Institut für Produktion und Industrielles Informationsmanagement

Universität Duisburg-Essen, Campus Essen
Fachbereich 5: Wirtschaftswissenschaften
Universitätsstraße 9, D - 45141 Essen
Tel.: ++49 (0) 201/ 183-4006, Fax: ++49 (0) 201/ 183-4017

KOWIEN-Projektbericht 4/2004

Erweiterung von Ontologien um dynamische Aspekte

Dipl.-Kfm. Yilmaz Alan

E-Mail: yilmaz.alan@pim.uni-essen.de

KOWIEN
KOOPERATIVES WISSENSMANAGEMENT IN ENGINEERING-NETZWERKEN

Das Drittmittelprojekt KOWIEN
("Kooperatives Wissensmanagement in Engineering-Netzwerken")
wird mit Mitteln des Bundesministeriums für Bildung und Forschung (BMBF)
(Förderkennzeichen Hauptband 02 PD 1060)
innerhalb des Rahmenkonzepts "Forschung für die Produktion von morgen"
gefördert und vom Projektträger Produktion und Fertigungstechnologien (PFT),
der Forschungszentrum Karlsruhe GmbH, betreut.
Die Mitglieder des Projektteams danken
für die großzügige Unterstützung ihrer Forschungs- und Transferarbeiten.

Juli 2004

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungs- und Akronymverzeichnis

Bd.	Band
DAML	Darpa Agent Markup Language
et al.	und andere
f.	folgende
ff.	fortfolgende
F-Logic	Frame-Logic – Programmiersprache
Fn.	Fußnote
Hrsg.	Herausgeber
Jg.	Jahrgang
KOWIEN	Kooperatives Wissensmanagement in Engineering Netzwerken (BMBF-Projekt)
Nr.	Nummer
Pr/T-Netz	Prädikat/Transition-Netz
PROLOG	Programming in Logic – Programmiersprache
RDF	Resource Description Framework
RDFS	Resorce Description Framework Schema
S.	Seite
ST/TR-Netz	Stelle/Transition-Netz
u.	und
XML	Extended Markup Language
z.B.	zum Beispiel

Symbolverzeichnis

$\notin$	nicht Element von
$\in$	Element von
$\wedge$	Konjunktor – logisches „Und“
$\veebar$	Disjunktor – logisches „Entweder...oder“
$\vee$	Adjunktor – logisches „Oder“
$\forall$	Allquantor („Für alle...“)
$\exists$	Existenzquantor („Es gibt ein...“)
$\rightarrow$	Subjugatspfeil („Wenn...dann...“)
$\leftrightarrow$	Bijugatspfeil („Genau dann...wenn...“)
λ	leeres Wort
Σ	Summensymbol
$\varnothing_M$	leere Multimenge
,	1.) Hilfszeichen 2.) Prolog – logische „Und“
:-	PROLOG - Subjugatspfeil
[...]	Liste
<-	F-Logic - Subjugatspfeil
=>	F-Logic – einwertiger Attributspfeil (Konzeptebene)
=>>	F-Logic – mengenwertiger Attributspfeil (Konzeptebene)
->	F-Logic – einwertiger Attributspfeil (Instanzenebene)
->>	F-Logic – mengenwertiger Attributspfeil (Instanzenebene)
$\mathcal{ALG}$(SIG)	Menge aller Algebren zu einer Signatur SIG
ARG_{OPS}	Argumentfunktion für Operationssymbole
ARG_{RS}	Argumentfunktion für Relationssymbole
EXISTS	F-Logic - Existenzquantor
F	Menge aller Flussrelationen im Petri-Netz
f	Wahrheitswert „falsch“
FORALL	F-Logic - Allquantor
$FORM_{RS}$	Menge aller Formeln über einer (relationalen) Signatur SIG_{RS}
$G,G_1,...,G_n$	Formeln
i	Index für Operationssymbole und Operationen
j	Index für Relationssymbole und Relationen

k	Index für Sorten
M(st)	Markierung der Stelle st
mult(a)	Multiplizität des Elementes a in der Multimenge mult
$\mathcal{MULT}$(A)	Menge aller Multimengen über der Trägermenge A
n	Index für formale Objekte
OBF	Familie aller Objektmengen
$ob_n,ob_1,...,ob_N$	formale Objekte
OB_s	Objektmenge zur Sorte s
$O_i,O_1,...,O_I$	Operationssymbole
$o_i,o_1,...,o_I$	Operation
OP_A	Menge aller Operationen
OPS	Menge der Operationssymbole
OPSF	Familie aller Operationssymbolmengen
pot(X)	Potenzmenge über der Menge X
$\mathcal{R}$	Menge der reellen Zahlen
$\mathcal{R}_-$	Menge der negativen reellen Zahlen
$R,R_1,...,R_J$	Relationssymbole
$r,r_1,...,r_J$	Relation
$\mathcal{R}_+$	Menge der positiven reelen Zahlen (inklusive 0)
RF	Familie aller Relationsmengen
RS	Menge der Relationssymbole
RSF	Familie aller Relationssymbolmengen
S	Menge der Sorten
SIG_{AS}	Algebraische Signatur
SIG_{OS}	Ontologische Signatur (Ontologie-Signatur)
SIG_{RS}	Relationale Signatur
$s_n,s_{n+1},s_k,s_1,...,s_K$	Sorten
ST	Menge aller Stellen im Petri-Netz
T	Maximale Sorte
$\mathcal{N}$	Menge der natürlichen Zahlen (inklusive 0)
$\mathcal{N}_+$	Menge der positiven natürlichen Zahlen
$t,t_1,...,t_n$	Terme
TERMF	Familie der sortenspezifischen Termmengen

TERM	Vereinigung aller sortenspezifischen Termmengen
$TERM_s$	Menge der Terme zur Sorte s
$TERM_{SIG}$	Menge aller Terme über der Signatur SIG
TR	Menge aller Transitionen im Petri-Netz
typ_{OPS}	Typisierungsfunktion für Operationssymbole
$typ_{RS\text{-}EOS}$	
typ_{RS}	Typisierungsfunktion für Relationssymbole
w	Wahrheitswert „wahr“
$x,y,x_1,...,x_n,y_1,...,y_n$	Variablen
$\mathbb{Z}$	Menge der ganzen Zahlen
$\mathbb{Z}_-$	Menge der negativen ganzen Zahlen
$\mathbb{Z}_+$	Menge der positiven natürlichen Zahlen (inklusive 0)
$ZIEL_{OPS}$	Zielfunktion für Operationssymbole

1 Exposition

1.1 Problemstellung

Traditionelle Methoden zur Unternehmensmodellierung weisen oft eine einseitige Ausrichtung auf. Entweder werden rein statische Aspekte des Unternehmens aufgezeigt oder es werden rein dynamische Aspekte hervorgehoben. In jüngster Zeit wird vermehrt der Versuch unternommen, beide Aspekte in einem ganzheitlichen Ansatz zu verfolgen. Dabei werden Methoden miteinander integriert, deren formale Basen miteinander „harmonieren".

Im Projekt KOWIEN werden zur Modellierung der statischen Aspekte *Ontologien* herangezogen. Ontologien wurden hierzu als eine „explizite und formalsprachliche Spezifikation der „sinnvollen" sprachlichen Ausdrucksmittel für eine von mehreren Akteuren gemeinsam verwendete Konzeptualisierung von realen Phänomenen, die in einem subjekt- und zweckabhängig einzugrenzenden Realitätsausschnitt als wahrnehmbar oder vorstellbar gelten und für die Kommunikation zwischen [...] Akteuren benutzt oder benötigt werden"[1)] eingeführt.

Ontologien beschränken sich auf die rein statischen Aspekte, da sie nur für die Repräsentation *deklarativen* Wissens verwendet werden können. Daher haben Ontologien auch nur eine deklarative Semantik[2)]. Sie äußert sich z.B. darin, dass die Reihenfolge der (Teil-)Spezifikationen für ihre Bedeutung irrelevant ist. Dadurch kann immer nur *ein* bestimmter *Zustand* der Realität modelliert werden. Eine Ontologie-gestützte Wissensbasis enthält nämlich nur *Fakten* über die Welt. Die abgelegten Fakten können *explizit* vorgegeben oder *implizit* in der Wissensbasis enthalten sein. Letztgenannte können mit Hilfe der Regelkomponente einer Ontologie expliziert werden.

---

1) ALAN (2003) S. 6. Die dortige Definition weist eine Anlehnung an GRUBER (1993) S. 199 auf. Die Basis-Definition von GRUBER – die sich in der Literatur durchgesetzt hat – wurde im Projekt KOWIEN um Aspekte erweitert, die üblicherweise implizit vorausgesetzt werden.

2) Die deklarative Semantik von Formelmengen entspricht der Menge an Formeln, die sich als logische Konsequenz aus der ursprünglichen Formelmenge ergibt (vgl. ZELEWSKI (1995) Bd. 4 S. 150). Eine Formel ist wiederum dann logische Konsequenz einer Formelmenge, wenn sie augrund einer *modelltheoretischen* oder *semantischen* Schlussfolgerungsrelation mit dieser verbunden ist. Die modelltheoretische Schlussfolgerungsrelation wird später noch vertieft werden.

Dynamische Aspekte lassen sich mit Ontologien allerdings nur bedingt[1)] umsetzen. Dynamische Aspekte sind dadurch gekennzeichnet, dass auch prozedurales Wissen in einem Modell repräsentiert wird. Bei einer prozeduralen Wissensrepräsentation würden Aktivitäten im Vordergrund stehen, die möglicherweise eine Modifikation der Wissensbasis zu Folge haben. Ontologie-gestützte Systeme können auch als *passive* Systeme bezeichnet werden, da zu ihrer Modifikation ein externer Eingriff notwendig ist. Um Ontologien in einen ganzheitlichen Ansatz einbinden zu können, bedarf es aufgrund daher ihrer Erweiterung um eine *prozedurale* Semantik. Dadurch würden Ontologie-gestützte Systeme um eine *reaktive* Komponente erweitert werden.

Eine solche Erweiterung könnte das Themenfeld Ontologien für die Wirtschaftsinformatik interessanter machen als es bisweilen ist. In der Wirtschaftsinformatik ist in den letzten Jahren vorwiegend das prozessorientierte Paradigma in den Vordergrund gerückt. Ein Anzeichen hierfür ist die Vielzahl an Methoden, die dem prozessorientierten Paradigma zuzurechnen sind. Hierzu zählen z.B. *Ereignisgesteuerte Prozessketten*, Verhaltens- und Aktivitätsdiagramme der *Unified Modeling Language* und *Petri-Netze* in unterschiedlichen Varianten. Würden nun Ontologien mit einer bestehenden Methode in Einklang gebracht werden, könnte dies zur gegenseitigen Befruchtung beitragen.

Mit dem vorliegenden Projektbericht wird ein erster Versuch unternommen, Ontologien in einen ganzheitlichen Ansatz einzubinden. Die Einbindung erfolgt durch die Integration von Ontologien in *Höheren Petri-Netzen*. Hinsichtlich der obigen Unterscheidung zwischen deklarativer und prozeduraler Wissensdarstellung lässt sich der hier vorgestellte Ansatz als *hybrid* klassifizieren. Dabei wird eine Klasse von Höheren Petri-Netze zu Grunde gelegt, die aufgrund ihrer strengen prädikatenlogischen Orientierung als besonders fruchtbar für das Vorhaben erscheint. Das prädikatenlogische Kalkül verträgt sich nämlich reibungslos mit dem algebraischen Kalkül, das hier für Ontologien vorausgesetzt wird.

---

1) Es ist nicht grundsätzlich ausgeschlossen, dass prozedurale Aspekte auch mittels Ontologien ausgedrückt werden können. Es wurden sogar bisweilen mehrere Ontologien vorgestellt, die zu Zwecken der Repräsentation von Wissen über Prozesse verwendet werden können (vgl. AITKEN/CURTIS (2002) S. 110 f.; SEIBT (2001) S. 334 ff.; USCHOLD ET AL. (1998) S. 46 ff.). In den genannten Arbeiten wird vorwiegend aufgezeigt, wie prozessspezifische Begriffe mittels einer Ontologie formal präzisiert werden können. Dieser Ansatz wird grundsätzlich in der vorliegenden Arbeit nicht ausgeschlossen. Darüber hinaus wird in den Arbeiten allerdings nicht aufgezeigt, wie *Übergänge* zwischen den Zuständen einer Ontologie-gestützten Wissensbasis durch die Erweiterung der prädikatenlogischen Wissensrepräsentation um eine prozedurale Semantik formalisiert werden können.

1.2 Gang der Untersuchung

Den Hauptteil der Arbeit bildet der folgende Abschnitt 2. In Abschnitt 2.1 wird zunächst der deklarative Hintergrund des Integrationskonzepts vorgestellt. Es handelt sich dabei um Signaturen, Algebren, Ausdrücke über Signaturen und Spezifikationen. Darauf folgend werden in Abschnitt 2.2 Ontologien als Sonderform algebraischer Spezifikationen charakterisiert. In Abschnitt 2.3 wird der prozedurale Hintergrund des Integrationskonzepts vorgestellt. Es handelt sich dabei um Petri-Netze, die in Form von Allgemeinen- und Stelle/Transition-Netzen vorgestellt werden. In Abschnitt 3 werden die deklarativen und prozeduralen Bausteine mit Hilfe des Integrationskonzeptes zusammengeführt. Die Arbeit wird mit einer kritischen Würdigung in Abschnitt 4 abgeschlossen.

2 Bausteine des integrativen Modellierungskonzeptes

2.1 Algebraisch-prädikatenlogischer Rahmen für das Integrationskonzept

2.1.1 Überblick über die mehrsortige Prädikatenlogik

Bei der Repräsentation von Wissen mittels der „konventionellen“ einsortigen Prädikatenlogik erster Ordnung[1)] wird ein *strukturloser* Individuenbereich vorausgesetzt, dessen Objekte lediglich durch Relationen und Funktionen beschrieben werden. Durch den Übergang zur *mehrsortigen Logik*[2)] wird der unstrukturierte Individuenbereich vor der relationalen und funktio-

1) Für den vorliegenden Projektbericht wird lediglich das Kalkül der Prädikatenlogik *erster* Ordnung benötigt. Daher wird im Folgenden der Zusatz „erster Ordnung“ weggelassen werden. Die Beschränkung auf die Prädikatenlogik erster Ordnung äußert sich darin, dass die – noch vorzustellenden – Quantoren und Relationssymbole sich lediglich auf – ebenso noch vorzustellende – Terme beziehen dürfen. Dieser Einschränkung des Ausdrucksvermögens liegt eine stufenweise Unterscheidung von formalen Objekten zu Grunde. Während die Elemente (Terme) des Grundbereichs als formale Objekte *erster* Ordnung behandelt werden, gelten Relations- und Operationssymbole als Objekte *zweiter* Stufe. Durch den Übergang auf die Prädikatenlogik *zweiter* Ordnung wird der Bezug von Quantoren und Relationssymbolen zu Objekten zweiter Stufe erlaubt (vgl. MANZANO (1993) S. 46 ff.). Dadurch lassen sich beispielsweise Eigenschaften von Relationssymbolen explizit ausdrücken, die in einer Prädikatenlogik erster Ordnung lediglich implizit enthalten sein können. Dieser Aspekt der Prädikatenlogik zweiter Ordnung erweist sich im Kontext von Ontologien als viel versprechend (vgl. GUARINO/WELTY (2000) S. 101 ff. für den Hinweis auf „Meta-Prädikate“, die zur Charakterisierung von Konzepten in einer Ontologie verwendet werden können). Allerdings wird der mit einem Übergang zur Prädikatenlogik zweiter Ordnung verbundene Komplexionssprung vom Verfasser als zu hoch eingeschätzt, als dass die Vorteile des Zugewinns an Ausdrucksvermögen gerechtfertigt wären. Darüber hinaus ist mit dem Übergang zur Prädikatenlogik zweiter Ordnung der Verlust beweistheoretischer Eigenschaften der Prädikatenlogik verbunden (vgl. KOHLHASE (1992) S. 226 ff.).

2) Für einen Überblick zur mehrsortigen Prädikatenlogik vgl. EHRIG ET AL. (1999) S. 307 ff.; GUESSERIAN (1993) S. 124 ff.; KREOWSKI (1991) S. 33 ff.; LOECKX ET AL. (1996) S. 78 ff.; MANZANO (1993) S. 3 ff.; OBERSCHELP (1962) S. 297 ff. Für eine Umsetzung eines konkreten Spezifikationsansatzes für das Produktionsmanagement mittels sortierter Prädikatenlogik vgl. PATIG (2001) S. 53 ff.

nalen Beschreibung der darin enthaltenen Objekte in *Sorten* aufgeteilt[1]. Durch die sortenspezifische Zuordnung von Objektmengen können Aussagen aufgrund der Unmöglichkeit ihnen einen *Sinn*[2] zuzuweisen ausgeschlossen werden.

Beispielsweise würde in einer einsortigen Logik die Aussage, dass eine Person im Arbeitsverhältnis mit einer natürlichen Zahl steht, als falsch ausgezeichnet werden. Es dürfte allerdings einsichtig sein, dass auch die Falschheit einer solchen Aussage einen qualitativen Unterschied zu Falschheiten beinhaltet, die zwar in der betrachteten Domäne falsch sind aber – rein begrifflich vorstellbar – in einer möglichen Welt auch wahr sein könnten. Eine solche Aussage wäre die Formel

arbeitet_fuer(mueller,DMT),

die in der betrachteten Domäne *falsch* sein kann, wenn die Person „mueller“ nicht für das Unternehmen „DMT“ arbeitet, aber grundsätzlich in einer anderen Welt zu *wahr* ausgewertet werden könnte.

Das Problem tritt deutlicher zu Tage, wenn *regelhafte* Aussagen formuliert werden. Im zweiten Beispiel soll ausgesagt werden, dass alle geraden Zahlen durch 2 teilbar sind:

$\forall x$: gerade_Zahl(x) $\rightarrow$ teilbar_durch_zwei(x)

Problematisch wird die Auswertung der Regel, wenn die Variable x aus der Formel durch eine Person („mueller“) interpretiert werden soll. Da in diesem Fall die Teilformel

gerade_zahl(mueller)

nicht als *wahr* ausgewertet werden kann, bleibt lediglich ihre Auswertung als *unwahr*. In diesem Fall könnte auch die Unwahrhaftigkeit der Teilformel

---

1) Die konventionelle Prädikatenlogik stellt einen Sonderfall der sortierten Prädikatenlogik dar (vgl. MANZANO (1993) S. 4; EHRIG ET AL. (1999) S. 311). Der Erweiterung durch die sortierte Prädikatenlogik liegt darin, dass die konventionelle Prädikatenlogik nur *eine* Sorte betrachtet. Die Sortenmenge der konventionellen Prädikatenlogik hat daher stets die Mächtigkeit $|S|=1$. In der sortierten Prädikatenlogik ist hingegen grundsätzlich eine Mächtigkeit der Sortenmenge $|S|\geq 1$ zugelassen. Darüber hinaus kann jede Sorte aus einer auch als einstelliges Prädikatsymbol mit entsprechender Interpretation durch ein Prädikat betrachtet werden (vgl. BEIERLE ET AL. (1993) S. 192).

2) Die Unterscheidung zwischen *Sinn* und *Bedeutung* einer Aussage geht zurück auf FREGE (vgl. z.B. FREGE (1966) S. 33). FREGE schlägt zur Kennzeichnung von Aussagen eine dreistellige Relation mit den Komponenten *Name*, *Sinn* und *Bedeutung* vor. Während der Name einer Aussage zu ihrer sprachlichen Kennzeichnung verwendet werden kann, beziehen sich Sinn und Bedeutung auf die inhaltliche Auswertung. Die Bedeutung einer Aussage entspricht im traditionellen Verständnis der Relation zwischen dem Namen einer Aussage und ihrer Extension. Durch die FREGEsche Erweiterung um den Sinn einer Aussage wird hingegen eine Relation zwischen der Aussage und ihrer Intention aufgebaut.

teilbar_durch_zwei(mueller)

zugelassen werden, wodurch die Gesamtformel bei sinnloser Belegung einer ihrer Variablen aufgrund der Unwahrhaftigkeit von sowohl ihrer Antezedenzformel als auch ihrer Konklusionsformel als *wahr* ausgewertet würde. Bei einer Beschränkung auf zweiwertige Aussagenformen, können also Formeln nicht sinnvoll ausgewertet werden, die *inhaltliche* Inkonsistenzen aufweisen. Eine Lösungsmöglichkeit würde im Auslassen der Auswertung solcher Aussagen liegen. Die Auswertung eines Formelsystems entspräche dann einer *partiellen Interpretation*. Solche Verfahren weisen allerdings anderweitige Inkonsistenzen auf, die die eindeutige Definition der Semantik erschweren.[1)]

Die Kollision von *Bedeutung*[2)] und *Sinn* einer prädikatenlogischen Aussage kann durch die sortenspezifische Zuordnung von Objekten überbrückt werden. Bei allen prädikatenlogischen Formeln stellt sich nämlich vor der Frage ihrer Bedeutung stets die Frage nach ihrer Sinnhaftigkeit. Bei einer mehrsortigen Logik kann beispielsweise im Vorfeld bestimmt werden, dass Arbeitsverhältnisse lediglich zwischen Personen definiert sind. Dadurch wird eine Aussage der obigen Form im Vorfeld wegen ihrer Sinnlosigkeit ausgeschlossen. Die Frage nach der Bedeutung (Wahrheitsgehalt) der Formel wird dann zweitrangig, weil die Formel aufgrund ihres fehlenden Sinns (Sortentreue) gar nicht konstruiert werden kann, da sie syntaktisch unzulässig ist. Der syntaktische Ausschluss sinnloser Formeln ist somit semantisch motiviert. Bereits bei der Konstruktion von Formeln werden solche Formeln ausgeschlossen, deren Bedeutung gar nicht *wahr* sein kann. Schließlich ist jede sinnlose Formel auch stets in ihrer Bedeutung falsch.[3)]

Die Umsetzung erfolgt durch eine Typisierung der Prädikate und Operationen, die verwendet werden.

---

1) Vgl. EHRIG ET AL. (1999) S. 310.

2) Der Begriff *Bedeutung* wird im Folgenden als Unterbegriff zum Begriff *Semantik* verwendet. Mit „Bedeutung“ wird der extensionale Aspekt von Semantik angesprochen. Der intensionale Aspekt einer Semantik wird durch den zweiten nUnterbegriff *Sinn* erfasst.
Bei der Semantik prädikatenlogischer Formeln handelt es sich um eine *formale* Semantik. Formale Semantiken weisen einem Formelsystem eine Interpretation zu, indem formalen Objekten wiederum formale Objekte zugewiesen werden. das zugewiesene Objektsystem ist dann die formale Semantik des interpretierten Formelsystems.

3) Ausnahmen hiervon sind zusammengesetzte Formeln, die zu „wahr“ ausgewertet werden müssen, weil ihre sinnlosen Teilformeln „falsch“ sind. Beispielsweise ist die Formel

$$\forall x: \text{ganze\_Zahl}(\text{mueller}) \rightarrow \text{arbeitet\_fuer}(3,\text{DMT})$$

offensichtlich sinnlos, da sowohl die Antezedenz- als auch die Konklusionsformel „sinnlos“ und somit „falsch“ sind. Die gesamte Subjugatsformel ist allerdings aufgrund der Falschheit der Antezedenzformel „wahr“. Es wird jedoch nochmals darauf hingewiesen, dass in einer mehrsortigen Prädikatenlogik eine solche Formel gar nicht zulässig wäre, da in den Argumenten der Relationssymbole Terme verwendet werden, die unzulässig sind.

2.1.2 Signaturen

2.1.2.1 Algebraische Signaturen

Eine algebraische Signatur SIG_{AS} ist definiert als ein Drei-Tupel:

$$SIG_{AS}=(S,OPS,typ_{OPS}).$$

Die Komponenten einer algebraischen Signatur sind:

(1.) eine Menge von *Sorten* S,

(2.) eine Menge von *Operationssymbolen* OPS und

(3.) eine Funktion typ_{OPS} zur *Operationssymboltypisierung*.

