIDT, THOMAS HILLENBRAND, ROSTISLAV RUSEV und DALIBOR TOPIC: *System Description: SPASS Version 3.0*. In: PFENNING, FRANK (Herausgeber): *Automated Deduction - CADE-21, Proceedings of the 21st International Conference on Automated Deduction, Bremen, Germany, July 17-20, 2007*, Band 4603 der Reihe *Lecture Notes in Computer Science*, Seiten 514–520. Springer-Verlag, 2007.

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe, insbesondere sind wörtliche oder sinngemäße Zitate als solche gekennzeichnet.

Mir ist bekannt, dass Zuwiderhandlung auch nachträglich zur Aberkennung des Abschlusses führen kann.

Ort	Datum	Unterschrift