Mit $S=\{s_1,...,s_K\}$ mit $K\in\mathcal{N}_+$ wird die Menge der *Sorten* bezeichnet, die später durch sortenspezifische Objektmengen aus einer Algebra interpretiert werden. S entspricht einem Alphabet, dessen Elemente $s_1,...,s_K$ einstellige Wörter über S sind. Mit S* wird die Menge aller Wörter bezeichnet, die über der Sortenmenge S definiert ist. Die Elemente von S* sind sowohl einstellige Wörter aus der Menge S als auch *Wortketten*, die aus der Aneinanderreihung von mindestens zwei Wörtern oder Wortketten hervorgehen. Jedes $w\in S^*$ ist somit ein(e) Wort(-kette) über S. S* hat auch das leere Wort λ als Element.

Mit $OPS=\{O_1,...,O_I\}$ mit $I\in\mathcal{N}$ wird die Menge aller Operationssymbole angesprochen. Sie hat Operationssymbole als Elemente. Jedem Operationssymbol O_i wird durch die Typisierungsfunktion[1)]

$$typ_{OPS}: OPS \rightarrow S^*\times S$$

eine Stelligkeit zugewiesen. Für eine Operationssymbol $O_i\in OPS$ mit $typ_{OPS}(O_i)=s_1...s_n,s_{n+1}$ geben $s_1...s_n$ die *Argumentsorten* und s_{n+1} die *Zielsorte* an. Die Sortenkette $s_1...s_n$ kann auch als das Wort $w=s_1...s_n$ angesprochen werden. Für den formalen Zugriff auf Argument- und Zielsorten von Operationssymbolen, können die *Argumentfunktion*

$$ARG_{OPS} : OPS \rightarrow S^*$$

bzw. die *Zielfunktion*

$$ZIEL_{OPS} : OPS \rightarrow S$$

---

1) Bei der Typisierungsfunktion handelt es sich um einen *metasprachlichen* Ausdruck, der verwendet wird, um auf der Signatur-Basis eine formale Sprache zu konstruieren. Es wird ausdrücklich darauf hingewiesen, dass es sich nicht um eine Funktion handelt, die mittels eines Operationssymbols aus der Signatur beschrieben wird.
Die Typisierung von Operations- und späteren Relationssymbolen wird auch als *Deklaration* angesprochen. Der Begriff der Typisierung legt es nahe, den Begriff der *Sorte* synonym mit dem Begriff *Typ* zu verwenden.

verwendet werden. Sie wurden nicht explizit in der Signaturdefinition aufgeführt, da beide Funktionen sich aus der Typisierungsfunktion typ_{OPS} ergeben. Für ein Operationssymbol $O_i \in OPS$ mit $typ_{OPS}(O_i)=s_1...s_n,s_{n+1}$ sind $ARG_{OPS}(O_i)=s_1...s_n$ und $ZIEL_{OPS}(O_i)=s_{n+1}$. Wenn ein Operationssymbol $O_i \in OPS$ mit $typ(O_i)=w,s$ mit $w=\lambda$ und $s \in S$ gegeben ist, wird das Operationssymbol auch als *Konstantensymbol* bezeichnet.

Die Operationssymbole $O_1,...O_I$ werden später durch Operationen aus einer signaturspezifischen Algebra interpretiert. Jedes Operationssymbol $O_i \in OPS$ mit $typ_{OPS}(O_i)=s_1...s_n,s_{n+1}$ kann daher auch durch eine funktionsnahe Notation der Form

$$O_i: s_1...s_n \rightarrow s_{n+1}$$

mit $n \in \mathcal{N}$ und $s_1,...,s_n,s_{n+1} \in S$ angegeben werden.

2.1.2.2 Relationale Signaturen

Eine relationale Signatur ist definiert als ein Fünf-Tupel

$$SIG_{RS}=(S,OPS,typ_{OPS},RS,typ_{RS})^{1)}.$$

Die Komponenten einer relationalen Signatur sind:

(1.) eine *algebraische Signatur* $SIG_{ALG}=(S,OPS,typ_{OPS})$,[2)]

(2.) eine Menge von *Relationssymbolen*[3)] RS und

(3.) eine Funktion typ_{RS} zur *Relationssymboltypisierung*.

Mit $RS=\{R_1,...,R_J\}$ mit $j=1...J$ und $J \in \mathcal{N}_+$ wird die Menge aller Relationssymbole angesprochen. Eine relationale Signatur setzt eine nicht-leere Menge RS von Relationssymbolen R_j voraus. Ähnlich der Typisierung von Operationssymbolen können auch Relationssymbole mittels einer Typisierungfunktion typ_{RS} deklariert werden. Ansonsten handelt es ich um eine algebraische Signatur. Jedem Relationssymbol R_j wird durch die *Relationssymbol-Typisierungsfunktion*

$$typ_{RS}: RS \rightarrow S^*$$

eine Stelligkeit zugewiesen. Für ein Relationssymbol $R_j \in RS$ mit $typ_{RS}(R_j)=s_1...s_n$ gibt die Relationssymbol-Typisierungsfunktion typ_{RS} ihre Argumentsorten an. Analog zu der Vorge-

---

1) Der Index AS bzw. RS wird im Folgenden nur dann verwendet werden, wenn zwischen einer algebraischen und relationalen Signatur explizit unterschieden werden soll. Die sonstige Verwendung ergibt sich aus dem Kontext.

2) Wenn im Folgenden der Unterschied zwischen SIG_{RS} und SIG_{AS} von Bedeutung ist, wird der Index beibehalten. Sollte die Unterscheidung nicht von Relevanz sein, wird $SIG \in \{SIG_{RS},SIG_{AS}\}$ verwendet.

3) Die Begriffe *Relationssymbol* und *Prädikatssymbol* werden synonym verwendet.

hensweise bei Operationssymbolen kann auf das Argument eines Relationssymbols durch die Argumentfunktion

$$ARG_{RS}: RS \rightarrow S^*$$

zugegriffen werden. Da das Argument eines Relationssymbols $R_j \in RS$ stets mit seiner Typisierung $typ_{RS}(R_j)$ übereinstimmt, gilt für jedes Relationssymbol $ARG_{RS}(R_j)=typ_{RS}(R_j)$. Für ein Relationssymbol $R_j \in RS$ mit $typ_{RS}(R_j)=s_1...s_n,s_{n+1}$ gilt somit $ARG_{RS}(R_j)=s_1...s_n$.

Für n>1 wird durch eine Relationssymbol $R_j \in RS$ mit $typ_{RS}(R_j)=s_1...s_n$ eine *Beziehungen* zwischen n Objekten aus sortenspezifischen Objektmengen bezeichnet. Wenn n=1 und somit w=s ist, wird durch das Relationssymbol R_j keine Beziehung zwischen Objekten angedeutet, sondern eine *Eigenschaft*, die einem sortenspezifischen Objekt zukommen kann. Ist n=0 und somit $w=\lambda$, dann handelt es sich um ein Relationssymbol, das die Qualität einer (argumentlosen) aussagenlogischen Formel hat.

2.1.3 Algebren

2.1.3.1 Algebren zu algebraischen Signaturen

Eine Algebra zu einer algebraischen Signatur ist ein Zwei-Tupel:

$$A_{AS}=(OBF, OPF).$$

Die Komponenten einer Algebra sind:

(1.) eine Familie $OBF=(OB_s)_{s \in S}$ von *Objektmengen* und

(2.) eine Familie $OPF=(o_i)_{i \in I}$ von *Operationen*.

Der Index AS deutet darauf hin, dass es sich dabei um eine Algebra zu einer algebraischen Signatur handelt. Im nächsten Abschnitt werden Algebren zu relationalen Signaturen vorgestellt. Sie werden dann zur Unterscheidung mit einem anderen Index (RS) versehen.

Signaturen entsprechen dem rein *syntaktischen* Aspekt des vorgestellten Ansatzes. Mit Signaturen wird lediglich die *Grammatik* zur Konstruktion einer formalen Sprache angegeben. Um Aussagen, die mit Hilfe der Signatur konstruiert werden, interpretieren zu können, bedarf es eines Objektbereiches durch den die Komponenten einer Signatur interpretiert werden.

Das *semantische*[1] Pendant zu algebraischen Signaturen wird durch *Algebren* gegeben, die den Sorten einer Signatur Mengen formaler Objekte und den Operationssymbolen Operationen[2] auf den Mengen formaler Objekte zuordnen. Somit werden durch Sorten auf einer abstrakten Ebene Mengen beschrieben.

$OBF=(OB_s)_{s\in S}$ ist eine Mengenfamilie[3], deren Mitglieder Mengen formaler Objekte entsprechen. Sie werden als *Trägermengen* bezeichnet. Jeder Sorte $s_k\in S$ mit k=1,...,K und $K\in\mathcal{N}_+$ einer Signatur ist eine Menge $OB_{s.k}\in OBF$[4] zugeordnet. Die Mitglieder der Mengenfamilie OBF können grundsätzlich gleich sein. Das heißt, das zwei unterschiedlichen Sorten s_1,s_2 die gleichen Objektmenge $OB_{s.1}=OB_{s.2}$ zugeordnet werden können.

Die Mengenfamilie $OPF=(o_i)_{i=1,...,I}$ mit $I\in\mathcal{N}$ enthält Mengen von Operationen auf den Trägermengen. Jedes Element $o_i\in OPF$ aus einer Operationsmengenfamilie OPF ist die Interpretationen eines Operationssymbols $O_i\in OPS$ aus der zu Grunde gelegten Signatur SIG_{AS}.[5] Jedem Operationssymbol $O_i\in OPS$ mit $typ_{OPS}(O_i)=s_1...s_n,s_{n+1}$ wird eine Operation

$$o_i:OB_{s.1}\times...\times OB_{s.n}\rightarrow OB_s$$

zugeordnet. Dabei wird das kartesische Produkt $OB_{s.1}\times...\times OB_{s.n}$ als *Argumentbereich* der Operation o_i bezeichnet. $OB_{s.n+1}$ ist der *Zielbereich* der Operation o_i. Als Spezialfall für ein Operationssymbol $O_i\in OPS$ mit der Typisierung $typ_{OPS}(O_i)=\lambda,s$, das bereits als *Konstantensymbol* eingeführt wurde, werden auch entsprechende Operationen mit leerem Vorbereich zugelassen. Sie werden dann als *Konstanten* aus der Menge OB_s bezeichnet, die der Zielsorte s des nullstelligen Operationssymbols $O_i\in OPS$ zugeordnet ist.

Grundsätzlich können für eine Signatur *mehrere* Algebren als Interpretationen in Frage kommen. Mit $\mathcal{ALG}(SIG_{AS})$ wird die Menge aller Algebren bezeichnet, die für eine Signatur gegeben ist. Es wird später aufgezeigt, wie die Menge der in Frage kommenden Algebren durch Anforderungen an die Algebren eingeschränkt werden kann.

---

1) Wie bereits zuvor erwähnt, handelt es sich hierbei um eine *formale* Semantik. Die formalen Konstrukte einer Signatur werden nämlich durch wiederum formale Konstrukte aus einer Algebra interpretiert. Bei einer materialen Semantik würden die Konstrukte einer Signatur durch reale Objekte interpretiert werden.

2) Die Begriffe *Operation* und *Funktion* werden synonym verwendet.

3) Eine Mengenfamilie ist eine Menge, deren Elemente selbst Mengen sind. Die Elemente einer Mengenfamilie werden als „Mitglieder der Mengenfamilie“ bezeichnet.

4) Aufgrund „layout-technischer“ Erfordernisse wird auf alle mehrfachen Indexierungen verzichtet. Der Index „s.k“ einer Objektmenge $OB_{s.k}$ ist daher als s_k zu lesen. Es gilt daher $s.k=s_k$ für alle $k\in 1...K$ und $K\in\mathcal{N}$.

5) Die Unterscheidung zwischen einem Operationssymbol $O_i\in OPS$ und der Operation $o_i\in OPF$ erfolgt durch Groß- bzw. Kleinschreibung von Operationssymbolen bzw. Operationen.

2.1.3.2 Algebren zu relationalen Signaturen

Eine Algebra zu einer relationalen Signatur hat die Form:

$$A_{RS}=(OBF,OPF,RF).$$

Die Komponenten einer relationalen Signatur sind:

(1.) eine Familie $OBF=(OB_s)_{s\in S}$ von Operationsmengen,

(2.) eine Familie $OPF=(o_i)_{i=1...I}$ mit $I\in\mathcal{N}_+$ von Operationsmengen und

(3.) eine Familie $RF=(r_j)_{j=1...J}$ mit $J\in\mathcal{N}_+$ von Relationsmengen.

Um eine stärkere begriffliche Trennung zwischen den semantischen Pendants zu algebraischen und relationalen Signaturen vorzunehmen, werden die Algebren zu relationalen Signaturen auch als *Strukturen* angesprochen.[1)] Da es sich um eine Ausweitung von Algebren zu algebraischen Signaturen handelt, werden sie weiterhin mit A – nun aber mit dem Index RS – bezeichnet.

Strukturen zu relationalen Signaturen erweitern Algebren zu algebraischen Signaturen um eine Familie RF von Relationsmengen r_j mit j=1,...,J und $J\in\mathcal{N}_+$, deren Elemente Relationen[2)] auf den Trägermengen definieren. Für jedes Relationssymbol $R_j\in RS$ mit der $typ_{RS}(R_j)=s_1...s_n$ enthält die Mengenfamilie RF eine Relation r_j[3)] der Form

$$r_j\subseteq OB_{s.1}\times...\times OB_{s.n}.$$

Mit Relationen werden Beziehungen zwischen formalen Objekten auf wiederum formale Weise ausgedrückt. Die Elemente einer Relation sind n-Tupel von Objekten, zwischen denen die Relation existiert. Diese Elemente müssen in ihren Komponenten jeweils Elemente sortenspezifischer Objektmengen aufweisen. Für die oben erwähnte Relation $r_j\subseteq OB_{s.1}\times...\times OB_{s.n}$ bedeutet dies z.B., dass das n-Tupel $(ob_1,...,ob_n)$ nur dann Element der Relation r_j sein kann, wenn für alle ob_i mit i=1,...,n gilt $ob_i\in OB_{s.i}$.

Die Relation r_j wird auch als *Extension* des Relationssymbols $R_i\in RS$ bezeichnet. Die Extension eines Relationssymbols R_i ist die Menge aller n-Tupel $(ob_1,...,ob_n)$ aus formalen Objekten ob_i mit i=1,...,n und $ob_i\in OB_{s.i}$, die als Argumente für das Relationssymbol R_i mit der Typisie-

---

1) Vgl. EHRIG ET AL. (1998) S. 312; KREOWSKI (1991) S. 39.

2) Die Begriffe *Relation* und *Prädikat* werden synonym verwendet

3) Analog zu der Vorgehensweise bei Operationssymbolen und Operationen wird zur Unterscheidung zwischen Relationssymbolen bzw. Relationen zwischen Groß- bzw. Kleinschreibung unterschieden.

rung $typ_{RS}(R_i)=s_1...s_n$ aus der zu Grunde liegenden Signatur SIG_{RS} in Betracht kommen und die die Relation $r_j(ob_1,...,ob_n)$ in der relationalen Algebra A_{RS} erfüllen. Durch diesen Ansatz wird eine *extensionale* Semantik aller prädikatenlogischen Formeln G aus der Menge $FORM_{SIG}$ fundiert.[1)] Diese Semantik weist jedem Relationssymbol R_i eine Menge extensional definierter Konstrukte aus Signatur-bezogenen relationalen Algebra als Interpretation zu. *Intensionale* Semantiken werden dadurch für Relationssymbole ausgeschlossen.

2.1.4 Ausdrücke

2.1.4.1 Terme als Ausdrücke über algebraischen Signaturen

2.1.4.1.1 Definition von Termen

Mit *Termen* kann auf die Objekte einer Algebra zugegriffen werden. Dadurch können die Ergebnisse der Anwendung von Funktionen auf ihre Argumente bereits in der Ebene der Signatur „errechnet" werden. Terme sind zudem die erste Form von *Ausdrücken*, die über einer Signatur formulierbar sind. Als zweite Form von Ausdrücken werden später *Formeln* vorgestellt werden. Sie werden die Ausdrucksmächtigkeit des vorgestellten Konzeptes um relationale Aspekte erweitern. Als drittes Ausdrucksmittel werden *Multimengen* vorgestellt werden. Alle drei Ausdrucksarten zusammen konstituieren schließlich eine prädikatenlogische Sprache. Dabei ist zu beachten, dass Formeln und formale Summen zu ihrer Konstruktion Terme voraussetzen. Insofern ist jede Unterteilung der konstruierten formalen Sprache in drei Teilsprachen – zumindest formal – nicht vertretbar.[2)]

$VARF=(VAR_s)_{s\in S}$ ist eine Mengenfamilie, deren Mitglieder sortenspezifische *Variablenmengen* sind.[3)]Die Elemente einer sortenspezifischen Variablenmenge VAR_s werden als Variablen

---

1) Vgl. ZELEWSKI (1995) Bd. 4 S. 115.

2) Eine solche Unterscheidung findet sich z.B. in TACKEN (1997) S. 9. Dort findet eine Unterteilung in die Sprache der Terme, die Sprache der Formeln und die Sprache der Multimengen statt. Die prädikatenlogische Sprache wird dann als Vereinigungsmenge dieser drei Sprachen aufgeführt. Formal mag eine solche Unterteilung korrekt sein. Allerdings wird durch eine solche Unterteilung die Vermutung nahe gelegt, es handele sich um drei paarweise disjunkte Sprachen. Die Disjunktheit trifft allerdings nur zu, wenn die Sprache der Formeln und die Sprache der Multimengen betrachtet werden. Beide Sprachen greifen allerdings bei ihren Konstruktionsregeln auf die Sprache der Terme zurück. Daher trifft die vorgenannte Disjunktheitsvermutung nicht zu, so dass die Unterteilung in drei Teilsprachen nicht im strengen Sinne falsch, aber zumindest kontraintuitiv ist. Daher wird auf eine „artifizielle" Dreiteilung hier verzichtet, zumal ihre Verwendung zu keinem nennenswerten Erkenntnisgewinn führt.

3) Es wir vorausgesetzt, dass die Vereinigung VAR der Variablenmengen VAR_s jeweils disjunkt zu der Menge der Sortensymbole S und der Vereinigung der Operationssymbolmengen OPS ist. Es gilt also
$VAR=\cup_{s\in S}VAR_s$,
$OPS=\cup_{s\in S}OPS$,
$VAR\cap S=VAR\cap OPS=\varnothing$.

zur entsprechenden Sorte s bezeichnet. Die Menge der Variablen ist eine Teilmenge aller Terme. Die Vereinigung aller sortenspezifischen Variablenmengen ist

$$VAR=\cup_{s\in S}VAR_s.$$

Zudem darf keine Variable in zwei unterschiedlichen sortenspezifischen Variablenmengen enthalten sein:

$$\forall s_1,s_2\in S:\ s_1\neq s_2 \rightarrow VAR_{s.1}\cap VAR_{s.2}=\varnothing.$$

$TERM_s$ mit $s\in S$ ist die Menge aller *Terme* zu einer Sorte s. Die Familie aller Termmengen ist durch

$$TERMF=(TERM_s)_{s\in S}$$

gegeben. Die Vereinigung aller sortenspezifischen Termmengen ist

$$TERM=\cup_{s\in S}TERM_s.$$

Die sortenspezifischen Mengen aller Terme sind rekursiv[1)] definiert durch:

(1.) Jede Variable x aus einer s-spezifischen Variablenmenge VAR_s ist ein (atomarer)[2)] Term zur Sorte s:

$$VAR_s\subseteq TERM_s \text{ für alle } s\in S.$$

(2.) Jedes Konstantensymbol O, das ein null-stelliges Operationssymbol $O\in OPS$ mit der Typisierung $typ_{OPS}(O)=(\lambda,s)$ ist, ist ein (atomarer) Term zur Sorte s:

$$O\in TERM_s$$
$$\text{für alle } O\in OPS \text{ mit } typ_{OPS}(O)=\lambda,s$$
$$\text{und für alle } s\in S.$$

(3.) Jeder Ausdruck $O(t_1...t_n)$ ist ein Term zur Sorte s, wenn O ein Operationssymbol mit der Typisierung $typ_{OPS}(O)=(s_1...s_n,s)$ ist und $t_1,...,t_n$ mit $t_i\in TERM_{s.i}$ jeweils Terme zur Sorte s_i für alle $i\in\{1,...,n\}$ sind:

$$O(t_1,...,t_n)\in TERM_s$$
$$\text{für alle } O\in OPS \text{ mit } typ_{OPS}(O)=s_1...s_n,s \text{ und}$$
$$t_i\in TERM_i \text{ für alle } i=1,...,n.$$

---

1) Das Rekursionsprinzip wird später anhand rekursiver Regeln weiter dargestellt werden.

2) Atomare Terme werden auch als *Basisterme* bezeichnet (Vgl. ZELEWSKI (1995) Bd. 4 S. 36). Die Menge der Konstantensymbole $KON\subseteq OPS$ mit $KON=\{O\in OPS|ARG(O)=\lambda\}$ ist eine zur Menge VAR der Variablen komplementäre Teilmenge der Basisterme BT mit $BT=KON\cup VAR$. Dabei gilt die bereits oben aufgeführte Disjunktheit der Menge VAR aller Variablen und der Menge OPS aller Operationssymbole.

Der Typ jedes Terms ist gegeben als das Bild der Term-Typisierungsfunktion

$$typ_{TERM}:TERM \rightarrow S.$$

Für jeden Term $t \in TERM_s$ und jede Sorte $s \in S$ ist sein Typ gegeben als $typ_{TERM}(t)=s$.

Dadurch, dass Prädikate in der Termdefinition keine Berücksichtigung finden, sind sowohl für algebraische als auch für relationale Signaturen Terme gleich definiert.

Die Menge der Variablen, die in einem Term enthalten sind, wird durch die Bilder der Funktion:

$$V_{TERM}:TERM \rightarrow pot(VAR)$$

angegeben. Im Argumentbereich der Funktion V_{TERM} ist die Vereinigungsmenge $TERM=\cup_{s \in S}Term_s$ aller sortenspezifischen Termmengen enthalten. Der Zielbereich der Funktionsvorschrift ist die Potenzmenge[1] über der Menge aller Variablen. Die Bilder der Funktion sind wie folgt definiert:[2]

(1.) $V_{TERM}(x)=\{x\}$
mit $x \in VAR$

(2.) $V_{TERM}(O)=\varnothing$
für $O \in OPS$ mit $ARG_{OPS}(O)=\lambda$.

(3.) $V_{TERM}(O(t_1,...,t_n))=\cup_{1 \leq i \leq n}V_{TERM}(t_i)$
für $O \in OPS$ mit $typ_{OPS}(O)=s_1...s_n,s_{n+1}$ und $t_i \in TERM_{s,i}$ für alle i $1 \leq i \leq n$.

Terme, die keine Variable enthalten, werden als *konstante Terme*[3] oder *Grundterme*[4] bezeichnet. Solche Grundterme umfassen einerseits atomare Terme, die durch Operationssymbole mit einem leeren Vorbereich (Konstantensymbole) definiert sind, und andererseits aus zusammengesetzten Termen, die nur Konstantensymbole im Argumentbereich aufweisen.[5]

2.1.4.1.2 Auswertung von Termen

---

1) Die Potenzmenge *pot*(VAR) über der Menge aller Variablen ist definiert als:
$$pot(VAR)=\{VAR' | VAR' \subseteq VAR\}.$$

2) Vgl. KANEIWA (2001) S. 28.

3) Vgl. EHRICH ET AL. (1989) S. 19.

4) Vgl. EHRIG ET AL. (1999) S. 155.

5) Vgl. ZELEWSKI (1995) Bd. 4 S. 36.

Die Verknüpfung einer Signatur mit ihrer Algebra erfolgt durch das Prinzip der *Termauswertung*. Die Auswertung eines Terms erfolgt durch eine Menge von *Termauswertungsfunktionen*:

$$tausw_s:TERM_s \rightarrow OB_s \text{ für alle } s \in S$$

Die Termauswertungsfunktionen ordnen jedem Term $t \in TERM_s$ zu einer Sorte s genau ein formales Objekt $ob \in OBS_s$ aus dem Objektbereich zur selben Sorte s zu. Jedes solche formale Objekt stellt aus prädikatenlogischer Sicht eine Konstante dar, die – in Abgrenzung zur Prädikatenlogiken zweiter und höherer Ordnung – auch als *Individuenkonstanten* bezeichnet wird.

Die Familie aller Termauswertungsfunktionen ist gegeben als:

$$tauswf=(tausw_s:TERM_s \rightarrow OB_s)_{s \in S}.$$

Der Sortenindex einer Auswertungsfunktion schränkt die Menge der Terme im Vorbereich der Funktionsvorschrift auf sortenspezifische Terme ein. Durch die Identität der Indizes für die Menge $TERM_s$ der Terme im Vor- und die Menge OB_s der formalen Objekte im Nachbereich der Termauswertungsfunktion ist gewährleistet, dass die Auswertung eines Terms $t \in TERM_s$ zu einem formalen Objekt ob führt, das Element der Objektmenge OB_s zur selben Sorte s ist.[1)]

Für die Auswertung von Basistermen, die aus Variablen hervorgehen, ist zusätzlich die *Belegung* der Variablen mit formalen Objekten aus der Objektmenge einer Algebra notwendig. Eine *Variablenbelegung* ist eine Familie belf sortentreuer Funktionen bel_s :

$$belf=(bel_s:VAR_s \rightarrow OB_s)_{s \in S}.$$

Durch eine Variablenbelegung wird jeder sortenspezifischen Variablen ein Element der Objektmenge zur selben Sorte zugeordnet.

Eine Familie von Termauswertungsfunktionen ist bei gegebener Belegungsfunktion bel_s für alle Sorten $s \in S$ und eine vorgegebene Algebra $A_{AS}=(S,OPS)$ rekursiv wie folgt definiert:

(1.) $tausw_s(x)=bel_s(x)$

für jede Variable $x \in VAR_s$ und jede Sorte $s \in S$,

---

1) Dieser Aspekt der Termauswertung wird als ihre *Sortentreue* bezeichnet (vgl. ZELEWSKI (1995) Bd. 4 S. 45). Die Sortentreue der Termauswertung behält im Signatur-Konzept mit geordneten Sortenmengen (vgl. Abschnitt 2.2.1) ihre Bedeutung, obwohl dort Objektmengen zu einer Sorte s stets Teilmengen von Objektmengen sind, die einer Supersorte von s zugeordnet sind. Dies wird später ausführlicher dargestellt werden. In HEISE (2001) S. 27 wird der Begriff Sortentreue im Petri-Netz-Konzept mit der Forderung verknüpft, dass die Stellen eines Petri-Netzes nur Terme zu einer stellenspezifischen Sorte aufnehmen können.

(2.) $tausw_s(O)=ob$ mit $ob \in OB_s$

für jedes Konstantensymbol $O \in OPS$ mit der Typisierung $typ_{OPS}(O)=\lambda,s$

(3.) $tausw_s(O(t_1...t_n))=o(tausw_{s.1}(t_1),...,tausw_{s.n}(t_n))$

für jedes Operationssymbol $O \in OPS$ mit der Typisierung $typ_{OPS}(O)=s_1...s_n,s$ mit $s_1,...,s_n,s_{n+1} \in S$ und Termen $t_1,...,t_n$ mit $t_i \in TERM_{s.i}$ für jede Sorte $s_i \in S$

und $o \in OPF$ mit $o:OB_{s.1} \times ... \times OB_{s.n+1}$.

2.1.4.2 Formeln als Ausdrücke über relationalen Signaturen

2.1.4.2.1 Definition von Formeln

Terme zeichnen sich dadurch aus, dass ihnen mit Hilfe von Termauswertungsfunktionen jeweils genau ein formales Objekt aus einer Algebra zugeordnet werden kann. Terme, die nur aus Komponenten einer (algebraischen) Signatur aufgebaut sind, werden daher mit Hilfe von Termauswertungsfunktionen im Hinblick auf eine Signatur-spezifische Algebra interpretiert. Terme sind für den algebraischen Teil (S,OPS,typ_{OPS}) einer relationalen Signatur $(S,OPS,typ_{OPS},RS,typ_{RS})$ identisch definiert, da Relationssymbole bei der Konstruktion von Termen keine Rolle spielen.

Die zweite Klasse von Ausdrücken zeichnet sich dadurch aus, dass diesen Ausdrücken ein eindeutiger *Wahrheitswert* zugeordnet werden kann. Diese zweite Klasse der Ausdrücke beinhaltet *Formeln*[1]. Der Wahrheitswert von Formeln kann zwar grundsätzlich als ein formales Objekt aus einer zweielementigen Objektmenge $OB_{WW}=\{w,f\}$ aufgefasst werden, so dass sich alle Relationssymbole „im Prinzip“ als Operationssymbole besonderer Art mit der Zielsorte WW auffassen lassen. Wegen der besonderen Bedeutung von „Wahrheit“ in logischen Analysen ist es aber allgemein üblich, Relationssymbole und die daraus konstruierten Formeln als formale Konstrukte *sui generis* zu behandeln. Diesem Vorgehen wird auch hier gefolgt.

Die Menge FORM der Formeln über einer relationalen Signatur SIG_{RS} ist rekursiv definiert als:

---

1) *Formeln* die über Termen einer Signatur konstruiert werden, werden auch als *Regeln*, *Axiome* (EHRICH ET AL. (1989) S. 18) oder *Gesetze* bezeichnet. Der Begriff Regel wird hier reserviert für alle relationalen Ausdrücke, die für Ontologien definiert sind. Rein algebraische Regeln werden dann als Teilmenge der prädikatenlogischen Formeln aufgeführt werden.

(1.) Wenn $R \in RS$ ein Relationssymbol mit der Typisierung $typ_{RS}(R)=s_1...s_n$ ist und $t_1...t_n$ jeweils Terme $t_i \in TERM_{s,i}$ zur Sorte $s_i \in S$ mit $i \in \{1,...,n\}$ sind, dann ist $R(t_1,...,t_n)$ eine (atomare) Formel.[1)]

(2.) Wenn G eine Formel ist, dann ist auch $\neg G$ eine (zusammengesetzte) Formel.

(3.) Wenn G und H Formeln sind, dann sind auch $G \wedge H$, $G \vee H$, $G \rightarrow H$ und $G \leftrightarrow H$ (zusammengesetzte) Formeln.

(4.) Wenn G eine Formel ist und $x \in VAR$ eine Variable ist, dann sind sowohl $\exists x:G(x)$ als auch $\forall x:G(x)$ (zusammengesetzte) Formeln.

Tritt eine Formel G als Teil einer zusammengesetzten Formel G auf, dann wird G eine *Teilformel* von G genannt. Das Auftreten einer Variablen in einer Formel kann durch ihren *Freiheitsgrad* charakterisiert werden. Das Vorkommen einer Variablen x in einer Formel G wird als *gebunden* bezeichnet, wenn x in einer Teilformel H von G der Form $\forall x:H$ oder $\exists x:H$ auftaucht. Falls keine Teilformel existiert, die eine Variable bindet, wird das Vorkommen der Variablen x in der Formel G als *frei* bezeichnet. Die Menge der freien Variablen in einer Formel wird durch das Bild der Funktion

$$FV:FORM \rightarrow pot(VAR)$$

an der Stelle der jeweiligen Formel bestimmt. Die Bilder der Funktion sind wie folgt definiert:[2)]

(1.) $FV(R(t_1,...,t_n))=\cup_{1 \leq i \leq n} V_{TERM}(t_i)$ für $R \in RS$ mit $typ_{RS}(R)=s_1...s_n$.

(2.) $FV(\neg G)=FV(G)$

(3.) $FV(G \bullet H)=FV(G) \cup FV(H)$ für $\bullet \in \{\vee,\wedge,\rightarrow,\leftrightarrow\}$

(4.) $FV(\bullet G)=FV(G) \setminus \{x\}$ für $\bullet \in \{\forall x:G(x),\exists x:G(x)\}$.

Eine Formel G wird als *Aussage* bezeichnet, wenn sie keine frei Variablen enthält und somit $FV(G)=\varnothing$ gilt. Eine Formel wird als *Grundformel* bezeichnet, wenn sie keine Variablen und keine Quantoren enthält. Alle atomaren Formeln sind immer Grundformeln. Zusammengesetzte Grundformeln, sind solche Formeln deren atomare Formelbestandteile Grundformeln sind.

---

1) Als Sonderfall kommt auch eine *aussagenlogische* Formel in Frage, wenn das r ein Relationssymbol mit der Typisierung $typ_{RS}(r)=\lambda$ ist.

2) Vgl. KANEIWA (2001) S. 29.

2.1.4.2.2 Auswertung von Formeln

Die Auswertung von Formeln erfolgt durch eine *Formelauswertungsfunktion* mit der Funktionsvorschrift:[1)]

$$\text{fausw}:\text{FORM}_{RS} \rightarrow \{w, f\}.$$

Das Bild der Formelauswertungsfunktion zu einer Formel gibt ihren *Wahrheitsgehalt* an. Die Bilder w und f werden dabei als „wahr“ bzw. „falsch“ interpretiert.[2)] Dadurch erfolgt eine Verknüpfung von Signaturen und Algebren, wie sie bereits durch das Prinzip der Termauswertung aus Abschnitt 2.1.4.1.2 umgesetzt werden kann. Diese Verknüpfung wird teilweise als Überbrückung zwischen Syntax und Semantik aufgefasst. Der Wahrheitswert einer Formel G aus der Formelmenge FORM_{RS} über einer Signatur SIG_{RS} ist bestimmt durch die (relationale) Algebra A_{RS}, mit deren Komponenten die Formel interpretiert wird. Die Algebra A_{RS} ist die formale Semantik der rein syntaktisch über einer relationalen Signatur SIG_{RS} konstruierten Formeln. Es erfolgt somit eine Erweiterung der Syntax, die durch die Signatur SIG_{RS} vorgegeben ist, um die Dimension der *Wahrheit*. Der Wahrheitsbegriff wird durch die Menge $\{w, f\}$ ebenso formal gegriffen. Die Bedeutung einer Formel ist demnach ihr Wahrheitswert. Der zweite Aspekt der formalen Semantik wurde bereits zuvor durch das Prinzip der Termauswertung vorgestellt. Die Bedeutung eines Terms war dort ein formales Objekt aus einer Objektmenge.

Für alle Formeln $G_1, G_2, G_3 \in \text{FORM}_{RS}$ gilt:

(1.) $\text{fausw}(G_1)=w$,
wenn $G_1=r(t_1,...,t_n)$
mit $R \in RS$, $\text{typ}_{RS}(R)=s_1...s_n$, $t_i \in \text{TERM}_{s.i}$, $i=1,...,n$
und $(\text{tausw}_{s.1}(t_1),...,\text{tausw}_{s.n}(t_n)) \in r_A$;
sonst $\text{fausw}(G_1)=f$.

(2.) $\text{fausw}(G_1)=w$,
wenn $G_1=(\neg G_2)$
und $\text{fausw}(G_2)=f$,
sonst $\text{fausw}(G_1)=f$,

---

1) Vgl. KREOWSKI (1991) S. 57 f.

2) Die Elemente w und f der Menge BOOL=$\{w, f\}$ werden teilweise auch mit „1“ und „0“ oder „ja“ und „nein“ bezeichnet (vgl. KREOWSKI (1991) S. 57).

(3.) $fausw(G_1)=w$,

wenn $G_1=(G_1 \wedge G_2)$

und sowohl

$fausw(G_2)=w$

als auch

$fausw(G_3)=w$;

sonst $fausw(G_1)=f$,

(4.) $fausw(G_1)=f$,

wenn $G_1=(G_2 \vee G_3)$

und sowohl

$fausw(G_2)=f$

als auch

$fausw(G_3)=f$;

sonst $fausw(G_1)=w$,

(5.) $fausw(G_1)=f$,

wenn $G_1=(G_2 \rightarrow G_3)$

und $fausw(G_2)=w$

und $fausw(G_3)=f$,

sonst $fausw(G_1)=w$,

(6.) $fausw(G_1)=w$,

wenn $G_1=G_2 \leftrightarrow G_3$

und $fausw(G_2)=fausw(G_3)$;

sonst $fausw(G_1)=f$,

(7.) $fausw(G_1)=w$,

wenn $G_1=\forall x:G_2(x)$

und $fausw(G_2(x))=w$

für alle Variablenbelegung bel[x/ob][1)], die der Variablen $x \in VAR_s$ jeweils ein

1) Die gesonderte Belegung einer Variablen $x \in VAR_s$ mit einem Wert $ob \in OB_s$, während alle anderen Variablen ihr Bild aus der Variablenbelegungsfunktion bel_s beibehalten, wird mit der Abbildung bel[x/a] angegeben. Sie ist definiert als:
$bel[x/ob](y)=ob$, wenn $y=x$ und
$bel[x/ob](y)=bel_s(y)$, wenn $y \neq x$.

beliebiges formales Objekt $ob \in OB_s$ zuordnen;

sonst $fausw(G_1)=f$.

(8.) $fausw(G_1)=w$,

wenn $G_1=\exists x:G_2(x)$, und $fausw(G_2)=w$

für alle Variablenbelegungen bel[x/ob] die der Variablen $x \in VAR_s$ jeweils ein beliebiges formales Objekt $ob \in OB_s$ zuordnen;

sonst $fausw(G_1)=f$.

Analog zu der Formelauswertungsfunktion kann eine *Folgerungsrelation* definiert werden. Sie gibt für ein Algebra-Formel-Tupel an, ob die Formel aus der Algebra abgeleitet werden kann. Die *Folgerungsrelation* $\models_{bel} \subseteq \mathcal{ALG}(SIG_{RS}) \times FORM_{RS}$ gibt an, ob aus einer Algebra $A_{RS} \in \mathcal{ALG}(SIG_{RS})$ zu einer Signatur SIG_{RS} bei gegebener Belegungsfunktion bel eine Formel $F \in FORM_{RS}$ zu dieser Signatur folgt. Sie ist wie folgt definiert:[1)]

(1.) $A_{SIG} \models_{bel} R(t_1,...,t_n)$ gilt genau dann, wenn $(tausw(t_1),...,tausw(t_n)) \in r$.

(2.) $A_{SIG} \models_{bel} \neg F$ gilt genau dann, wenn $A_{SIG} \models_{bel} F$ nicht gilt.

(3.) $A_{SIG} \models_{bel} F \wedge G$ gilt genau dann, wenn $A_{SIG} \models_{bel} F$ und $A_{SIG} \models_{bel} G$ gelten.

(4.) $A_{SIG} \models_{bel} F \vee G$ gilt genau dann, wenn $A_{SIG} \models_{bel} F$ oder $A_{SIG} \models_{bel} G$ gelten.

(5.) $A_{SIG} \models_{bel} F \rightarrow G$ gilt genau dann, wenn nicht $A_{SIG} \models_{bel} F$ gilt oder $A_{SIG} \models_{bel} G$ gilt.

(6.) $A_{SIG} \models_{bel} F \leftrightarrow G$ gilt genau dann, wenn sowohl $A_{SIG} \models_{bel} F \rightarrow G$ als auch $A_{SIG} \models_{bel} G \rightarrow F$ gelten.

(7.) $A_{SIG} \models_{bel} \forall x:F$ gilt genau dann, wenn für jedes $ob \in OB$ gilt: $A_{SIG} \models_{bel[x/a]} F$.

(8.) $A_{SIG} \models_{bel} \exists x:F$ gilt genau dann, wenn für mindestens ein $ob \in OB$ gilt: $A_{SIG} \models_{bel[x/a]} F$.

Eine Formel $F \in FORM_{RS}$ ist *gültig* in einer Algebra $A_{SIG} \in \mathcal{ALG}(SIG)$, wenn bei *jeder* Variablenbelegung bel die Relation $A_{SIG} \models_{bel} F$ gilt. Die Gültigkeitsbeziehung kann auf Formelmengen ausgeweitet werden. So wird eine Formelmenge $FM \subseteq FORM_{RS}$ als gültig in einer Algebra A_{RS} bezeichnet, wenn jede Formel $F \in FM$ bei jeder Variablenbelegung gültig in der Algebra A_{RS} ist.

---

1) Vgl. EHRIG ET AL. (1999) S. 320.; SPERSCHNEIDER/HAMMER (1998) S. 18 f.

Eine Formel $F \in FORM_{RS}$ ist *erfüllbar*, wenn es *mindestens eine* Algebra $A_{RS} \in \mathcal{ALG}(SIG)$ gibt, in der sie gültig ist. Die Formel $F \in FORM_{RS}$ ist *allgemeingültig*, wenn sie in *jeder* Algebra $A_{RS} \in \mathcal{ALG}(SIG)$ gültig ist. Die Formel $F \in FORM_{RS}$ ist *kontradiktorisch*, wenn sie in keiner Algebra $A_{RS} \in \mathcal{ALG}(SIG)$ gültig ist.

2.1.4.3 Multimengen

Mit einer *Multimenge*[1] wird zugelassen, dass ein Element mehrfach in einer Menge enthalten ist.[2] Dies geschieht, indem jedem Element a einer „Trägermenge" A eine natürliche Zahl $n_a \in \mathcal{N}$ zugeordnet wird, die anzeigt, wie oft das Element a in der Menge A enthalten ist. n wird dann als Multiplizität des jeweiligen Elements a bezeichnet. Das Konzept der Multimengen wird in höheren Petri-Netzen benötigt, um das mehrfache Vorkommen identischer Kopien eines Objektes auf Stellen zu kennzeichnen.[3]

Multimengen sind – neben der Menge der Formeln und der Menge der Terme – die dritte Ausdrucksvariante, von der im Integrationskonzept Gebrauch gemacht wird. Es wurde bereits vorher darauf hingewiesen, dass die drei Ausdrucksvarianten nicht unabhängig voneinander existieren. Sie sind jedoch paarweise disjunkt, da kein Ausdruckselement in einer anderen Ausdrucksmenge vorkommen kann. Wie noch zu sehen sein wird, werden durch die Konstruktionsregel für Multimengen syntaktische Konstrukte eingeführt, die weder für Formeln noch für Terme vorgesehen sind.

Für eine beliebige Menge A ist ihre Multimenge durch die Funktionsvorschrift

$$mult: A \rightarrow \mathcal{N}$$

definiert. Sie wird als *Multimenge über der Trägermenge B* bezeichnet. Der Funktionswert mult(a) für ein Element $a \in A$ aus der Menge A gibt dessen *Multiplizität* wieder. Die Menge aller Multimengen über der Menge A wird mit $\mathcal{MULT}(A)$ angesprochen. Bei einer gegebenen Multimenge $mult \in \mathcal{MULT}(A)$ bezeichnet mult(a)=0, dass a keine Element der Menge A ist:

---

1) Zum Konzept der *Multimengen* vgl. GENRICH (1986) S. 214 ff.; GIRAULT/VALK (2003) S. 43; KORCZYNSKI ET AL. (1990) S. 37 ff.; REISIG (1991A) S. 150 ff.; ZELEWSKI (1995) Bd. 4, S. 256. Vgl. KIFER ET AL. (1995) S. 820 f. für eine Umsetzung von Multimengen in F-Logic.

2) Das mehrfache Vorkommen von Elementen in Mengen wurde bereits früher für Mengenfamilien zugelassen. Dort erfolgt allerdings eine Differenzierung der Mitgliedsmengen durch ein Indexierung der Mitglieder. Das ist zwar grundsätzlich für Multimengen nicht vorgesehen, hat aber keinen gegenseitigen Ausschluss der beiden Konzepte zur Folge. Denn jede Mengenfamilie ist auch eine Multimenge.

3) Das mehrfache Vorkommen eines Termtupels in der Markierung einer Stelle wird in Ontologie-Netzen durch eindeutige Identifikatoren für Objekte ausgeschlossen.

$a \notin A$. Umgekehrt gilt $mult(a) \geq 1$, wenn $a \in A$ ist. Mit $\varnothing_M$ wird die *leere Multimenge* bezeichnet. Es gilt für alle $a \in A$:

$$\varnothing_M(a)=0.$$

Summen und Differenzen zweier Multimengen $mult_1, mult_2 \in \mathcal{MULT}(A)$ über einer Trägermenge A werden dann durch Summen bzw. Differenzen der Multiplizitäten der betroffenen Multimengen für jedes Element $a \in A$ aus der gemeinsam zu Grunde liegenden Trägermenge A wie folgt definiert:

- $(mult_1+mult_2)(a) = mult_1(a)+mult_2(a)$
- $mult_1 \leq mult_2 \Leftrightarrow \forall a \in A: mult_1(a) \leq mult_2(a)$
- $(mult_1-mult_2)(a) = mult_1(a)-mult_2(a)$ wenn $mult_2 \leq mult_1$
- $(mult_1 \bullet mult_2)(a)=mult_1(a) \bullet mult_2(a)$

Um Multimengen ausdrücken zu können, bieten sich grundsätzlich mehrere Möglichkeiten an. Eine explizite Trennung der Ausdrücke, mit denen Multimengen beschrieben werden, erfolgt z.B. bei *algebraischen* Petri-Netzen. Neben der „originären" Signatur wird dort zusätzlich eine „derivative" *Multimengensignatur* konstruiert.[1)] Jeder Sorte der originären Signatur wird in einer Multimengensignatur eine Multimengensorte zugewiesen. Zudem werden die semantischen Anforderungen an Multimengen in Form von Anforderungen formuliert, so dass die Multimengensignatur zu einer *Multimengenspezifikation* ausgeweitet wird. Für Prädikat/Transition-Netze ist die Konstruktion von Multimengen auf der Basis einer Multimengenspezifikation nicht üblich. Daher werden drei Ausdruckvarianten vorgestellt, die mit Prädikat-Transition-Netzen öfter in Einklang gebracht werden.

Die erste Variante wird als *binäre Mengennotation* bezeichnet.[2)] Eine Multimenge mult über einer Trägermenge A wird dann über eine konventionelle Relation

$$Mult_A=\{(mult(a),a) | a \in A\} \subseteq (\mathcal{N} \times A)$$

formuliert. Jedes Element (mult(a),a) der Relation $Mult_A$ gibt in seiner ersten Komponente mult(a) die Multiplizität des Elementes a aus der zweiten Komponente der Relation an. Jedes Element $a \in A$ muss genau einmal in einem Tupel als zweite Komponente enthalten sein. Die natürliche Zahl, die als erste Komponente im Tupel vorkommt, kann hingegen in unterschied-

1) Vgl. AOUMEUR (2001) S. 35; REISIG (1991B) S. 143.

2) Vgl. ZELEWSKI (1995) Bd. 4 S. 259 f.

lichen Tupeln vorkommen. Denn zwei Elementen $a,b \in A$ mit $a \neq b$ kann die gleiche Multiplizität $mult(a)=mult(b)$ zugeordnet sein.

In der zweiten Variante werden Multimengen als *Listen* in der Form mult=[...] ausgedrückt.[1] Dabei wird der leeren Multimenge $\varnothing_M$ die leere Liste [] zugeordnet. Das mult(a)-fache Vorkommen des Elements a in der Multimenge mult kann in der Liste durch die mult(a)-fache Notation von a erfolgen. In einer „multiplikativen Listennotation" kann auch abkürzend mult(a)*a notiert werden, wenn mult(a)>1 ist. Listen sind jedoch ungeordnet und daher wirkt diese Notation unübersichtlich, wenn aufgrund der Permutation äquivalente Multimengen in unterschiedlichen Listen dargestellt werden.

In der dritten Ausdrucksvariante werden Multimengen als *formale Summen* notiert.[2] Eine Multimenge $mult \in \mathcal{MULT}(A)$ mit $A=\{a_1,...,a_n\}$ wird dann als formale Summe der Form

$$mult=m_1a_1+...+m_na_n$$
$$=\Sigma_{i\in\{1,...,n\}}m_ia_i$$

mit $m_i=mult(a_i)$ angegeben. Dabei werden nur Summanden mit $m_i \neq 0$ aufgeführt. Die leere Multimenge $\varnothing_M$ wird als 0 notiert.

Im Folgenden wird das Prinzip der Multimengen und ihrer Notation als formale Summen mittels eines *Beispiels* verdeutlicht. Für die Trägermenge $A=\{a,b,c\}$ seien die beiden Multimengen $mult_1=\{a,a,a,b,b,c,c\}$ und $mult_2=\{a,b,b,c\}$ gegeben. Sie können notiert werden als $mult_1=3a+2b+2c$ und $mult_2=a+2b+c$. Die Summe der beiden Multimengen lautet $mult_1+mult_2=4a+4b+3c$. Die Differenz der beiden Multimengen lautet $mult_1-mult_2=a+c$.

Die *Kardinalität* |mult| einer Multimenge mult entspricht der Summe der Multiplizitäten für jedes Element der zu Grunde gelegten Menge:

$$|mult|=\Sigma_{a.i\in A}mult(a_i).$$

Für die oben angegebenen Multimengen lauten die Kardinalitäten $|mult_1|=3+2+2=7$ und $|mult_2|=1+2+1=4$.

Mit Multimengen, die über den sortenspezifischen Objektmengen einer Algebra definiert sind, werden später die Stellen von Petri-Netzen markiert. Mit Multimenge, die über sortenspezifischen Termmengen definiert sind, werden die Flussrelationen beschriftet.

---

1) Vgl. ZELEWSKI (1995) Bd. 4 S. 260.

2) Vgl. GENRICH (1986) S. 215; GIRAULT/VALK (2003) S. 43; KORCZYNSKI ET AL. (1990) S. 37. ZELEWSKI Bd. 4 (1995) S. 263.

2.1.5 Spezifikationen

Eine Signatur SIG_{RS} gibt lediglich einen formalen Rahmen vor, dem alle zugehörige Algebren $A_{RS} \in \mathcal{ALG}(SIG_{RS})$ entsprechen müssen. Über die Sorten-, Operationssymbol- und Relationssymboldeklarationen hinaus wird in einer Signatur keine inhaltliche Einschränkung für ihre Algebren getroffen. Die Interpretation einer Signatur durch eine Algebra entspricht daher der *losen* Semantik der Signatur, wenn keine weiteren Einschränkungen definiert sind.[1)] Sie ist in dem Sinne lose, als dass unterschiedliche Algebren zu einer Signatur gegeben sein können, die auch unterschiedliche Objektmengen beinhalten. Eine weitere Eingrenzung der in Frage kommenden Algebren für eine Signatur erfolgt durch ihre Ausdehnung um eine Teilmenge der Menge $FORM_{RS}$ aller Formeln, die auch als *Anforderungen* ANF mit $ANF \subseteq FORM_{RS}$ angesprochen werden. Die Ausdehnung der Signatur SIG_{RS} um eine Menge ANF von Formeln wird dann als *Spezifikation* SPEZ bezeichnet. Die Elemente der Formelmenge ANF haben den Charakter von Restriktionen, die seitens jeder Algebra $A_{RS} \in \mathcal{ALG}(SIG_{RS})$ erfüllt sein müssen, um konform mit der Spezifikation SPEZ zu sein.

Eine Signatur erweitert um Anforderungen ANF und einen Variablenvorrat VAR, der für die Spezifikation benötigt wird, wird auch *Spezifikation* genannt[2)]. Sie entspricht einem Tripel der Form

$$SPEZ=(SIG_{RS},VAR,ANF)$$

Die Komponenten einer Spezifikation sind

(1.) eine *Signatur* SIG_{RS},

(2.) eine Menge von *Variablen* VAR und

(3.) eine Menge von *Anforderungen* ANF.

---

1) Vgl. EHRIG ET AL. (1999) S. 182.

2) Vgl. KREOWSKI (1991) S. 67.

Eine Algebra A_{RS} wird als *Modell*[1)] einer Spezifikation bezeichnet, wenn jede Formel $F \in ANF$ in der Algebra A_{RS} gültig ist.

Eine Spezifikation kann schematisch aufgeführt werden. Es werden für die Komponenten der Spezifikation jeweils Abschnitte aufgeführt.

Spez_Name
Sorten:
$s_1 \ldots s_n$
Operationssymbole:
$o_1: \rightarrow s$
...
$o_j: s_1 \ldots s_n \rightarrow s$
Relationssymbole:
$pr_1 < s >$

...
$pr_k < s_n >$
Anforderungen:
anf_l
...
anf_l

1) Das hier unterlegte *Modell*verständnis unterscheidet sich von dem üblicherweise in den Wirtschaftswissenschaften verwendeten Modellverständnis in vielfacher Weise. Die Modellverständnisse einzelner Wissenschaftsdisziplinen werden öfter hinsichtlich ihrer *epistemischen* Positionierung klassifiziert. In der *Betriebswirtschaftslehre* dominieren das *abbildungs*- und das *konstruktionsorientierte* Modellverständnis (vgl. RUPPRECHT (2002) S. 12 ff.; SCHÜTTE (1998) S. 45 ff.; WOLF (2001) S. 43 ff.). Die begrifflich Trennung erscheint jedoch artifiziell, wenn angeführt wird, dass auch das konstruktionsorientierte Modellverständnis von einer Abbildung ausgeht. Es werden allerdings nicht *direkte* Abbilder der Realität angenommen, wie es vom abbildungsortientierten Modellverständnis vorausgesetzt wird, sondern Abbildungen von mentalen Modellen der Realität. Diese mentalen Modelle entsprechen einer subjektiven Konstruktionsleistung des Modellierers. Somit entsprechen Modelle im konstruktionsorientierten Modellverständnis *indirekten* Abbildern der Realität.
In der *Wirtschaftsinformatik* wird vorwiegend dem abbildungsorientierten Modellverständnis gefolgt (vgl. z.B. FERSTL/SINZ (1998) S. 18; LEHNER ET AL. (1995) S. 27; HANSEN/NEUMANN (2001) S. 994). Dem konstruktionsorientierten Modellverständnis wird insbesondere im Themenfeld *Referenzmodelle* gefolgt. Referenzmodelle haben grundsätzlich eine *normative* Komponente. Sie geben stets *Empfehlungen* dazu, wie Informationssysteme aussehen *sollten* (vgl. SCHÜTTE (1998) S. 69). Sie werden deswegen öfter mit Ontologien in Einklang gebracht. Hinsichtlich des normativen Aspekts von Ontologien erfolgt hier allerdings keine Einschränkung. Ontologien können potenziell auch als Referenzmodelle Verwendung finden. Es bedarf hierzu lediglich ihrer Auszeichnung als *normative* Begriffsysteme.
Für die vorliegende Arbeit wird hinsichtlich der epistemischen Grundhaltung das konstruktionsorientierte Modellverständnis unterstellt. Allerdings wird dabei nicht vorausgesetzt, dass die Konstruktion des Modellierers einer empirisch beobachtbaren Realität folgt. Die Konstruktion ist auf eine *Problemwahrnehmung* des Modellierers zurückzuführen, die unabhängig von der Realität gegeben sein kann. Somit entspricht das hiesige Konzept einem *problemorientierten Modellierungsverständnis* (vgl. ZELEWSKI (1995) Bd. 2, S. 19). Durch die obige Einschränkung wird in diesem formalwissenschaftlichen Ansatz eine konkrete *Anforderung* an Modelle gestellt: Es sind nämlich nur jene Algebren auch Modelle, die in Folgerungsrelation zu einer Formelmenge stehen. Als Modell im Sinne der Wirtschaftsinformatik kommen in der vorliegenden Arbeit jene formalsprachlichen Konstrukte in Betracht, die mit den Komponenten einer Signatur konstruiert und mittels einer Algebra interpretiert werden.

2.2 Formale Präzisierung des Verständnisses von Ontologien

2.2.1 Ontologie-Signaturen

2.2.1.1 Definition von Ontologie-Signaturen

Der Begriff *Ontologie* wird bisweilen mit sehr unterschiedlichen Verständnissen verbunden. Zumeist wird allerdings das jeweilige Verständnis natürlichsprachlich angedeutet und in den „günstigsten" Fällen mit exemplarischen Formalisierungen verdeutlicht. Eine solche Vorgehensweise birgt die Gefahr, den Begriff zu „verwässern" und gegenüber alternativen Konzeptionen keine klare Trennschärfe zu bieten. Um für die vorliegende Arbeit eine präzise Argumentationsgrundlage zu schaffen, wird daher das Verständnis, das mit dem Begriff „Ontologie" verbunden wird, *formal* festgelegt. Dabei wird auf das zuvor aufgeführte algebraische Signaturkonzept zurückgegriffen. Im Vergleich zum Signaturkonzept erfolgt allerdings einerseits eine *Einschränkung*. Sie betrifft Deklarationsmöglichkeiten für Operationssymbole. Andererseits erfolgt eine *Ausweitung*. Es werden Modellierungsprimitive eingeführt, die den Charakter einer Logik zweiter Ordnung in sich bergen, da sie Eigenschaften von wiederum Eigenschafts-beschreibenden Modellierungsprimitiven beschreiben.

Für die Konstruktion einer Ontologie wird eine *ontologische Signatur*[1)] benötigt. Eine ontologische Signatur SIG_{OS} ist definiert als das 13-Tupel

$$SIG_{OS}=(S_{OS},men,\leq_S,=_S,||_S,bez,def,OPS_{OS},inv_{OPS},typ_{OPS},min,max,RS_{OS}).$$

Die Bestandteile jeder ontologischen Signatur SIG_{OS} sind:

(1.) eine Menge von *Konzepten*

$$S_{OS}=S_{EW}\cup S_{MW}$$
$$\text{mit } S_{EW}\cap S_{MW}=\varnothing$$
$$\text{und } S_D\subseteq S_{EW}$$

(2.) eine *Mengenfunktion*

$$men: S_{EW} \rightarrow S_{MS},$$

(3.) eine *Subkonzeptrelation*

$$\leq_S \subseteq S_{OS} \times S_{OS},$$

1) Die Begriffe *ontologische Signatur* und *Ontologie-Signatur* werden synoynm verwendet.

(4.) eine *Äquivalenzrelation*

$$=_S \subseteq S_{OS} \times S_{OS},$$

(5.) eine *Exklusivitätsrelation*

$$||_S \subseteq S_{OS} \times S_{OS},$$

(6.) eine Familie von *Bezeichnungsfunktionen*

$$bez=(bez_{lan})_{lan \in \{ger,eng,fr,...\}}$$

$$\text{mit } bez_{lan}=S_{OS} \times pot(ALPH^*) \text{ für } lan \in \{ger,eng,fr,...\},$$

(7.) eine Familie von *Definitionsfunktionen*

$$def=(def_{lan})_{lan \in \{ger,eng,fr...\}}$$

$$\text{mit } def_{lan}=S_{OS} \times ALPH^* \text{ für } lan \in \{ger,eng,fr,...\},$$

(8.) eine Menge von *Attributssymbolen*

$$OPS_{OS}=OPS_{ST} \cup OPS_{SYM} \cup OPS_{TR},$$

(9.) eine *Inverserelation*

$$inv_{OPS} \subseteq OPS_{OS} \times OPS_{OS},$$

(10.) eine *Typisierungsfunktion*

$$typ_{OPS\text{-}ONT}: OPS_{OS} \rightarrow S_{OS} \times S_{OS},$$

(11.) eine *Minimum-Kardinalitätsfunktion*

$$min: OPS_{OS} \times \mathcal{N}$$

(12.) eine *Maximum-Kardinalitätsfunktion*

$$max: OPS_{OS} \times \mathcal{N}_+.$$

(13.) eine Menge von *Ontologie-Relationssymbolen*

$RS_{OS}=\{$ equal,element_of
add,subt,multip,div,concat,cut
greater,greater_or_equal$\}$.

Dabei sind die Elemente der Konzeptmenge S_{OS}, die Elemente der Attributssymbolmenge OPS_{OS} und die Elemente der Ontologie-Relationssymbolmenge RS_{OS} *objektsprachliche* Ausdrücke. Bei den restlichen Komponente einer Ontologie-Signatur SIG_{OS} handelt es sich um *metasprachliche* Ausdrücke.

2.2.1.2 Komponenten von Ontologie-Signaturen S_{OS}

2.2.1.2.1 Konzepte S_{OS}

Die Menge S_{OS} der *Konzepte* ist definiert durch

$$S_{OS}=S_{EW}\cup S_{MW},$$

mit $$S_{EW}\cap S_{MW}=\varnothing$$

$$S_D\subseteq S_{EW}$$

Die Menge S_{OS} der Konzepte ist in die zueinander disjunkten Konzeptmengen S_{EW} und S_{MW} unterteilt. Bei den Elementen der Menge S_{EW} handelt es sich um „originären" Konzepte. Sie werden als *einwertige* Konzepte angesprochen. Die Menge der einwertigen Konzepte S_{EW} hat als Teilmenge[1)] die Menge S_D der *Datenkonzepte*. Die Menge der Datenkonzepte S_D ist[2)]:

$$S_D=\{\text{Int, Integer, Int\_pois, Int\_neg, Real, Real\_pos, Real\_neg, Bool, String}\}.$$

Datenkonzepte werden extensional durch *primitive Datenmengen* interpretiert, die üblicherweise in höheren Programmiersprachen vorausgesetzt werden. Durch die Datenkonzepte werden die Menge der ganzen Zahlen (Integer), die Menge der positiven ganzen Zahlen (OB_{Int_pos}), die Menge der positiven ganzen Zahlen (OB_{Int_neg}) die Menge der reellen Zahlen (OB_{Real}), die Menge der positiven reellen Zahlen (OB_{Real_pos}), die Menge der negativen reellen Zahlen (OB_{Real_neg}), die Menge der Wahrheitswerte (OB_{Bool}) und die Menge der Zeichenketten (OB_{String}) bezeichnet.

Die Elemente der Menge S_{MW} sind hingegen *derivative* Konzepte. Sie werden von den originären Konzepten aus der Menge S_{EW} abgeleitet. Die Menge S_{MW} der mengenwertigen Konzepte ist:

---

1) Es muss sich nicht notwendigerweise um eine *echte* Teilmenge handeln, da auch Ontologien mit lediglich Konzepten aus der Menge S_D denkbar sind.

2) Vgl. PATIG (2001) S. 64.

$$S_{MW}=\{men(s)|s\in S_{EW}\} \text{ mit}$$
$$men: S_{EW} \rightarrow S_{MW}.^{1)}$$

Die Elemente der Menge S_{EW} werden als *mengenwertige* Konzepte bezeichnet.[2)] Es handelt sich bei der der Funktion men um eine *partielle* Funktion. Es wird nämlich nicht von jedem einwertigen Konzept ein mengenwenwertiges Konzept abgeleitet. Aus wird nur dann von einem einwertigen Konzept s ein einwertiges Konzepte men(s) abgeleitet, wenn es notwendig ist, im Modell auch Mengen von formalen Objekten aus einer s-spezifischen Objektmenge zu berücksichtigen. Somit ist die Mächtigkeit $|S_{MS}|$ der Menge S_{MS} maximal so groß wie die Mächtigkeit $|S_{EW}|$ der Menge S_{EW}. men ist zudem eine *injektive* Funktion. Es wird nämlich *jedem* Konzept $s\in S_{OS}$ *höchstens* ein mengenwertiges Konzept men(s) zugeordnet. Somit gilt:

$$\forall s_1,s_2\in S_{EW},\ s\in S_{MW}: men(s_1)=s \wedge men(s_2)=s \rightarrow s_1=s_2.$$

Es handelt sich ebenso um eine *surjektive* Funktion, da für jedes mengenwertige Konzept gilt:

$$\forall s_2\in S_{MS}\ \exists s_1\in S_{EW}: men(s_1)=s_2.$$

Somit handelt es sich bei men um eine partielle, *bijektive* Funktion.

Durch den Ausschluss von mengenwertigen Konzepten im Vorbereich der Funktion men können keine mengenwertigen Konzepten von wiederum mengenwertigen Konzepten abgeleitet werden. Es können nur mengenwertige Konzepten zu einwertigen Konzepten aus der Menge $S_{EW}=S_{OS}\backslash S_{MW}$ abgeleitet werden.

2.2.1.2.2 Strukturierungsrelationen $\leq_S$, $=_S$ und $||_S$

Die *Strukturierungsrelationen* $\leq_S$,$=_S$ und $||_S$ sind weitere Differenzierungsmerkmale gegenüber dem traditionellen Signaturkonzept. Mit Hilfe der Strukturierungsrelationen wird es ermöglicht, vereinfachte Ausdrücke zu formulieren, um Anforderungen an Modelle einer Ontologie

---

1) Mengensorten werden im Signaturkonzept oft durch listenartige Darstellungen umgesetzt. Diese Ausdrucksvariante kommt einer informationstechnischen Implementierung mit dem oft vorgegebenen Datentypen „Array“ am nächsten. Dafür wird in einer Signatur zunächst die leere Liste „[]“ als Konstantensymbol zu einer Mengensorte aufgeführt. Der Anschluss weiterer Terme an die Liste wird dann durch ein entsprechendes Operationssymbol umgesetzt (vgl. z.B. MÜLLER (1999) S. 20):
S: {s,men};
OPS: []: men(s)
anhaengen: s men(s) → men(s).
Die Listennotation wird teilweise auch zum Ausdrücken von *Multimengen* verwendet. Multimengen werden später noch vorgestellt werden (vgl. Abschnitt 2.1.4.3).

2) Die mengenwertige Konzepte sind für das hier vorgestellte Verständnis für Ontologien nicht notwendig. Jede Menge von Objekten kann nämlich durch ein Objekt repräsentiert werden, dass mittels Attributen mit den einzelnen Objekten verknüpft ist. Dieses Vorgehen entspricht der *Reifizierung* der Menge. Die Reifizierung wird später im Kontext von Kompetenzaussagen noch einmal verdeutlicht werden.

zu richten. Sie sind in der Hinsicht vereinfachend, dass ihre Funktionalität auch über *Formeln* gewährleistet werden könnte. In diesem Fall müsste allerdings für jede konkrete Relation, die zwischen zwei Konzepten besteht, eine Formel konstruiert werden um die gleiche Funktionalität[1)] zu gewährleisten. Mit Hilfe der Strukturierungsrelationen werden Eigenschaften von Konzeptpaaren kompakter formuliert als durch Formeln. Da die Semantik der Strukturierungsrelationen vorgegeben ist, muss ihre Funktionalität nicht für jedes betroffene Konteptpaar einzeln beschrieben werden.

Auf der Konzeptmenge S_{OS} ist eine *partielle Ordnung* durch die *Subkonzeptrelation* $\leq_S \subseteq S_{OS} \times S_{OS}$ definiert.[2)] Es handelt sich dabei um Konzeptpaare, bei denen die Extension des ersten Konzeptes eine Teilmenge der Extension des zweiten Konzeptes ist. Die partielle Ordnung drückt sich in den folgenden Eigenschaften der *Subkonzeptrelation* $\leq_S$ aus:

Reflexivität $\forall s \in S_{OS}: s \leq_S s$

Antisymmetrie $\forall s_1,s_2 \in S_{OS}: s_1 \leq_S s_2 \wedge s_2 \leq_S s_1 \rightarrow s_1 =_S s_2$[3)]

Transitivität $\forall s_1,s_2,s_3 \in S_{OS}: s_1 \leq_S s_2 \wedge s_2 \leq_S s_3 \rightarrow s_1 \leq_S s_3$.

Stehen zwei Konzepte $s_1,s_2 \in S_{OS}$ in Relation $\leq_S$ zueinander ($s_1 \leq_S s_2$), wird s_1 als *Sub-* oder *Unterkonzept* des Konzeptes s_2 bezeichnet. Das Konzept s_2 ist dann ein *Super-* oder *Oberkonzept* des Konzeptes s_1. Aufgrund der Transitivität der *Subkonzeptrelation* müssen die beiden kon-

1) Die Semantik der Strukturierungsrelationen und ihre damit verbundene Funktionalität wird bei der Definition von *Ontologie-Algebren* weiter unten ersichtlich.

2) Die Subkonzeptrelation wird hier über dem kartesischen Produkt der Menge S_{OS} aller Konzepte definiert. Dies entspricht der üblichen Vorgehensweise für Ontologien (vgl. MAEDCHE ET AL. (2003) S. 287, BENCH-CAPON/MALCOLM (1999) S. 252). Die Definition von Subsorten wird in ZELEWSKI (1995) Bd. 4. S. 8 f. über einstellige Operationssymbole vorgeschlagen. Auf die Stelligkeit von Operationssymbolen wird später eingegangen werden. Der Nachteil einer solchen Subsortenspezifikation liegt allerdings in der Verlagerung der metasprachlichen Ordnungseigenschaft auf eine objektsprachliche Ebene.

3) Die Relation *Äquivalenzrelation* $=_S \subseteq S_{OS} \times S_{OS}$ wird weiter unten erklärt.

zepte nicht in einem unmittelbaren[1)] Verhältnis zueinander stehen. Wenn für ein Konzept $s \in S_{OS}$ gilt, dass es kein Super- bzw. Subkonzept hat, wird s als *maximales* bzw. *minimales* Konzept bezeichnet.

Die Menge aller Super- bzw. Subkonzepte zu einem Konzept lassen sich mittels der Funktionen

$$\text{sup: } S_{OS} \rightarrow \textit{pot}(S_{OS})$$

mit $$\sup(s_1)=\{\{s_i\}\ i=1,...,I,\ I\in\mathcal{N}_+ \mid \forall i: s_1 \leq_S s_i\}$$

bzw. $$\text{sub: } S_{OS} \rightarrow \textit{pot}(S_{OS})$$

$$\text{sub}(s_1)=\{\{s_i\}\ i=1,...,I,\ I\in\mathcal{N}_+ \mid \forall i: s_i \leq_S s_1\}.$$

Der Funktionswert $\sup(s_1)$ zu einem Konzept $s_1 \in S_{OS}$ ist eine Menge von Konzepten. Es sind genau die Konzepte für die s_1 ein Subkonzept ist. Der Funktionswert $\text{sub}(s_1)$ ist genau die Menge von Konzepten, für die s_1 ein Superkonzept ist.

Die Subkonzeptrelation ist auf dem kartesischen Produkt aller Konzepte definiert. Somit ist sie sowohl über der Menge S_{EW} der einwertigen als auch über der Menge S_{MW} der mengenwertigen Konzepte definiert. Eine Ordnung auf der Menge der Datenkonzepte ist in der Literatur des Öfteren vorzufinden.[2)] Auf der Menge S_D der Datenkonzepte wird die partielle Ordnung von S_{OS} beibehalten. Dabei gilt die partielle Ordnung entsprechend Abbildung 1.

---

1) Die Relation $\leq_S$ entspricht der Vereinigung der Relation $\leq_{dS} \subseteq S \times S$ mit ihrem transitiven und reflexiven Abschluss, wobei $\leq_{dS}$ bei einer unmittelbaren („direkten“) Ordnung zwischen zwei Konzepten definiert ist (vgl. HATZILYGEROUDIS/REICHGELT (1997) S. 255, allerdings lediglich mit dem Verweis auf den transitiven Abschluss der unmittelbaren Konzeptordnung).
Der reflexive Abschluss der Relation $\leq_{dS}$ ist gegeben als
$$r(\leq_{dS}) := \leq_{dS} \cup \{(s,s) \mid s \in S\}.$$
Der transitive Abschluss ist gegeben als
$$t(\leq_{dS}) := \cup_{n\in\mathcal{N}}\, u_n(\leq_{dS}),$$
mit
$$u_0(\leq_{dS}) := \leq_{dS} \text{ und}$$
$$u_{n+1}(\leq_{dS}) := \{(s_1,s_2) \in S\times S \mid \exists s \in S: (s_1,s) \in u_n(\leq_{dS}) \wedge (s,s_2) \in \leq_{dS}\} \setminus \cup_{r\leq n}\, u_r(\leq_{dS}) \text{ für alle } n \in \mathcal{N}.$$
Es gilt daher $\leq_S := \leq_{dS} \cup\, r(\leq_{dS}) \cup\, t(\leq_{dS})$ (Vgl. EHRIG ET AL. (1999) S. 81 f.).
Die Unterscheidung kommt auch einer implementierungsnahen Spezifikation zu Gute. Transitive Relationen lassen sich in deklarativen Programmiersprachen als *rekursive* Regeln umsetzen. Rekursive Regeln sind dadurch gekennzeichnet, dass sie die Mengen der Formeln erweitern, indem sie auf bereits bestehende Formeln zurückgreifen (vgl. AMBLE (1987) S. 48). Bei einer Umsetzung einer Konzepthierarchie in der Programmiersprache PROLOG ist es notwendig, rekursive Regeln so zu definieren, dass sie „terminieren“. Die Termination einer rekursiven Regel in PROLOG ist dann gewährleistet, wenn die Regel eine „Abbruchbedingung“ beinhaltet. Bei einer Regel der Form
$$\text{sub}(s_1,s_3) :- \text{sub}(s_1,s_2),\ \text{sub}(s_2,s_3)$$
ist dies nicht gegeben. Der PROLOG-Interpreter würde bei der obigen Regel im Suchbaum eine *Endlos-Schleife* durchlaufen. Eine Lösungsmöglichkeit besteht darin, im Regel-Rumpf die unmittelbare Konzeptordnungsrelation „direct_sub“ aufzunehmen:
$$\text{sub}(s_1,s_3) :- \text{direct\_sub}(s_1,s_2), \text{sub}(s_2,s_3).$$

2) Vgl. z.B. MÜLLER (1999) S. 13.

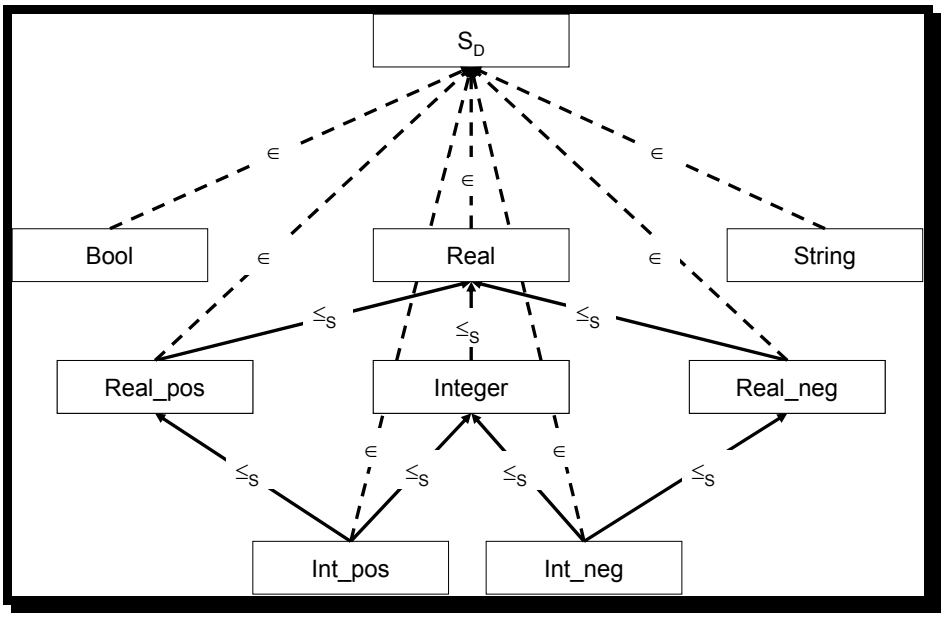

Abbildung 1: Menge der Datenkonzepte S_D

So wird z.B. das Konzept Integer, das extensional durch die Menge $OB_{Integer}$ der ganzen Zahlen interpretiert wird, dem Konzept Real untergeordnet, dass wiederum durch die Menge OB_{Real} der reellen Zahlen interpretiert wird. Eine Ordnungsrelation auf der Menge der mengenwertigen Konzepte wird hingegen seltener angesprochen.[1)]

Während die eine Teilmenge der Relation $\leq_S$ einer einfachen *Taxonomie* entspricht, in der die Konzeptpaare in eine Ordnungsrelation überführt werden, ist die dazu komplementäre Teilmenge komplexer. Für zwei mengenwertige Konzepte $s_3,s_4 \in S_{MS}$ mit $s_3=men(s_1)$, $s_4=men(s_2)$, $s_1,s_2 \in S_{EW}$ gilt nämlich, dass sie in einer Subkonzeptrelation stehen, wenn die einwertigen Konzepte s_1 und s_2, aus denen sie abgeleitet werden, in einer Subkonzeptrelation stehen:

$$\forall s_1,s_2 \in S_{OS}: s_1 \leq_S s_2 \rightarrow men(s_1) \leq_S men(s_2)$$

Wenn beispielsweise das Konzept mensch$\in S_{OS}$ das Subkonzept mann$\in S_{OS}$ hat (mann $\leq_S$ mensch), dann ist das mengenwertige Konzept men(mann) ein Subkonzept des mengenwertigen Konzeptes men(mensch). Diese rein syntaktische Festlegung spiegelt sich später in den Teilmengendeklarationen der konzeptspezifischen Objektmengen wider.

Wenn die Subkonzeptrelation $\leq_S$ zwischen den Konzepten einer Ontologie-Signatur S_{OS} in einer zugehörigen Ontologie-Algebra als Teilmengenrelation $\subseteq$ zwischen konzeptspezifischen Objektmengen interpertiert wird, bereitet dies keine Probleme. Jedes (immer mengenwertige)

1) Vgl. MÜLLER (1999) S. 49.

Subkonzept eines mengenwertigen Konzeptes bezeichnet dann eine Teilmenge der Objektmenge die das Superkonzept extensional interpretiert. Nun handelt es sich in der Algebra nur nicht mehr um Mengen formaler Objekte, sondern um Familien von Mengen formaler Objekte, auf die sich die Teilmengenrelation anwenden lässt. Im obigen Beispiel ist die Objektmenge $OB_{men(neg\_nat)}$, die das mengenwertige Konzept men(neg_nat) extensional interpretiert, eine Teilmenge der Objektmenge $OB_{men(nat)}$, die das mengenwertige Konzept men(nat) extensional interpretiert.[1)]

$=_S \subseteq S_{OS} \times S_{OS}$ und $||_S \subseteq S_{OS} \times S_{OS}$ sind weitere binäre Relationen, die über dem kartesischen Produkt der Konzeptmenge S_{OS} mit sich selbst definiert sind. Mit $=_S$ wird die *Äquivalenzrelation* ausgedrückt.[2)] Zwei Konzepte $s_1,s_2 \in S_{OS}$ stehen in Relation $=_S$ zueinander, wenn sie äquivalent sind. Es handelt sich dabei um Konzepte mit immer gleicher Extension. Sie ist als reflexive, symmetrische und transitive Relation definiert:

Reflexivität $\forall s \in S_{OS}: s =_S s$

Symmetrie $\forall s_1,s_2 \in S_{OS}: s_1 =_S s_2 \leftrightarrow s_2 =_S s_1$

Transitivität $\forall s_1,s_2,s_3 \in S_{OS}: s_1 =_S s_2 \wedge s_2 =_S s_3 \rightarrow s_1 =_S s_3.$

Äquivalente Konzepte sind zudem in der Subkonzeptrelation identisch definiert. Die Identität der Subkonzeptrelationen für äquivalente Konzepte ist zweigeteilt. Im ersten Aspekt wird die Gleichheit der Subkonzepte von äquivalenten Konzepten ausgedrückt:

$$\forall s_1,s_2 \in S_{OS}: s_1 =_S s_2 \rightarrow (\forall s_3 \in S_{OS}: s_3 \leq_S s_2 \leftrightarrow s_3 \leq_S s_1)$$

und

$$\forall s_1,s_2 \in S_{OS}: s_1 =_S s_2 \rightarrow sub(s_1)=sub(s_2).$$

Der zweite Aspekt erstreckt sich auf die Gleichheit der Superkonzepte von äquivalenten Konzepten:

$$\forall s_1,s_2 \in S_{OS}: s_1 =_S s_2 \rightarrow (\forall s_3 \in S_{OS}: s_2 \leq_S s_3 \leftrightarrow s_1 \leq_S s_3)$$

und

$$\forall s_1,s_2 \in S_{OS}: s_1 =_S s_2 \rightarrow sup(s_1)=sup(s_2).$$

Mit $||_S$ wird die *Exklusivitätsrelation* ausgedrückt.[3)]

---

1) Ein ausführliches Beispiel für mengenwertige Konzepte und Objektmengen zu mengenwertigen Konzepten folgt im nächsten Abschnitt.

2) Vgl. KANEIWA (2001) S. 76

3) Vgl. KANEIWA (2001) S. 76.

Zwei Konzepte s_1 und s_2 stehen in Exklusivitätsrelation zueinander, wenn die Objektmengen $OB_{s.1}$ und $OB_{s.2}$, die die Konzepte jeweils extensional interpretieren, disjunkt zueinander sein müssen. Aus der Exklusivitätsrelation ergeben sich keine besonderen Anforderungen an die Ontologie-Signatur. Die Auswirkungen der Spezifikation einer Exklusivitätsrelation zwischen zwei Konzepten werden als Anforderung bei der Definition von Ontologie-Algebren im nächsten Abschnitt formuliert.

Die Subkonzept-, Äquivalenz- und Exklusivitätsrelationen sind lediglich Beispiele dafür, wie semantisch ausdrucksmächtige Modellierungsprimitive in Ontologien eingebunden werden können. Sie werden vielfach von Sprachen zur Konstruktion von Ontologien gar nicht zur Verfügung gestellt. Es ließen sich mühelos noch andere Modellierungsprimitive[1)] in den hier vorgestellten Ansatz einführen, die allerdings für die hier angegangene Problemstellung irrelevant sind.

2.2.1.2.3 Bezeichnungs- und Definitionsfunktionen bez_{ONT} und def_{ONT}

Die Familie der Bezeichnungsfunktionen

$$bez=\{bez_{ger},bez_{eng},bez_{fr},...\}$$

hat als Mitglieder Funktionen, mit denen Konzepten $s \in S_{OS}$ ein Menge natürlichsprachlicher *Bezeichner* zugeordnet werden. Der Index $lan \in \{ger,eng,fr,...\}$ gibt an, welcher natürlichen Sprache die Bezeichner entstammen. Es wurden hier lediglich Deutsch, Englisch und Französisch exemplarisch aufgezählt. Die Mengenfamilie lässt sich problemlos ausweiten, um auch andere Sprachen berücksichtigen zu können.

Der Zielbereich der Bezeichnungsfunktion ist immer[2)] die Potenzmenge der Menge ALPH*. Jedes Element von ALPH* ist eine Konkatenation von Buchstaben über dem Alphabet

$$ALPH=\{0,1,...,9,a,b,...,z,A,B,...,Z,,,;,.,:,+,-,?,!,\$,\%,/,(,),=, ,\}.$$

Jedes Element $w \in ALPH^*$ ist ein Wort über dem Alphabet ALPH. ALPH* kann z.B. wie folgt aussehen:[3)] ALPH*={`Person`,`Mensch`,`Auto`,`SL 500`,...}.

---

1) Beispiele für weitere Modellierungsprimitive sind die *Aggregation*, *Ausschöpfung* und *erweiterte Kardinalitäten* für Attributssymbole.

2) Grundsätzlich ist eine Unterscheidung der Zielbereiche der sprachabhängigen Bezeichnungsfunktion denkbar. Sie würde dann notwendig werden, wenn sprachspezifische Buchstaben wie z.B. „é“ und „à“ berücksichtigt werden müssten. Der Einfachheit halber wurde das hier ausgelassen. Dies entspricht der implementierungsnahesten Vorgehensweise, bei der der Unicode-Zeichensatz vorausgesetzt wird.

3) Um die Elemente der Menge ALPH* von den Elementen der Menge S_{OS} unterscheiden zu können, werden die erstgenannten im Folgenden in der Schriftart `Courier` angegeben.

Die Potenzmenge *pot*(ALPH*) sieht wie folgt aus:

pot(ALPH*)={{`Person`},{`Mensch`},{`Person`,`Mensch`},{`Person`,`Auto`}...}. Das Konzept Person$\in S_{OS}$ könnte z.B. die Bezeichnermenge

bez_{ger}(Person)={`Person`,`Mensch`} haben.

Die Familie der Definitionsfunktionen

$$def=\{def_{ger},def_{eng},def_{fr},...\}$$

hat als Mitglieder Funktionen, mit denen Konzepten $s\in S_{OS}$ eine *Definition* in einer natürlichen Sprache zugeordnet werden kann. Die Bilder der Definitionsfunktionen def_{lan} sind Wörter[1)] über dem Alphabet ALPH. Im Unterschied zu den Bezeichnungsfunktion bez_{lan} bilden die Definitionsfunktionen def_{lan} Konzepte nicht auf eine Teilmenge von ALPH* ab. Jedem Konzept wird in jeder Sprache höchstens *eine* Definition zugeordnet.

Durch die Zuordnung einer Definition zu jedem Konzept, erhalten letztgenannte eine *intensionale Semantik*. Die Zuordnung einer intensionalen Semantik ist der *Sinn* eines Konzeptes, während die Zuordnung einer Menge formaler Objekte in einer Algebra seine *Bedeutung* ist. Die Bedeutung eines Konzeptes ist somit seine extensionale Semantik. Der Zusammenhang wird des Öfteren[2)] mit dem *Bedeutungsdreieck* illustriert. In Abbildung 2 ist ein *erweitertes Bedeutungsdreieck* wiedergegeben.

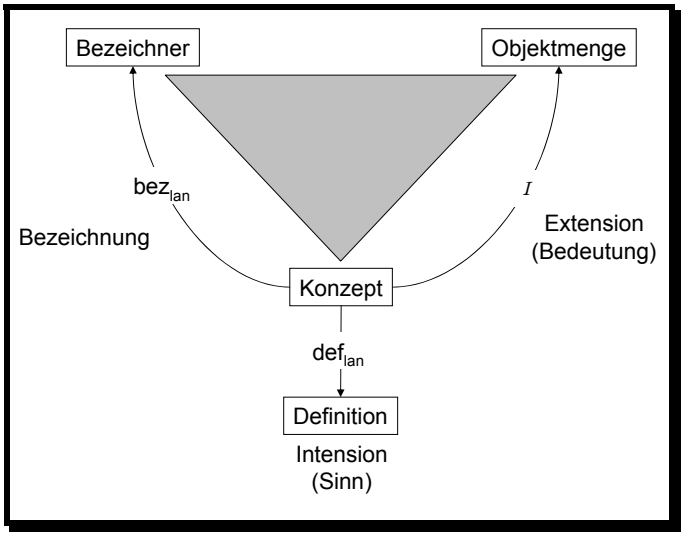

Abbildung 2: Erweitertes Bedeutungsdreieck

1) Bei den Bildern der Funktionen def_{lan} handelt es sich im intuitiven Verständnis um *Sätze*. Formal unterscheiden sich allerdings Sätze über einem Alphabet nicht von Wörtern über einem Alphabet.

2) Vgl. z.B. SURE (2003) S. 27.

Die Erweiterung des Bedeutungsdreieckes ist notwendig, um die beiden Aspekte der Semantik eines Konzeptes erfassen zu können. Der erste Aspekt erstreckt sich auf die extensionale Semantik eines Konzeptes. Der zweite Aspekt erstreckt sich auf die intensionale Semantik. Im traditionellen Bedeutungsdreieck wird der letztgenannte Aspekt nicht berücksichtigt. Für die vorliegende Arbeit wurde er dennoch aus einem Grund aufgenommen: Die extensionale Semantik berücksichtigt bei der Interpretation von Konzepten lediglich den *aktuellen* Zustand der Ontologie-Algebra. Bei einem Zustandsübergang wird ein Konzept eventuell anhand einer alternativen Extension interpretiert. Den Extremfall stellt der Zustand dar, in dem zu einem Konzept keine Extension gegeben ist. Das kann beispielsweise dann eintreten, wenn durch „Abräumkanten" in Ontologie-Netzen sämtliche Extensionen zu einem Konzept gelöscht werden. Ein solches Konzept könnte nicht extensional interpretiert werden, da es keine Extension im aktuellen Zustand hätte. Um in einem solchen Zustand dennoch eine – zumindest für Menschen – verarbeitbare Semantik zu haben, kann auf die Intension des Konzeptes zurückgegriffen werden.

Darüber hinaus lassen sich anhand der Bezeichnungs- und Definitionsfunktion *Synonyme*, *Homonyme* und *Äquipollenzen* voneinander unterscheiden.[1)]

Bezeichnerbeziehung	**Konzepte**	**Bezeichnung**	**Intension**	**Extension**
Homonym	$s_1 \neq s_2$	$bez_{lan}(s_1)=bez_{lan}(s_2)$	$def_{lan}(s_1)\neq def_{lan}(s_2)$	$OB_{s.1} \neq OB_{s.2}$
Synonyme	$s_1 = s_2$	$bez_{lan}(s_1)\neq bez_{lan}(s_2)$	$def(s_1)_{lan}=def_{lan}(s_2)$	$OB_{s.1}=OB_{s.2}$
Äquipollenzen	$s_1 \neq s_2$	$bez_{lan}(s_1)\neq bez_{lan}(s_2)$	$def_{lan}(s_1)\neq def_{lan}(s_2)$	$OB_{s.1}=OB_{s.2}$

Tabelle 1: Homonyme, Synonyme und Äquipollenzen

Homonyme sind solche Bezeichner, die für unterschiedliche Konzepte gelten, die unterschiedliche Intensionen und unterschiedliche Extensionen haben. Z.B. sowohl das Konzept Handlungsfähigkeit als auch das Konzept Weisungsbefugnis mit dem Bezeichner `Kompetenz` bezeichnet sein. Ein Bezeichner b ist Element der Menge

$$hom \subseteq ALPH^*$$

1) Vgl. ORTNER (1997) S. 31 f.

wenn b ein Homonym ist. Es gilt für eine Sprache lan∈{ger,eng,fr,...}:

$$\forall b \in ALPH^*: hom(b) \leftrightarrow$$

$$\exists s_1,s_2 \in S_{OS}: s_1 \neq s_2 \wedge bez_{lan}(s_1)=bez_{lan}(s_2)=b \wedge def_{lan}(s_1) \neq def_{lan}(s_2) \wedge OB_{s.1} \neq OB_{s.2}$$

Synonyme sind solche voneinander unterschiedlichen Bezeichner, die für ein Konzept gelten. Z.B. sind die Bezeichner `C++` und `C Plus Plus` Synonyme, wenn beide Bezeichner *eines* Konzeptes sind und sowohl intensional als auch extensional jeweils gleich interpretiert werden. Das Bezeichnerpaar (b_1,b_2) ist Element der Synonymrelation

$$syn \subseteq ALPH^* \times ALPH^*,$$

wenn b_1 und b_2 Synonyme sind. Es gilt für eine Sprache lan∈{ger,eng,fr,...}:

$$\forall b_1,b_2 \in ALPH^*: syn(b_1,b_2) \leftrightarrow$$

$$\exists s \in S_{OS}: b_1 \in bez_{lan}(s) \wedge b_2 \in bez_{lan}(s) \wedge b_1 \neq b_2 \wedge OB_{s.1}=OB_{s.2}.$$

Äquipollenzen Bezeichner von unterschiedlichen Konzepten mit zwar unterschiedlichen Intensionen aber gleichen Extensionen. Beispielweise werden die beiden Konzepte *Warenkonto* und *Lagerbestand* mit den Bezeichnern `Warenkonto` bzw. `Lagerbestand` zwar unterschiedlich definiert und bezeichnet haben aber immer die gleiche Extension. Das Bezeichnerpaar (b_1,b_2) ist Element der Äquipollenzrelation

$$äquip \subseteq ALPH^* \times ALPH^*,$$

wenn b_1 und b_2 Äquipollenzen sind. Es gilt für eine Sprache lan∈{ger,eng,fr,...}:

$$\forall b_1,b_2 \in ALPH^*: äquip(b_1,b_2) \leftrightarrow$$

$$\exists s_1,s_2 \in S_{OS}: s_1 \neq s_2 \wedge bez_{lan}(s_1) \neq bez_{lan}(s_2) \wedge def_{lan}(s_1) \neq def_{lan}(s_2) \wedge OB_{s.1}=OB_{s.1}.$$

Bereits bei den aufgeführten Beispielen wird ein grundsätzliches Dilemma offenbart, mit dem sich bereits Teil-)Disziplinen der Wissenschaftstheorie, der sprachanalytischen Philosophie, der Informatik und der Künstlichen-Intelligenz-Forschung auseinandergesetzt haben, und dem sich auch der hier vorgestellte Ansatz nicht problemlos entziehen kann: In der vorliegenden Arbeit werden Konzepten Bezeichner zugeordnet, um syntaktische und semantische Differenzen und Gemeinsamkeiten zwischen Konzepten besser beleuchten zu können. Wenn Konzepte als „sprachunabhängige Entitäten“[1)] verstanden werden, denen sprachliche Entitäten (Bezeichner) zugeordnet werden, bleibt jedoch die Frage offen, wie auf Konzepte zugegriffen werden kann, ohne sie sprachlich ausdrücken zu können. Schließlich werden Konzepte – wie

1) HARRAS ET AL. (1996) S. 10.

die o.a. Konzepte Handlungsfähigkeit, Warenkonto usw. – bereits im Text mittels der Bezeichner `Handlungsfähigkeit`, `Warenkonto` usw. angesprochen.

Ein Ausweg aus dem Dilemma besteht darin, Konzepte nicht als sprachunabhängig anzunehmen. Diese Basisposition deckt sich mit Annahmen des *gemäßigten Konstruktivismus*.[1)] Die Thematik wird hier allerdings nicht weiter vertieft, da sie den Rahmen der vorliegenden Arbeit sprengen würde. Aus pragmatischer Perspektive wird vorgeschlagen, zu jedem Konzept $s \in S_{OS}$ mindestens den Bezeichner $\mathtt{s} \in bez_{lan}(s)$ anzunehmen.

2.2.1.2.4 Attributssymbole OPS_{OS}

2.2.1.2.4.1 Überblick zu Attributssymbolen OPS_{OS}

Bei den Attributssymbolen einer Ontologie-Signatur handelt es sich um Bezeichner für Operationen aus einer Ontologie-Algebra A_{OS}. Die Operationen sind als *Attribute* formaler Objekte zu interpretieren. Die Bilder der Operationen sind *Attributswerte*. Entsprechend werden im Folgenden die Ontologie-Operationssymbole aus einer Ontologie-Signatur auch als *Attributssymbole* angesprochen.[2)]

Sämtliche Attributssymbole $O_1,...,O_n \in OPS_{OS}$ werden mit der Attributssymboltypisierungsfunktion

$$typ_{OPS\text{-}ONT}: OPS_{OS} \rightarrow S_{OS} \times S_{OS}$$

typisiert. Ähnlich zu dem Vorgehen bei Operationssymbolen in algebraischen Signaturen, kann mittels der Funktionen

$$ARG_{OPS\text{-}ONT}: OPS_{OS} \rightarrow S_{OS}$$

bzw.

$$ZIEL_{OPS\text{-}ONT}: OPS_{OS} \rightarrow S_{OS}$$

auf den Argument- bzw. den Zielbereich eines Attributssymbols zugegriffen werden. Für ein Attributssymbol $O_i \in OPS_{OS}$ mit $typ_{OPS\text{-}ONT}(O_i)=s_1,s_2$ sind $ARG_{OPS\text{-}ONT}(O_i)=s_1$ und $ZIEL_{OPS}(O_i)=s_2$.

Jedem Attributssymbol $O_i \in OPS_{OS}$ wird durch $typ_{OPS\text{-}ONT}$ eine Stelligkeit zugewiesen. Das Bild (s_1,s_2) der Funktion $typ_{OPS\text{-}ONT}$ entspricht an der Stelle O_i zu einem Attributssymbol O_i dessen *Argument-* (s_1) bzw. *Zielkonzept* (s_2). Attributssymbole, deren Zielkonzept $s_2 \in S_{OS}$ ein

---

1) Vgl. SCHÜTTE (1999) S. 227 ff.

2) Zur funktionalen Erfassung von Attributen vgl. RAUH/STICKEL (1997) S. 47.

mengenwertiges Konzept s_2=men(s_1) ist, die von einer Sorte $s_1 \in S_{EW}$ abgeleitet wurde, werden als *mengenwertige* und sonstige Attributssymbole als *einwertige* Attributssymbole bezeichnet.[1)]

Teilweise wird in der Literatur eine Einschränkung des Vorbereiches von Attributssymbolen auf die Menge S_{EW} vorgenommen.[2)] Die Einschränkung erfolgt unter der Annahme, dass die Elementen der Objektmengen $OB_{s.1}$...$OB_{s.n}$, die Konzepte $s_1,...,s_n \in S^{impl}$ aus der Konzeptmenge S_{OS} extensional interpretieren, keine Attributwerte zugeordnet werden sollen. Es handelt sich dabei um die Mengen OB_s mit $s \in S_D$ und die Mengen OB_s mit $s \in S_{MW}$. Eine solche Einschränkung findet sich auch in den Metamodellen von Sprachen wieder, die für die Konstruktion von Ontologien definiert wurden. Im Metamodell[3)] zu RDF sind zum Beispiel nur Aussagen (*Statements*) zugelassen, deren Vorbereich der Menge der *Resources*, deren Nachbereich aber der Vereinigungsmenge von Resources und *Literals* entspricht[4)]. Die Resources entsprechen der Konzeptmenge $S_{EW} \backslash S_D$, während die Literals der Menge S_D der Datenkonzepte entsprechen. Eine Unterscheidung zwischen ein- und mengenwertigen Attributssymbolen ist in RDF nicht vorgesehen.

Problematisch wird dies allerdings, wenn ein Ontologie-gestütztes Modell beispielsweise für die Menge der natürlichen Zahlen oder die Menge der Zeichenketten konstruiert werden soll. In einem solchen Modell könnte es von Interesse sein, Attribute für eine bestimmte Zahl oder eine bestimmte Zeichenkette zu definieren. Solche Konstruktionen sind zwar für das hiesige Konzept weniger von Relevanz, da es sich hier um einen Ansatz handelt, der primär die Modellierung *betrieblicher* Phänomene verfolgt. In betrieblichen Szenarien sind nämlich Eigenschaften von konkreten Zahlen oder Zeichenketten in der Regel nicht von Bedeutung. Jedoch sind auch Szenarien vorstellbar, in denen sie von Interesse sein können. Beispielsweise kann es sein, dass für Produkte Seriennummern als zeichenkettenartige Attribute definierbar sein müssen, bei denen die Zusammensetzung der Zeichenkette einem bestimmten Muster zu folgen hat. Darüber hinaus kann es auch sein, dass Attribute für Mengen formaler Objekte vergebbar sein sollten. Ein Beispiel hierfür sind Attribute von Aufträgen, die sich aus einer Menge von Produkten zusammensetzen. Wenn eine bestimmte Produktkonstellation gesondert

---

1) Vgl. PATIG (2001) S. 65.

2) Vgl. BENCH-CAPON/MALCOLM (1999) S. 252; BENCH-CAPON ET AL. (2003) S. 704.

3) Als Metamodell einer Sprache wird ein Modell der Modellierungsprimitive und ihrer Beziehungen bezeichnet, die eine Modellierungssprache zur Verfügung stellt (vgl. RAUH/STICKEL (1997) S. 101 ff.

4) Vgl. KLAPSING (2003) S. 41

ausgezeichnet werden muss, kann dies nur über Attribute erfolgen. Um keine „unnötigen“ Beschränkungen der Ausdruckmächtigkeit des hier vorgestellten Ansatzes zu verantworten, werden im Vorbereich von Attributssymbolen alle Konzepte aus der Menge S_{OS} zugelassen.

Die Mächtigkeit des Vorbereichs eines Operationssymbols für algebraische, relationale und ontologische Signaturen ist durch die Stelligkeit des Operationssymbols gegeben. Für ontologische Signaturen ist im Gegensatz zu algebraischen und relationalen Signaturen nur jeweils ein Konzept im Vorbereich von Attributssymbolen definiert. Für jedes Attributssymbol $O \in OPS_{OS}$ gilt, dass das Bild s der Funktion $ARG_{OPS\text{-}ONT}(O)$ an der Stelle O ein Konzept aus der Menge S_{OS} ist. Durch die Modifikation der Deklarationsmöglichkeit für den Vorbereich eines Operationssymbols ergibt sich zwei Besonderheiten gegenüber Ontologie-Signaturen zwei Unterschiede.

Es werden im traditionellen Signatur-Ansatz sowohl Operationssymbole $o_1 \in OPS$ mit $ARG_{OPS}(O_1)=\lambda$ als auch Operationssymbole $O_2 \in OPS$ mit $ARG_{OPS}=w=s_1...s_n$ mit $n>1$ zugelassen. Beide Fälle führen zu einem Unterschied aufgrund der Stelligkeit der Operationssymbole. Im ersten Fall handelt es sich um Operationssymbole mit einem leeren Vorbereich in ihrer Typisierung. Sie wurden bereits als *Konstantensymbole* eingeführt. Das Wort im Vorbereich des Operationssymbols, das ein Konstantensymbolen ist, weist dann die Stelligkeit n=0 auf. Attributssymbole in Ontologie-Signaturen haben allerdings stets *genau ein* Konzept $s \in S_{OS}$ im Vorbereich. Im zweiten Fall kann der Vorbereich eines Operationssymbols durch eine Verkettung von Sorten gegeben sein. Das Wort im Vorbereich eines Operationssymbols mit einer Verkettung von Sorten weist dann die Stelligkeit $n>1$ auf. In Ontologie-Signaturen sind allerdings nur Attributssymbole zugelassen, die eine Stelligkeit n=1 haben.

Die Menge OPS_{OS} der Attributssymbole ist als eine Vereinigung der Form

$$OPS_{OS}=OPS_{ST} \cup OPS_{SYM} \cup OPS_{TR}$$

definiert. Die Menge OPS_{ST} ist definiert als die Menge der *Standard-Attributssymbole*. Es handelt sich hierbei um Attributssymbole, denen in der Ontologie-Signatur keine näher spezifizierten Eigenschaften zukommen. Die Menge OPS_{SYM} der symmetrischen Attributssymbole und die Menge OPS_{TR} der transitiven Attributssymbole werden in den folgenden Abschnitten behandelt.

2.2.1.2.4.2 Symmetrische Attributssymbole OPS_{SYM}

Die Menge OPS_{SYM} umfasst Attributssymbole, die im „Normalfall“ durch solche Operationen aus einer Ontologie-Algebra interpretiert werden, deren Umkehroperationen[1)] sie selbst sind. Daher werden sie im Folgenden als *symmetrische Attributssymbole* bezeichnet. Aufgrund der Einführung mengenwertiger Konzepte ist eine Unterscheidung in drei Fälle notwendig. Im ersten Fall sind das Argument- und das Zielkonzept eines symmetrischen Attributssymbols identisch. Es kann sich dabei sowohl um eine einwertiges Konzept als auch um ein megenwertiges Konzept handeln. Dieser Fall bezeichnet den „Normalfall“ einer symmetrischen Operation aus der Ontologie-Algebra. Im zweiten Fall ist das Zielkonzept eines symmetrischen Attributssymbols das mengenwertige Konzept zu dem Argumentkonzept des Attributssymbols. Im dritten Fall ist das Argumentkonzept eines symmetrischen Attributssymbols das mengenwertige Konzept zu dem Zielkonzept.

Damit gilt für alle symmetrischen Attributssymbole:

$$\forall O \in OPS_{SYM}: (typ_{OPS\text{-}ONT}(O)=(s_1,s_2)) \rightarrow ((s_1=s_2) \veebar (s_2=men(s_1)) \veebar (s_1=men(s_2))).$$

Die erste Teilformel der disjunktiv verknüpften Konklusionsformel beinhaltet den ersten Fall, bei dem Ziel- und Argumentkonzept eines Attributssymbols übereinstimmen. Die zweite Teilformel beinhaltet den zweiten Fall, bei dem das Zielkonzept das mengenwertige Konzept zu dem Argumentkonzept ist.

Um die zwei Varianten symmetrischer Attributssymbole formal kompakter handhaben zu können, kann die Menge OPS_{SYM} zu den disjunkten Mengen $OPS_{SYM.EW}$ und $OPS_{SYM.MW}$ unterteilt werden:

$$OPS_{SYM}=OPS_{SYM.EW} \cup OPS_{SYM.MW}.$$

Die zwei Fälle für symmetrische Attributssymbole werden wie folgt definiert:

(1.) 1. Fall: $OPS_{SYM.EW}$:

Die Argument- und Zielkonzepte des Attributssymbols sind gleich:

$$\forall O \in OPS_{SYM.EW}: typ_{OPS\text{-}ONT}(O)=(s_1,s_2) \rightarrow s_1=s_2.$$

(2.) 2. Fall: $OPS^{sym.mw.1}$:

Das Zielkonzept des Attributssymbols ist das mengenwertige Konzept zum Argumentkonzept des Attributssymbols:

$$\forall O \in OPS_{SYM.MW}: typ_{OPS\text{-}ONT}(O)=(s_1,s_2) \rightarrow s_1 \in S_{EW} \wedge s_2=men(s_1)).$$

---

1) Die Umkehroperation $o^{-1}:OB_2 \rightarrow OB_1$ zu einer Operation $o:OB_1 \rightarrow OB_2$ ist die Operation, für die gilt:
$\forall ob_1 \in OB_1, ob_2 \in OB_2: o^{-1}(ob_2)=ob_1 \leftrightarrow o(ob_1)=ob_2$.

In der Abbildung 3 sind die zwei Fälle symmetrischer Attributssymbole dargestellt.

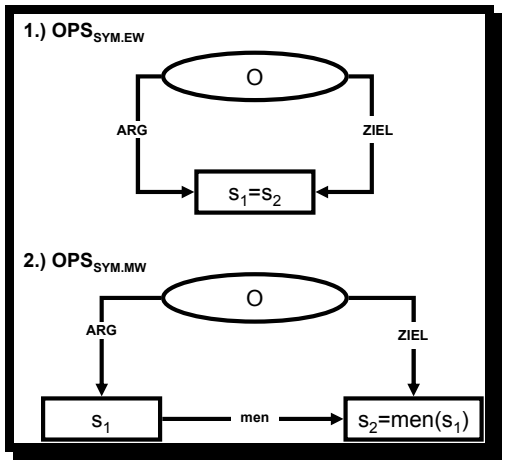

Abbildung 3: Zwei Fälle für Attributssymbole aus der Menge OPS^{sym}

Aufgrund der vorherigen Vereinbarung $OPS_{SYM}=OPS_{SYM.EW} \cup OPS_{SYM.MW}$ gilt stets:

$$\forall i \in I: O_i \in OPS_{SYM.EW} \veebar O_i \in OPS_{SYM.MW} \rightarrow OPS_{SYM}.$$

Somit sind die disjunkten Mengen $OPS_{SYM.EW}$ und $OPS_{SYM.MW}$ hinsichtlich OPS_{SYM} exhaustiv.

2.2.1.2.4.3 Transitive Attributssymbole OPS_{TR}

Die Menge OPS_{TR} umfasst Attributssymbole, die durch Attribute aus einer Ontologie-Algebra A_{OS} interpretiert werden, welche sich als transitive Operation charakterisieren lassen. Daher werden sie im Folgenden auch als „transitive Attributssymbole“ angesprochen. Für Attributssymbole $O_i \in OPS_{TR}$ ist der Fall mit einem einwertigen Zielkonzept $ZIEL_{OPS\text{-}ONT}(O)=s$ mit $s \in S_{EW}$ nicht definiert. Das Zielkonzept zu einem transitiven Attributssymbol ist immer das mengenwertige Konzept zu dessen Argumentkonzept. Es muss sich bei dem Zielkonzept um eine mengenwertiges Konzept handeln, da der Transitivität eine Zuordnung von mehreren Objekten als Attributswerte zu einem Objekt inhärent behaftet ist. Es gilt demnach für alle transitiven Attributssymbole:

$$\forall O \in OPS_{TR}: typ_{OPS\text{-}ONT}(O)=(s_1,s_2) \rightarrow s_1 \in S_{EW} \wedge s_2=men(s_1).$$

In der Abbildung 4 ist die Typisierung transitiver Attributssymbole dargestellt.

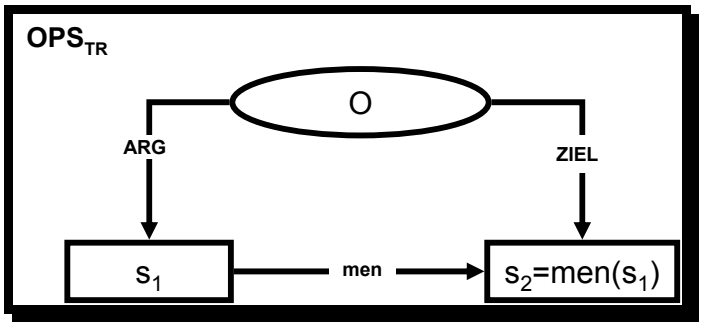

Abbildung 4: Transitive Attributssymbole OPS^{trans}

2.2.1.2.4.4 Inverserelation inv_{OPS}

Die Inverserelation inv_{ONT} ist auf dem kartesischen Produkt der Attributssymbolmenge OPS_{OS} mit sich selbst definiert. Für zwei Attributssymbole $O_1,O_2 \in OPS_{OS}$ gilt für den „Normalfall“ $(O_1,O_2) \in inv_{ONT}$, dass wenn das Attributssymbol O_1 durch ein Attribut o_1 interpretiert wird, die Umkehroperation $o_2=o_1^{-1}$ das Attributssymbol O_2 interpretiert. Das bedeutet, dass wenn eine Formel $G_1=(O_1(t_1)=t_2)$ gültig in einer Ontologie-Algebra A_{OS} ist, auch die Formel $G_2=(O_2(t_2)=t_1)$ gültig in der Algebra A_{OS} sein muss. Darüber hinaus ist aber noch eine Erweiterung des Prinzips inverser Attributssymbole notwendig, um Mengensorten berücksichtigen zu können.

In der Anforderung:

$$\forall(O_1,O_2) \in inv_{OPS}: ((typ_{OPS\text{-}ONT}(O_1)=(s_1,s_2)) \wedge (typ_{OPS\text{-}ONT}(s_3,s_4))) \rightarrow$$

$$(((s_1=s_4) \wedge (s_2=s_3)) \veebar ((s_4=men(s_1)) \wedge (s_2=s_3)) \veebar ((s_4=men(s_1)) \wedge (s_2=men(s_3))).$$

sind drei exklusive Fälle für Attributssymbole berücksichtigt, die in Relation inv_{OPS} zueinander stehen können. Die erste Teilformel der disjunktiv verknüpften Konklusionsformel beinhaltet den „normalen“ Fall von Attributssymbolen, die zueinander in Relation inv_{OPS} zueinander stehen. Die zweite und dritte Teilformel sind notwendig, um Mengenkonzepte berücksichtigen zu können.

Die Inverserelation inv_{OPS} zwischen zwei Attributssymbolen kann nur in folgenden drei exklusiven Fällen bestehen:

(1.) Fall 1: inv_{EW}

Die Argument- und Zielkonzepte der Attributssymbole sind jeweils „vertauscht“:

$$\forall(O_1,O_2) \in inv_{EW}: ((typ_{OPS\text{-}ONT}(O_1)=(s_1,s_2)) \wedge (typ_{OPS\text{-}ONT}(s_3,s_4))) \rightarrow$$

$$((s_1=s_4) \wedge (s_2=s_3)).$$

(2.) Fall 2: $inv_{MW.1}$

Das Zielkonzept des ersten Attributssymbols ist das Argumentkonzept des zweiten Attributssymbols und das Zielkonzept des zweiten Attributssymbols ist das mengenwertige Konzept zum Argumentkonzept des ersten Attributssymbols:

$$\forall(O_1,O_2)\in inv_{MW.2}: (typ_{OPS\text{-}ONT}(O_1)=(s_1,s_2) \wedge typ_{OPS\text{-}ONT}(s_3,s_4)) \rightarrow$$
$$((s_2=s_3) \wedge s_4\in S_{EW} \wedge (s_4=men(s_1)).$$

(3.) Fall 3: $inv_{MW.2}$

Die Zielsorten der Attributssymbole sind jeweils eine Mengensorte zu der Argumentsorte des anderen Attributssymbols:

$$\forall(O_1,O_2)\in inv_{MW.2}: typ_{OPS\text{-}ONT}(O_1)=(s_1,s_2) \wedge typ_{OPS\text{-}ONT}(s_3,s_4) \rightarrow$$
$$(s_2\in S_{EW} \wedge (s_2=men(s_3) \wedge s_1\in S_{EW} \wedge s_4=men(s_1)).$$

In der Abbildung 5 sind alle drei Fälle dargestellt.

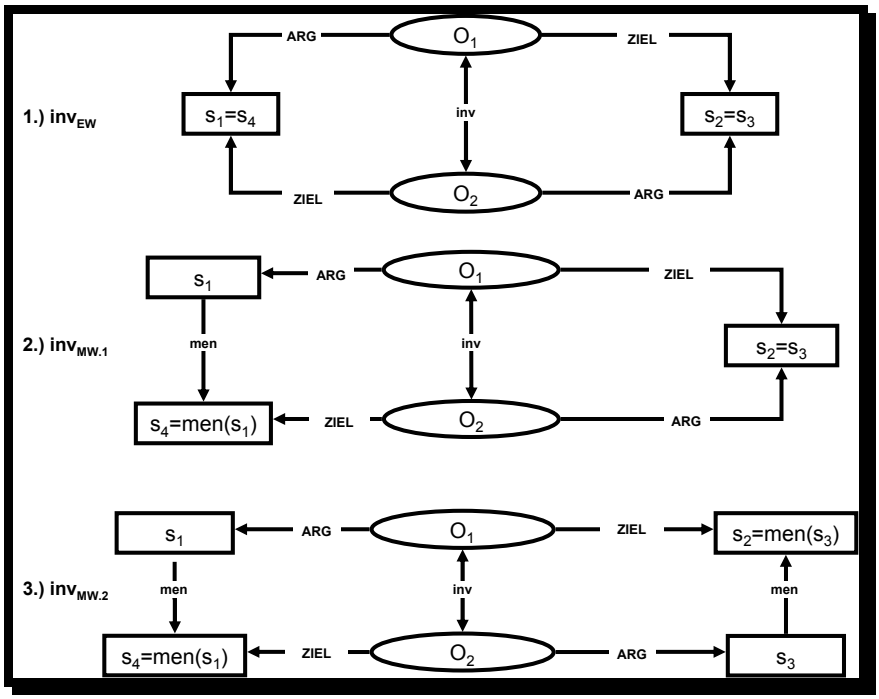

Abbildung 5: Drei Fälle der Inverserelation inv_{OPS}

Die Relationen inv_{EW}, $inv_{MW.1}$ und $inv_{MW.1}$ sind Subrelationen[1)] der Relationen inv_{OPS}. Es gilt nämlich für alle drei Relationen:

$$\forall O_1,O_2 \in OPS_{OS}: (O_1,O_2) \in inv_{EW} \veebar (O_1,O_2) \in inv_{MW.1} \veebar (O_1,O_2) \in inv_{MW.2}$$

$$\leftrightarrow$$

$$(O_1,O_2) \in inv_{OPS}.$$

Es handelt sich dabei um eine Bijugatsformel, da keine Inverselation inv_{OPS} außerhalb der drei Fälle existiert.

Die Inverselation $inv_{OPS}(O_1,O_2)$ zwischen zwei Attributssymbolen deutet im „Normalfall" inv_{EW} darauf hin, dass die erste Komponente O_1 durch die Umkehroperation o_1^{-1} zu der Operation o_2 interpretiert wird, die das Attributsymbol O_2 interpretiert. Dieser Fall trifft auch auf alle symmetrischen Operationen zu, die Attributssymbole aus der Menge OPS_{SYM} interpretieren. Daher gilt, das jedes Attributssymbol $O_i \in OPS_{OS}$, dass als Element der Menge $OPS_{SYM} \subseteq OPS_{OS}$ ausgezeichnet ist, mit sich selbst in Inverserelation inv_{OPS} steht :

$$\forall i \in I: O_i \in OPS_{SYM} \rightarrow inv_{OPS}(O_i,O_i).$$

Die Zuordnung symmetrischer Attributssymbole aus der Menge OPS^{sym} zur Relation inv_{OPS} kann auch entsprechend ihrer Unterteilung zu $OPS_{SYM.EW}$ und $OPS_{SYM.MW}$ erfolgen. Es gelten:

(1.) $\forall i \in I: (O_i \in OPS_{SYM.EW}) \rightarrow (inv_{EW}(O_i,O_i))$ und

(2.) $\forall i \in I: (O_i \in OPS_{SYM.MW}) \rightarrow (inv_{MW.2}(O_i,O_i))$.

Die erste Zuordnung ist dadurch bedingt, dass symmetrische Attributssymbole, bei denen Argument- und Zielkonzept gleich sind, mit sich selbst in Inverserelation stehen. Beispielsweise ist das Attributssymbol „Verheiratet_mit", dass das Argument- und Zilekonzept „Person" hat, die Inverse ihrer selbst. Im zweiten Fall ist das Zielkonzept eines symmetrischen Attributssymbols ein mengenwertiges Konzept zum einwertigen Argumentkonzept. Beispielsweise steht das Attributssymbol „Arbeitet_mit", mit dem Argumentkonzept „Person" und dem Zielkonzept „men(Person)", mit sich selbst in Inverserelation. In der Abbildung 5 gilt in diesem Fall für den Fall 3.):

$s_1=s_3$=Person und $s_2=s_4$=men(s_3)=men(s_1)=men(Person).

2.2.1.2.5 Kardinalitätsfunktionen min und max

1) Eine Relation rel_1 ist Subrelation Relation rel_2, wenn die Regel gilt: wenn $rel_1(x_1,...,x_i)$ mit i=1...I $I \in \mathcal{N}$ gilt, dann gilt auch $rel_2(x_1,...,x_i)$. Das formale Erfassen der Regel würde einen Übergang zur Prädikatenlogik zweiter Ordnung erfodern.

Mit den Funktionen

$$\min:OPS_{OS} \rightarrow \mathcal{N} \text{ und}$$

$$\max:OPS_{OS} \rightarrow \mathcal{N}_+$$

werden allen Attributssymbolen *Kardinalitäten* zugewiesen.[1)]

Die *Minimum-Kardinalität* min(O) zu einem Attributssymbol $O \in OPS_{OS}$ gibt an, wie viele formale Objekte *mindestens* für ein formales Objekt ob als Attributswerte o(ob) bezüglich des Attributs o definiert sein müssen. Ist die Minimum-Kardinalität eines Attributssymbols größer als 1, muss das Zielkonzept des Attributssymbols ein mengenwertiges Konzept sein:

$$\forall i \in I: O:OPSOS \wedge typ_{OPS\text{-}ONT}(O)=(s_1,s_2) \wedge \min(O)>1 \rightarrow s_2 \in S_{MW}.$$

Wenn die Minimum-Kardinalität eines Attributssymbols O gleich 1 ist, handelt es sich bei dem Attribut o, das O interpretiert, um eine *totale* Operation, da jedes formale Objekt ob_1 aus der konzeptspezifischen Menge $OB_{s.1}$ mittels $o(ob_1)$ auf ein formales Objekt ob_2 aus der konzeptspezifischen Menge $OB_{s.1}$ abgebildet werden muss. Es kann sich bei dem Zielkonzept des Attributssymbol O entweder um ein einwertiges Konzept aus der Menge S_{EW} oder eine mengenwertiges Konzept aus der Menge S_{MW} handeln. Wenn die Minimum-Kardinalität des Attributssymbols O gleich 0 ist, handelt es sich bei dem Attribut o, das O interpretiert, um eine *partielle* Operation. Die Operation kann linkstotal und somit eine auch eine totale Operation sein, muss sie aber nicht. Sie ist linkstotal, wenn *jedes* formale Objekte ob_1 aus der konzeptspezifischen Menge $OB_{s.1}$ auf ein formales Objekt ob_2 aus $OB_{s.2}$ mittels $o(ob_2)$ abgebildet ist, obwohl es nicht aufgrund der Minimum-Kardinalität notwendig ist.

Die *Maximum-Kardinalität* max(O) zu einem Attributssymbol $O \in OPS_{OS}$ gibt an, wie viele formale Objekte *höchstens* für ein formales Objekt ob als Attributswerte $o(ob_1)$ bezüglich des Attributs o definiert sein dürfen. Wenn die Maximum-Kardinalität max(O) eines Attributssymbols genau 1 ist, dann darf das Zielkonzept des Attributssymbols kein mengenwertiges

1) Vgl. zu Kardinalitäten in Bezug auf Ontologien MAEDCHE ET AL. (2003) S. 287; MCGUINESS/VAN HARMELEN (2004) Abschnitt 3.5.

Konzept sein:[1)]

$$\forall\ O \in OPS_{OS}: typ_{OPS\text{-}ONT}(O)=(s_1,s_2) \wedge max(O)=1 \wedge \rightarrow s_2 \in S_{EW}.$$

Wenn die Maximum-Kardinalität des Attributssymbols größer als 1 ist, muss es ein mengenwertiges Zielkonzept haben:

$$\forall\ O \in OPS_{OS}: typ_{OPS\text{-}ONT}(O)=(s_1,s_2) \wedge max(O)>1 \wedge \rightarrow s_2 \in S_{MW}.$$

Der Fall mit einer Maximum-Kardinalität eines Attributssymbols von genau 0 macht nur Sinn, wenn die Minimum-Kardinalität auch genau 0 ist. Die beiden anderen Fälle für die Minimum-Kardinalität sind aus rechnerischen Gründen ausgeschlossen. Dennoch stellt der zulässige Fall mit einer Kardinalität min(O)=0 und max(O)=0 einen Sonderfall dar. Es würde nämlich bedeuten, dass das Attributssymbol O durch eine Operation interpretiert wird, die keine Bilder hat. Der Sinn dieser Konstellation liegt darin, genau diese Fälle in der Ontologie erfassen zu können. Beispielsweise könnte das Attributssymbol „Verheiratet_mit“, dass für das Konzept „Unverheiratete_Person“ zulässig sei, diesen Fall darstellen.[2)]

Die Kardinalitäts-Konstellationen (min(O)≥1,max(O)=1) und (min(O)>1,max(O)=1) sind beide für alle Attributssymbol ausgeschlossen:

$$\forall i \in I: O_i \in OPS_{OS} \rightarrow \neg(min(O)\geq 1 \wedge max(O)=1) \wedge \neg(min(O)>1,max(O)=1)$$

In der Tabelle 2 sind die Zusammenhänge zwischen den Kardinalitätsfunktionen dargestellt. Die Felder der Tabelle enthalten Angaben dazu, aus welcher Konzeptmenge das Zielkonzept eines Attributssymbols bei gegebenen Kardinalitäten sein muss.

$O \in OPS$ $typ(O)=(s_1,s_2)$	max(O)=0	max(O)=1	max (O)>1
min (O)=0	$s_2 \in S_{OS}$	$s_2 \in S_{EW}$	$s_2 \in S_{MW}$
min (O)=1	undefiniert	$s_2 \in S_{EW}$	$s_2 \in S_{MW}$
min (O)>1	undefiniert	undefiniert	$s_2 \in S_{MW}$

Tabelle 2: Zusammenhänge zwischen den Kardinalitätsfunktionen

1) Grundsätzlich könnte auch ein mengenwertiges Konzept $men(s) \in S_{MW}$ das Zielkonzept zu einem Attributssymbol O mit Maximum-Kardinalität max(O)=1 sein. Denn eine konzeptspezifische Objektmenge $OB_{men(s)}$ zu einem mengenwertigen Konzept $men(s) \in S_{MW}$ ist die Potenzmenge $pot(OB_s)=OB_{men(s)}$ über der Objektmenge OB_s zum einwertigen Konzept $s \in S_{EW}$, von der das mengenwertige Konzept men(s) abgeleitet ist. In dieser Potenzmenge $pot(OB_s)=\{OB' | OB' \subseteq OB_s\}$ sind auch die formalen Objekte enthalten, die in OB_s enthalten sind. Denn bei den formalen Objekten $ob_i \in OB_{men(s)}$ handelt es sich nunmehr um alle denkbaren Teilmengen von OB_s. Somit sind auch solche Mengen $ob_1,...,ob_n$ als Elemente in $OB_{men(s)}$ enthalten, die eine Mächtigkeit $|ob_1|=...=|ob_n|=1$ haben. Die (einzigen) Elemente dieser Mengen sind ebenso Elemente von OB_s. Somit könnten sie Werte eines Attributs o zu einem Attributssymbol O mit max(O)=1 sein. Weitere Elemente $ob_{n+1},...,ob_m$ von $OB_{men(s)}$ haben allerdings eine Mächtigkeit $|ob_{n+1}|=...=|ob_m|>1$. Um diese Mengen als Werte des Attributs o auszuschließen, darf das Zielkonzept von O keine mengenwertiges Konzept sein.

2) Vgl. McGuiness/van Harmelen (2004) Abschnitt 3.5.

2.2.1.2.6 Relationssymbole RS_{OS}

Die Relationssymbole equal, element_of, add, subt, multip, div, concat, greater und greater_or_equal werden benötigt, um Formeln über einer Ontologie_Signatur konstruieren zu können. Mittels einer Teilmenge aller Formeln über einer Ontologie-Signatur werden *Inferenzregeln* konstruiert. Inferenzregeln werden dazu verwendet, die Menge aller Algebren einzuschränken, die in einer Modellbeziehung zu der Ontologie-Signatur stehen.

Die Typisierung der Relationssymbole erfolgt durch die Relationssymbol-Typsierungsfunktion:

$$typ_{RS}: RS_{OS} \rightarrow S_{OS}^*.$$

Die Typisierungen lauten im einzelnen

(1.) typ_{RS}(equal) = (T,T),

(2.) typ_{RS}(element_of) = (s_1,s_2) mit $s_1 \in S_{EW}$ und $s_2 \in S_{MW}$,[1)]

(3.) typ_{RS}(add) = (Real,Real,Real),

(4.) typ_{RS}(subt) = (Real,Real,Real),

(5.) typ_{RS}(multip) = (Real,Real,Real),

(6.) $typ_{RS\text{-}OS}$(div) = (Real,Real,Real),

(7.) $typ_{RS\text{-}OS}$(greater) = (Real,Real),

(8.) typ_{RS}(greater_or_equal) = (Real,Real).

(9.) typ_{RS}(concat) = (String,String,String),

(10.) typ_{RS}(cut) = (String,Integer,String)

Die Beschränkung auf die Relationssymbole aus RS_{OS} dürfte zunächst befremdlich wirken, da üblicherweise Ontologien mit dem *ganzen* Ausdruckvermögen der Prädikatenlogik in Einklang gebracht werden. Hier wird jedoch ein Ansatz verfolgt, der möglicherweise zunächst den Charakter einer *Einschränkung* des Ausdrucksvermögen von Ontologien haben dürfte. In der Tat ist dieser Ansatz hinsichtlich seiner Ausdrucksmächtigkeit der Prädikatenlogik unterlegen. Das Ausdruckspotenzial der Prädikatenlogik wird allerdings für die Konstruktion von Ontologien auch nicht benötigt.

---

1) Für s_1 und s_2 gilt, dass es jeweils die oberen Schranken der partiellen Ordnung $(S_{EW},\leq_S)$ bzw. $(S_{MW},\leq_S)$ sein sollten.

2.2.1.3 Beispiel zu Ontologie-Signaturen

Im Folgenden ist eine beispielhaft Ontologie-Signatur gegeben:

SIG_{OS}-Bsp=
S_{OS}
S_{EW}: {T,Frau,Mann,Person,Unternehmen,Datum, Menge};
S_D:{Real_pos};
S_{MW}: {men(Unternehmen), men(Person)};

men: {(Unternehmen,men(Unternehmen)), (Person,men(Person))};

$\leq_S$: {(Entitaet,T), (Datum,T), (Menge,T),
(Person,Entitaet), (Unternehmen,Entitaet)
(Frau,Person), (Mann,Person),
(men(Unternehmen),Menge), (men(Person),Menge),
(Rat_pos,Datum)};

$=_S$: {(Person,Mensch)};

$\|_S$: {(Frau,Mann)};

OPS_{OS}

OPS_{st}:	{Arbeitet_fuer	:Person	→	Unternehmen,
	Hat_Gehalt	:Person	→	Rat_pos,
	Hat_fruehere_Arbeitgeber	:Person	→	men(unternehmen),
	Hat_Mitarbeiter	:Unternehmen	→	men(Person),
	Tochterunternehmen_von	:Unternehmen	→	men(Unternehmen),
	Mutterunternehmen_von	:Unternehmen	→	men(Unternehmen)};
$OPS_{SYM.EW}$:	{Verheiratet_mit	:Person	→	Person};
$OPS_{SYM.MW}$:	{Arbeitet_mit	:Person	→	men(Person)};
OPS_{TRANS}:	{Ist_Vorgesetzter_von	:Person	→	men(Person)};

inv_{OPS}:
inv_{EW}: {Verheiratet_mit,Verheiratet_mit};
$inv_{MW.1}$: {(Arbeitet_fuer,Hat_Mitarbeiter)};
$inv_{MW.2}$: {(Tochterunternehmen_von,Mutterunternehmen_von),
Arbeitet_mit,Arbeitet_mit}.

RS_{OS}:	equal	T	→T;
	element_of	Entitaet	→Menge;

In der beispielhaften Signatur sind aus Gründen der Übersichtlichkeit lediglich die *direkten* Subkonzeptrelationen $\leq_S$ zwischen Konzepten angegeben. In der vollständigen Auflistung müssten zudem der transitive und der reflexive Abschlüsse jedes Elements der Relation $\leq_S$ angegeben werden. Zudem wurde die Attributssymboltypisierungsfunktion $typ_{OPS\text{-}ONT}$ und die Kardinalitäten min und max weggelassen. Die einzelnen Bilder der Funktion $typ_{OPS\text{-}ONT}$ zu

den Attributsymbolen wurden in der Zeilen zu OPS_{OS} bei jedem Attributssymbol schematisch dargestellt.

Ontologie-Signaturen können mittels verschiedener graphischer Modellierungssprachen visualisiert werden. Hierzu gehören u.a. UML-Klassendiagramme[1)] und hyperbolische Graphen. Die meisten der Ansätze basieren auf *semantischen Netzen*[2)]. In der Abbildung ist ein modifiziertes semantisches Netz zu einem Auszug aus der oben aufgeführten Ontologie-Signatur wiedergegeben.

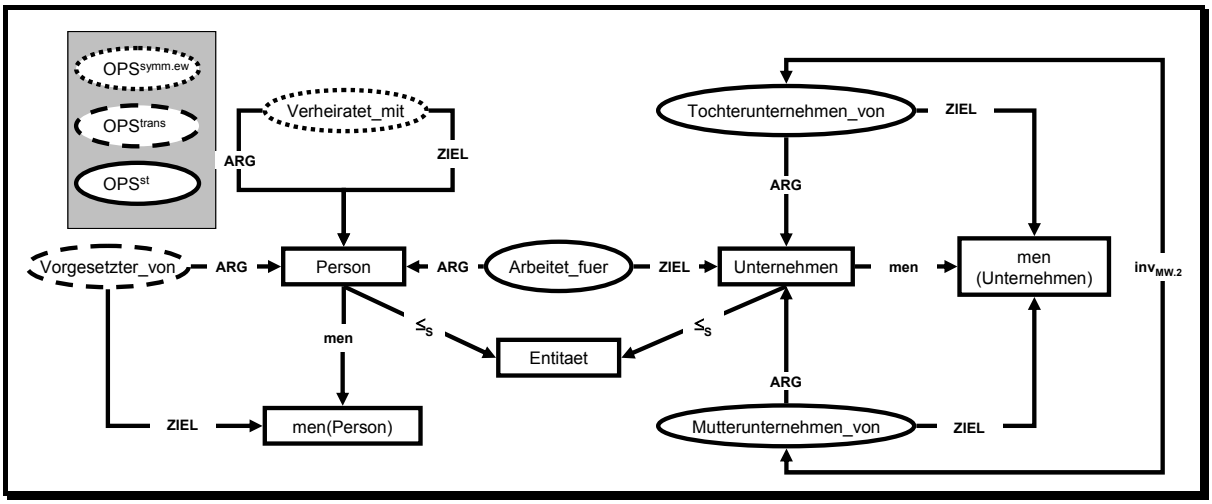

Abbildung 6: Graphische Darstellung von Ontologie-Signaturen

2.2.2 Ontologie-Algebren

2.2.2.1 Definition von Ontologie-Algebren

Eine Ontologie-Algebra A_{OS} zu einer Ontologie-Signatur SIG_{OS} ist ein Drei-Tupel

$$A_{OS}=(OBF_{OS}, OPF_{OS}, RF_{OS}).$$

Die Komponenten einer Ontologie-Algebra sind:

(1.) eine Familie $OBF_{OS}=(OB_s)_{s\in S}$ von Operationsmengen, für die gilt:

(1.1.) $\forall s_1\in S_{EW}, s_2\in S_{MW}: \rightarrow OB_{men(s)}=pot(OB_s)$;

(1.2.) $\forall s_1,s_2\in S_{OS}: s_1 \leq_S s_2 \rightarrow OB_{s.1}\subseteq OB_{s.2}$;

(1.3.) $OB_{ONT}=\cup OB_s$ für alle $s\in S_{OS}$;

(1.4.) $\forall s_1,s_2\in S_{OS}: s_1 =_S s_2 \rightarrow OB_{s.1}=OB_{s.2}$;

---

1) Vgl. CRANEFIELD/PURVIS (2002) S. 31 ff.

2) Vgl. SOWA (1984) S. 69 ff.; SOWA (2000) S. 4. u. 23.

(1.5.) $\forall s_1,s_2 \in S_{OS}: s_1 \|_S s_2 \rightarrow OB_{s.1} \cap OB_{s.2} = \varnothing$;

(1.6.) $OB_{Integer}=\mathcal{Z}$, $OB_{int\_pos}=\mathcal{Z}_+$, $OB_{int\_neg}=\mathcal{Z}_-$,

$OB_{Real}=\mathcal{R}$, $OB_{Real\_pos}=\mathcal{R}_+$, $OB_{Real\_neg}=\mathcal{R}_-$,

$OB_{Bool}=\{true,false\}$, $OB_{String}=ALPH^*$

(2.) eine Familie $OPF_{OS}=(o_i)_{i=1...I,\ I\in\mathcal{N}}$ von Operationsmengen, für die gilt:

(2.1.)

(2.1.1.) $\forall\ O \in OPS_{SYM.EW},\ ob_1,ob_2 \in OB_{ONT}$:

$o(ob_2)=ob_1 \leftrightarrow o(ob_1)=ob_2$;

(2.1.2.) $\forall\ s \in S_{EW},\ men(s) \in S_{MW},\ O \in OPS_{SYM.MW},\ ob_1,ob_2,ob_3,ob_4 \in OB_{ONT}$:

$typ_{OPS\text{-}ONT}(O)=(s,men(s)) \wedge o(ob_1)=ob_3 \wedge ob_2 \in ob_3 \rightarrow$

$\exists\ ob_4 \in OB_{men(s)}: o(ob_2)=ob_4 \wedge ob_1 \in ob_4$;

(2.2.) $\forall\ s \in S_{EW},\ men(s) \in S_{MW}\ O \in OPS^{trans},\ ob_1,ob_2,ob_3,ob_4,ob_5 \in OB_{ONT}$:

$typ_{OPS\text{-}ONT}(O)=(s,men(s)) \wedge$

$o(ob_1)=ob_4 \wedge ob_2 \in ob_4 \wedge o(ob_2)=ob_5 \wedge ob_3 \in ob_5 \rightarrow ob_3 \in ob_4$;

(2.3.)

(2.3.1) $\forall\ s_1,s_2 \in S_{OS},\ ob_1,ob_2 \in OB_{ONT}$:

$(O_1,O_2) \in inv_{EW} \wedge typ_{OPS\text{-}ONT}(O_1)=(s_1,s_2) \wedge typ_{OPS\text{-}ONT}(O_2)=(s_2,s_1) \wedge$

$o_1(ob_1)=ob_2 \leftrightarrow o_2(ob_2)=ob_1$,

(2.3.2)

(2.3.2.1.) $\forall\ O_1,O_2 \in OPS_{OS},\ s_1,s_2,s_3,s_4 \in S_{OS}$:

$(O_1,O_2) \in inv_{MW.1} \wedge typ_{OPS\text{-}ONT}(O_1)=(s_1,s_2) \wedge$

$typ_{OPS\text{-}ONT}(O_2)=(s_3,s_4) \wedge s_2=s_3 \wedge s_4=men(s_1) \rightarrow$

$(\forall\ ob_1,ob_2 \in OB_{ONT}:\ o_1(ob_1)=ob_2 \rightarrow$

$\exists\ ob_3 \in OB_{men(s.1)}: o_2(ob_2)=ob_3 \wedge ob_1 \in ob_3)$

$\wedge$

$(\forall\ ob_1,ob_3 \in OB: o_2(ob_2)=ob_3 \wedge ob_1 \in ob_3 \rightarrow o_1(ob_1)=ob_2)$,

(2.3.2.2.) $\forall O_1,O_2 \in OPS_{OS},\ ob_1,ob_2,ob_3 \in OB_{ONT}$:

$(O_1,O_2) \in inv_{MW.2} \wedge$

$typ_{OPS\text{-}ONT}(O_1)=(s_1,s_2) \wedge typ_{OPS\text{-}ONT}(O_2)=(s_3,s_4) \wedge$

$s_2=men(s_3) \wedge s_4=men(s_1)$

$o_1(ob_1)=ob_3 \wedge ob_2 \in ob_3 \rightarrow$

$\exists ob_4 \in OB_{men(s.1)}: o_2(ob_2)=ob_4 \wedge ob_1 \in ob_4$

(2.4.)

(2.4.1) $\forall O \in OPS_{OS}, s_1,s_2 \in S_{OS}$:

$typ_{OPS\text{-}ONT}(O)=(s_1,s_2) \wedge min(O)=0 \wedge max(O)=0 \rightarrow$

$\forall ob_1 \in OB_{ONT} \neg \exists ob_2 \in OB_{ONT}: o(ob_1)=ob_2$,

(2.4.2.) $\forall O \in OPS_{OS}, s_1,s_2 \in S_{OS}, n \in \mathcal{N}_+$:

$typ_{OPS\text{-}ONT}(O)=(s_1,s_2) \wedge max(O)=n \wedge n>1 \rightarrow$

$\forall ob_1 \in OB_{s.1}, ob_2 \in OB_{s.2}: o(ob_1)=ob_2 \rightarrow |ob_2| \leq n$,

(2.4.3.) $\forall O \in OPS_{OS}, s_1,s_2 \in S_{OS}, n \in \mathcal{N}_+$:

$typ_{OPS\text{-}ONT}(O)=(s_1,s_2) \wedge max(O)=min(O)=n \rightarrow$

$\forall ob_1 \in OB_{s.1}, ob_2 \in OB_{s.2}: o(ob_1)=ob_2 \rightarrow |ob_2|=n$,

(2.4.3.) $\forall O \in OPS_{OS}, s_1,s_2 \in S_{OS}: typ_{OPS\text{-}ONT}(O)=(s_1,s_2) \wedge max(O)=1 \rightarrow$

$\forall ob_1 \in OB_{s.1}, ob_2, ob_3 \in OB_{s.2}: o(ob_1)=ob_2 \wedge o(ob_1)=ob_3 \rightarrow ob_2=ob_3$.

(3.) eine Familie RF_{ONT} von Relationsmengen mit den Mitgliedern

$RF_{OS}=\{$ $(= \subseteq OB_T \times OB_T)$,

$(\in \subseteq OB_T \times OB_T)$

$(+ \subseteq OB_{Real} \times OB_{Real} \times OB_{Real})$,

$(- \subseteq OB_{Real} \times OB_{Real} \times OB_{Real})$,

$(\times \subseteq OB_{Real} \times OB_{Real} \times OB_{Real})$,

$(/ \subseteq OB_{Real} \times OB_{Real} \times OB_{Real})$,

$(konk \subseteq OB_{string} \times OB_{string} \times OB_{string})$,

$(cut \subseteq OB_{string} \times OB_{int} \times OB_{string})$,

$(> \subseteq OB_{Real} \times OB_{Real} \times OB_{Real})$,

$(\geq \subseteq OB_{Real} \times OB_{Real} \times OB_{Real})\}$.

$OBF_{ONT}=(OB_s)_{s \in S}$ ist eine Mengenfamilie, deren Mitglieder Mengen OB_s formaler Objekte $ob_1,...,ob_n$ sind. Die Menge $OB_{ONT}=\cup_{k \in K} OB_{s.k}$ wird als das *Universum* zu einer Ontologie-Signatur bezeichnet. Es beinhaltet alle formalen Objekte aus den sortenspezifischen Objektmengen $OB_s,...,OB_{s.k}$.

Jedem Konzept $s \in S_{OS}$ einer Ontologie-Signatur SIG_{OS} ist eine konzeptspezifische Menge $OB_s \in OBF_{ONT}$ von formalen Objekten $ob_1,...,ob_n \in OB_s$ zugeordnet. Bei den Objektmengen, die mengenwertigen Konzepten men(s) für $s \in S_{MW}$ zugeordnet sind, handelt es sich um Mengen von Mengen. Somit sind es Mengenfamilien. Jedem mengenwertigen Konzept

men(s)$\in S_{MW}$ wird entsprechen Eigenschaft (1.1.) der Objektbereich $OB_{men(s)}=pot(OB_s)$ zugeordnet, wobei $OB_{men(s)}$ die Potenzmenge der zu Grunde gelegten Objektmenge OB_s ist. Somit gilt für jede Objektmenge zu einer Mengensorte:

$$\forall s \in S_{EW},\ men(s) \in S_{MW}: OB_{men(s)} = pot(OB_s)$$

$$\text{mit } pot(OB_s) = \{OB_s' \mid OB_s' \subseteq OB_s\}.$$

Durch die partielle Ordnung $(S,\leq_S)$ auf der Menge der Konzepte ergibt sich Eigenschaft (1.2.) der Mengenfamilie OBF:[1)]

$$OB_{s.1} \subseteq OB_{s.2} \text{ für alle } s_1, s_2 \in S_{OS} \text{ mit } s_1 \leq_S s_2.$$

Jedes Objekt ob_1, das zu einer s_1-spezifischen Objektmenge $OB_{s.1}$ gehört, gehört auch zu einer s_2-spezifischen Objektmenge $OB_{s.2}$ wenn $s_1 \in S_{OS}$ Subkonzept von $s_2 \in S_{OS}$ ist. Dadurch dass, die Subkonzeptrelation $\leq_S$ als partielle Ordnung auf der Konzeptmenge S_{OS} definiert ist, ergibt sich für die konzeptspezifischen Objektmengen, dass sie ineinander verschachtelt sein können. In der Abbildung 7 ist das Prinzip der Verschachtelung von Objektmengen verdeutlicht.

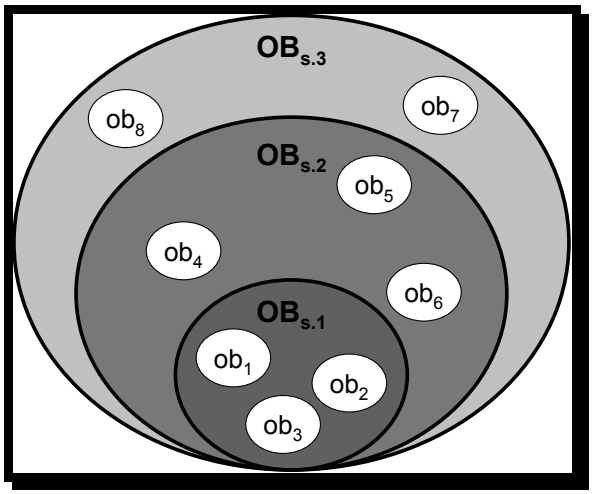

Abbildung 7: Verschachtelung sortenspezifischer Objektmengen

Die Objektmengen $OB_{s.1}$,$OB_{s.2}$ und $OB_{s.3}$ zu den Konzepten $s_1,s_2,s_3 \in S_{OS}$ stehen in Teilmengenbeziehung $OB_{s.1} \subseteq OB_{s.2} \subseteq OB_{s.3}$ zueinander, da für die Konzepte s_1,s_2,s_2 die Subkonzeptrelationen $s_1 \leq_S s_2 \leq_S s_3$ definiert sind. Daher sind die formalen Objekte $ob_1,ob_2,ob_3 \in OB_{s.1}$ auch in den Objektmengen $OB_{s.2}$ und $OB_{s.3}$ und die formalen Objekte $ob_4,ob_5,ob_6 \in OB_{s.2}$ auch

1) Vgl. BEIERLE (1993) S. 194; EHRICH ET AL. (1989) S. 178; GOGUEN/MESEGUER (1992) S. 226; HATZILIYGEROUDIS/REICHGELT (1997) S. 256; KANEIWA (2001) S. 30; LOECKX (1996) S. 235; MÜLLER (1999) S. 12.

in der Objektmenge $OB_{s.3}$ enthalten. Für die formalen Objekte ob_7 und ob_8 gilt: $ob_7,ob_8 \in OB_{s.3}$.

Die Objektmenge OB_T zu dem maximalen Konzept $T \in S_{OS}$ entspricht dem oben angesprochenen Universum OB_{ONT} aus Eigenschaft (1.3.). Für jedes Konzept $s_i \in S_{OS}$ gilt nämlich: $s_i \leq_S T$ für i=1,...,I und $I \in \mathcal{N}_+$. Somit sind alle formalen Objekte $ob_1,...,ob_n \in OB_{s.k}$ zu jedem Konzept s_k auch in der Objektmenge OB_T enthalten. Daher gilt:

$$OB_T = \cup_{k \in K} OB_{s.k} = OB_{ONT}.$$

Die Eigenschaft (1.4.) der Mengenfamilie OBF_{ONT} fordert, dass zwei Objektmengen $OB_{s.1},OB_{s.2} \in OBF$ dann gleich sind, wenn die Konzepte $s_1,s_2 \in S_{OS}$ in *Äquivalenzrelation* $s_1 =_S s_2$ zueinander stehen. Da es sich bei s_1,s_2 um äquivalenten Konzepte handelt, müssen die konzeptspezifischen Objektmengen $OB_{s.1},OB_{s.2}$ alle Elemente gemeinsam haben. Die Gleichheit der konzeptspezifischen Objektmengen lässt sich auch als Formel

$$\forall s_1,s_2 \in S_{OS}: (s_1 =_S s_2) \rightarrow (\forall ob \in OB: ob \in OB_{s.1} \leftrightarrow ob \in OB_{s.2})$$

ausdrücken. Die Konklusionsformel der oben angegebenen Subjugatsformel ist äquivalent mit der zuvor in (1.4.) verwendeten Formel $OB_{s.1}=OB_{s.2}$.

Die Eigenschaft (1.5.) der Mengenfamilie OBF fordert, dass zwei Objektmengen $OB_{s.1},OB_{s.2} \in OBF$ dann disjunkt zueinander sind, wenn die Konzepte $s_1,s_2 \in S_{OS}$ in Exklusivitätsrelation zueinander stehen. Das bedeutet, dass kein formales Objekt ob aus einer s_1-spezifischen Objektmenge $OB_{s.1}$ auch in einer s_2-spezifischen Objektmenge $OB_{s.2}$ enthalten sein darf, wenn für die Konzepte $s_1,s_2 \in S_{OS}$ gilt: $s_1 \|_S s_2$.

Durch die Eigenschaft (1.6.) werden die Objektmengen zu den Datenkonzepten festgelegt. Das Datenkonzept Integer wird die Menge $\mathcal{Z}$ der natürlichen Zahlen zugeordnet. Dem Datenkonzept Int_pos wird die Menge $\mathcal{Z}_+$ der positiven ganzen Zahlen zugeordnet. Dem Datenkonzept Int_neg wird die Menge $\mathcal{Z}_-$ der negativen ganzen Zahlen zugeordnet. Dem Datenkonzept Real wird die Menge $\mathcal{R}$ der reellen Zahlen zugeordnet. Dem Datenkonzept Real_pos wird die Menge $\mathcal{R}_+$ der positiven reellen Zahlen zugeordnet. Dem Datenkonzept Real_neg wird die Menge $\mathcal{R}_-$ der negativen reellen Zahlen zugeordnet. Dem Datenkonzept Bool wird die Menge {true,false} der Wahrheitswerte zugeordnet. Dem Datenkonzept String wird die Menge ALPH* aller Wörte über dem Alphabet ALPH={0,1,...,9,a,b,...,z,A,B,...,Z,,,;,.,:,+,-,?,!,$,%;/,(,),=, ,} zugeordnet.

Die Mengenfamilie $OPF_{ONT}=(o_i)$ mit i=1,...,I und $I \in \mathbb{N}$ enthält *Operationen*, die als Vor- und Nachbereich jeweils eine sortenspezifische Menge formaler Objekte beinhalten. Der Vor- und der Nachbereich einer Operationen o_i ist entsprechend der Typisierung $typ_{OPS\text{-}ONT}(O_i)$ des Attributssymbols $O_i \in OPS_{OS}$ gegeben, dass durch die Operation o_i interpretiert wird. Daher werden die Elemente der Mengenfamilie OPF_{ONT} auch als *Attribute* angesprochen. Die Bilder zu den Elementen von OPF_{ONT} werden als *Attributswerte* bezeichnet. Es gilt somit, dass für eine Attribut o_i, zu einem Attributssymbol O_i mit der Typisierung $typ_{OPS\text{-}ONT}(O_i)=(s_1,s_2)$ die Operationsdeklaration $o_i:OB_{s.1} \rightarrow OB_{s.2}$ gegeben ist.

Im Folgenden werden die Eigenschaften unter dem Eigenschaftsoberpunkt (2.) anhand von Beispielen dargestellt.

Beispiel zu Eigenschaft (2.1.1.):

Wenn gilt:

$OPS_{SYM.EW}$: {Verheiratet_mit: Person → Person};
OB_{Person}: {adam,eva};
OPF: {verheiratet_mit: $OB_{Person} \rightarrow OB_{Person}$
verheiratet_mit(hans) = eva};

dann muss auch gelten:

OPF: {verheiratet_mit(eva) = adam.

Beispiel zu Eigenschaft (2.1.2.):

Wenn gilt:

$OPS_{SYM.MW}$: {Arbeitet_mit: Person → men(Person)};
OB_{Person}: {hans,werner,...};
$OB_{men(Person)}$: {{hans},{werner},{hans,werner},{hans,...},{hans,werner,...},...};
OPF: arbeitet_mit: $OB_{Person} \rightarrow OB_{men(Person)}$;
arbeitet_mit(hans) = {...,werner,...}

dann muss auch gelten:

OPF: arbeitet_mit(werner)={...,hans,...};

Die Eigenschaft 2.2.) aller Ontologie-Algebren berücksichtigt transitive Attributssymbole.

Beispiel zu Eigenschaft (2.2.):

Attribute zu transitiven Attributsymbolen OPS_{TR}:

Wenn gilt:

OPS_{TR}:	{Vorgesetzter_von:	Person → men(Person)};
OB_{Person}:	{hans,werner,uli...};	
$OB_{men(Person)}$:	{{hans},{werner},{hans,werner},{hans,uli}	
	{hans,werner,uli,...},{hans,werner,...},...};	
OPF:	vorgesetzter_von:	$OB_{person} \rightarrow OB_{men(person)}$;
	vorgesetzter_von(hans)	= {...,werner,...};
	vorgesetzter_von(werner)	= {...,uli,...};

dann muss auch gelten:

OPF: vorgesetzter_von(hans)={...,werner,uli,...};

Es ist dabei zu beachten, dass es sich bei der Menge {...,werner,...}, dass dem Bild der Operation vorgesetzter_von an der Stelle hans entspricht, um die gleiche Menge handelt, die unten als {...,uli,...} handelt. Das heißt, dass beide formalen Objekte werner,uli$\in OB_{Person}$ in der Menge ob$\in OB_{men(Person)}$ enthalten sind, die dem Bild ob=vorgesetzter_von(hans) entspricht.

Beispiel zu Eigenschaft (2.3.1.):

Wenn gilt:

OPS:	{Hat_Frau:	Mann → Frau,
	Hat_Mann:	Frau → Mann};
inv_{EW}	(Hat_Frau,Hat_Mann)	
OB_{Person}:	{adam,eva};	
OPF:	{hat_mann:	$OB_{Frau} \rightarrow OB_{Mann}$,
	hat_frau:	$OB_{Mann} \rightarrow OB_{Frau}$,
	hat_frau(adam)	= eva},

dann muss auch gelten:

OPF: hat_mann(eva) = adam.

Beispiel zu Eigenschaft (2.3.2.1.):

Wenn gilt:

OPS:	{Hat_Mutter:	Kind	$\rightarrow$ Frau,
	Hat_Kinder:	Frau	$\rightarrow$ men(Kind)},
$inv_{MW.1}$	(Hat_Mutter,Hat_Kinder),		
OB_{Kind}	{kain,abel,...},		
OB_{Frau}	{eva},		
OPF:	{hat_mutter:	$OB_{Kind} \times OB_{Frau}$,	
	hat_kinder:	$OB_{Frau} \times OB_{men(Kind)}$,	

dann gilt:

1.) Wenn gilt:

OPF: hat_mutter(kain) = eva;

dann muss auch gelten:

OPF: hat_Kinder(eva) ={...,kain,...},

2.) Wenn gilt:

OPF: hat_Kinder(eva) ={...,kain,...},

dann muss auch gelten:

OPF: hat_mutter(kain) ={eva}.

Beispiel zu Eigenschaft (2.3.2.2.):

Wenn gilt

OPS:	{Hat_Kompetenz:	Person	$\rightarrow$ men(Kompetenz),
	Wird_beherrscht_von:	Kompetenz	$\rightarrow$ men(Person)},
$inv_{MW.2}$:	(Hat_Kompetenz,Wird_beherrscht_von),		
OB_{Person}:	{hans,werner,uli...};		
$OB_{men(Person)}$:	{{hans},{werner},{hans,werner},{hans,uli}		
	{hans,werner,uli,...},{hans,werner,...},...};		
$OB_{Kompetenz}$:	{java,c++,sql,englisch},		
OPF:	{hat_kompetenz:	$OB_{Person} \times OB_{men(Kompetenz)}$,	
	wird_beherrscht_von:	$OB_{Kompetenz} \times OB_{men(Person)}$,	
	hat_kompetenz(werner)	={java,sql},	
	hat_kompetenz(uli)	={sql,englisch);	

dann gilt:

OPF: wird_beherrscht_von(java) ={werner},
wird_beherrscht_von(sql) ={werner,uli},
wird_beherrscht_von(englisch) ={uli}.

Mit den Eigenschaften (2.4.1.) und (2.4.2.) wird gewährleistet, dass die Kardinalitäten der Attributsymbole berücksichtigt werden.

2.2.2.2 Beispiel zu Ontologie-Agebren

Eine Algebra A_{OS}-Bsp zu der oben aufgeführten Ontologie-Signatur SIG_{OS}-Bsp kann wie folgt lauten:

A_{OS}-Bsp=
OBF: {OB_{Person} ={hans,peter,werner,sabine,elke},
OB_{Mann} ={hans,peter,werner},
OB_{Frau} ={sabine,elke},
$OB_{Unternehmen}$ ={arnulf_GmbH,bruederich_AG,bleidorf_KG},
$OB_{men(Unternehmen)}$ =*pot*($OB_{Unternehmen}$),
OB_{Real_pos} =$\mathcal{R}_+$};
OPF: {verheiratet_mit(sabine) = hans,
verheiratet_mit(hans) = sabine,
verheiratet_mit(werner) = elke,
verheiratet_mit(elke) = werner,
arbeitet_fuer(hans) = arnulf_GmbH,
arbeitet_fuer(werner) = Arnulf_GmbH,
hat_Gehalt(hans) = 3144.14,
ist_Vorgesetzer_von(hans) ={werner,elke},
ist_Vorgesetzter_von(werner) ={elke};
hat_fruehere_Arbeitgeber(elke) = {bleidorf_KG,bruederich_AG},
tochterunternehmen_von(arnulf_GmbH) = {bruederich_AG},
mutterunternehmen_von(bruederich_AG) = {arnulf_GmbH},
Mitarbeiter(bruederich_AG) = {elke,werner}}.

Die Objektmengen der Ontologie-Algebra A_{OS}-Bsp interpretieren extensional die Konzepte aus der Ontologie-Signatur SIG_{OS}-Bsp. Beispielsweise wird das Konzept Person$\in S_{OS}$ von der Objektmenge OB_{Person} interpretiert. Die Attributssymbole aus der Ontologie-Signatur SIG_{OS}-Bsp werden durch Attribute aus der Ontologie-Algebra A_{OS}-Bsp interpretiert. Z.B. interpretiert das Attribut verheiratet_mit: $OB_{Person} \rightarrow OB_{person}$ das Attributssymbol Verheiratet_mit$\in OPS_{OS}$ mit $typ_{OPS\text{-}ONT}$(Verheiratet_mit)=(Person,Person).

Die Algebra A_{OS}-Bsp ist nur eine mögliche Interpretation der Ontologie-Signatur SIG_{OS}-Bsp. Sie ist nur *ein* Element der Menge $\mathcal{ALG}$(SIG_{OS}-Bsp) der in Betracht kommenden Interpretationen für SIG_{OS}-Bsp. Wie sich bereits aus dem Beispiel andeutet, kommen andere Algebren in Betracht, die die Menge der Interpretationen der Ontologie-Signatur einschränken. Um An-

forderungen an die Ontologie-Algebra formulieren zu können, werden – identisch zu algebraischen und relationalen Signaturen – Ausdrücke benötigt. Die algebraische Komponente der Ontologie-Signatur, wird genutzt, um *Terme* auszudrücken. Die algebraische Komponente wird schließlich in Verbindung mit der relationalen Komponente dazu genutzt, *Formeln* auszudrücken.

2.2.3 Ausdrücke über Ontologie-Signaturen

2.2.3.1 Terme über Ontologie-Signaturen

Bei der Definition von Termen über einer Ontologie-Signatur SIG_{ONT} ergeben sich zwei Erweiterungen gegenüber der Termdefinition bei üblichen Signaturen SIG_{AS} und SIG_{RS}. Die erste Erweiterung betrifft die Termdefinition aufgrund der Subkonzeptrelation $\leq_S$. Die zweite Erweiterung betrifft die Termdefinition aufgrund der *Äquivalenzrelation* $=_S$. Die dritte Erweiterung betrifft den Ausschluss von Termen aufgrund der Exklusivitätsrelation $||_S$.

In Abschnitt 2.1.4.1.1 wurden die sortenspezifischen Termmengen definiert als:

(1.) $x \in TERM_s$ für alle $x \in VAR_s$,

(2.) $O \in TERM_s$ für alle $O \in OPS$ mit $typ_{OPS}(O)=\lambda,s$ und

(3.) $O(t_1,...,t_n) \in TERM_s$ für alle $O \in OPS$ mit $typ_{OPS}(op_i)=s_1...s_n,s$ und $t_i \in TERM_{s.i}$ für $i=1...n$.

Regel (1.) wird für die Definition von Termen über Ontologie-Signaturen beibehalten. Alle konzeptspezifischen Variablen sind somit auch konzeptspezifische Terme. Regel (2.) behält zwar weiterhin Gültigkeit, hat allerdings keine Auswirkungen auf die konzeptspezifischen Termmengen über einer Ontologie Signatur SIG_{ONT}, da im vorherigen Abschnitt Attributssymbole O mit leerem Vorbereich $ARG_{OPS\text{-}ONT}(O)=\lambda$ ausgeschlossen wurden. Daher sind auch Grundterme über einer Ontologie-Signatur nicht zugelassen. Demnach besteht die Termmenge TERM über einer Ontologie Signatur SIG_{ONT} nur aus Variablen und Anwendungen von Attributssymbolen auf andere Terme.

Regel (3.) wird beibehalten, kann allerdings verkürzt werden zu:

(3.b.) $O(t_1) \in TERM_{s.2}$ für alle $O \in OPS_{OS}$ mit $typ_{OPS\text{-}ONT}(O)=s_1,s_2$ und $t_1 \in TERM_{s.1}$,

da nur jeweils ein Konzept im Vorbereich von Attributssymbolen über einer Ontologie-Signatur SIG_{ONT} zugelassen ist.

Über einer Ontologie-Signatur SIG_{ONT} werden nun weitere Eigenschaften der konzeptspezifischen Termmengen $TERM_s$ für $s \in S_{OS}$ definiert:

(4.) Jeder Term $t \in TERM_{s.1}$ zum Konzept $s_1 \in S_{OS}$ ist auch ein Term $t \in TERM_{s.2}$ zum Konzept $s_2 \in S_{OS}$, wenn die Subkonzeptrelation $s_2 \leq_S s_1$ existiert:

$$\forall (s_1,s_2) \in \leq_S: t \in TERM_{s.1} \rightarrow t \in TERM_{s.2}.$$

(5.) Jeder Term $t \in TERM_{s.1}$ zum Konzept $s_1 \in S_{OS}$ ist genau dann auch ein Term $t \in TERM_{s.2}$ zum Konzept $s_2 \in S_{OS}$, wenn die *Äquivalenzrelation* $s_1 =_S s_2$ existiert:

$$\forall (s_1,s_2) \in =_S: t \in TERM_{s.1} \leftrightarrow t \in TERM_{s.2}.$$

(6.) Kein Term $t \in TERM_{s.1}$ zum Konzept $s_1 \in S_{OS}$ ist auch ein Term $t \in TERM_{s.1}$ zum Konzept $s_2 \in S_{OS}$, wenn die Exklusivitätsrelation $s_1 \|_S s_2$ existiert:

$$\forall (s_1,s_2) \in \|_S: t \in TERM_{s.1} \rightarrow t \notin TERM_{s.2}.$$

Durch die Regel (4.) zur Identifikation von Termen wird sichergestellt, dass alle Terme die zu einem bestimmten Konzept s_1 gültig sind, auch für alle Superkonzepte $sup(s_1)$ des Konzeptes s_1 gültig sind. Somit ist die Termmenge $TERM_{s.1}$ zu einer Konzept $s_1 \in S_{OS}$ stets eine Teilmenge der Termmenge $TERM_{s.2}$, wenn s_1 ein Subkonzept von s_2 ist:

$$\forall s_1,s_2 \in S_{OS:}\ s_2 \leq_S s_1 \rightarrow TERM_{s.2} \subseteq TERM_{s.1}.$$

Diese Erweiterung wird sich später bei der Definition von Formeln über einer Ontologie-Signatur darin auswirken, dass die Menge der Terme, die im Argument von Relationssymbolen aus der Menge RS_{OS} vorkommen dürfen, um die Vereinigung der Termmengen erweitert wird, die aufgrund obiger Regel sich aus den Subkonzepttermen ergeben. Insbesondere bedeutet diese Erweiterung, dass alle Terme $t_k \in TERM_{s.k}$ zu einem beliebigen Konzept $s_k \in S_{OS}$ auch Terme $t_i \in TERM_T$ zum maximalen Konzept $T \in S_{OS}$ sind. Daher können alle Terme in den Argumentstellen von Relations- und Operationssymbolen verwendet werden, die mit $T \in S_{OS}$ typisiert sind.

In Verbindung mit der Regel (3.) bewirkt die Regel (4.), dass alle Terme $t_1,...,t_5$ zu einem Konzept $s_3 \in S_{OS}$ im Argument eines Attributsymbols O mit $typ_{OPS\text{-}ONT}(s_3,s_4)$ verwendet werden dürfen, auch wenn sie teilweise „ursprüngliche“ Terme zu den Konzepten $s_1,s_2 \in S_{OS}$ mit $s_1 \leq_S s_2 \leq_S s_3$ sind. Der Zusammenhang ist in Abbildung 8 dargestellt.

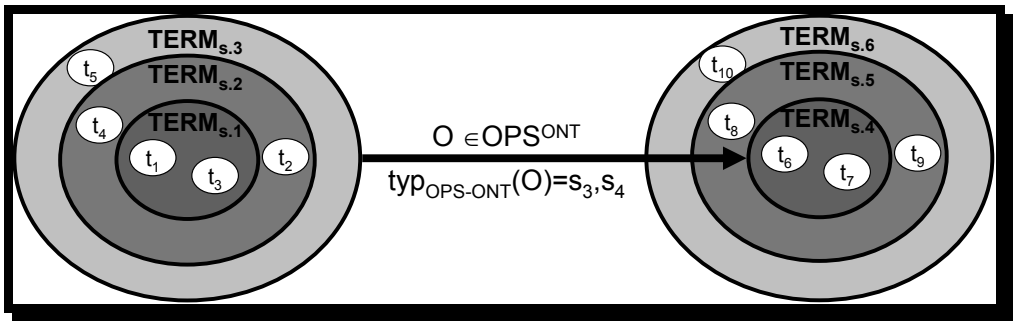

Abbildung 8: Termmengenverschachtelung und Attributssymboltypisierung

Durch die Anwendung des Attributsymbols auf die Terme $t_1,...,t_5$ erhält man einen Term zum Konzept $s_4 \in S_{OS}$. Für alle s_4-spezifischen Terme gilt wiederum, dass sie ebenso Terme zu den Konzepten $s_5,s_6 \in S_{OS}$ sind, wenn gilt: $s_4 \leq_S s_5 \leq_S s_6$.

Die zweite Erweiterung durch Regel (5.) ergibt sich aus der Semantik der *Äquivalenzrelation* $=_S$. Da die *Äquivalenzrelation* als symmetrische Relation ausgezeichnet wurde, gilt:

$$\forall s_1,s_2 \in S_{OS:}\ s_1 =_S s_2 \rightarrow TERM_{s.1}=TERM_{s.2}.$$

Es wurde bereits zuvor darauf hingewiesen, dass die konzeptspezifischen Objektmengen gleich sind, wenn sie aus Konzepten abgeleitet werden, die in *Äquivalenzrelation* zueinander stehen. Analog sind auch die konzeptspezifischen Termengen gleich, da Terme Platzhalter in der Ontologie-Signatur SIG_{OS} für formale Objekte aus einer Ontologie-Algebra A_{OS} fungieren.

Die Erweiterung durch Regel (6.) führt dazu, dass zwei konzeptspezifische Termmengen $TERM_{s.1}$ und $TERM_{s.2}$ disjunkt zueinander sein müssen, wenn sie mit Konzepten $s_1,s_2 \in S_{OS}$ typisiert sind, die miteinander in Exklusivitätsrelation $\|_S$ stehen:

$$\forall s_1,s_2 \in S_{OS:}\ s_2 \|_S s_1 \rightarrow TERM_{s.1} \cap TERM_{s.2}=\varnothing.$$

Trotz der Einschränkung der Regel (3) zur Regel (3.b.) der obigen Termdefinition sind auch weiterhin zusammengesetzte Terme zugelassen. Allerdings ist keine Konstruktion *komplexer* Terme möglich, die sich aus mindestens zwei Termen im Vorbereich eines Operationssymbols zusammensetzen, da nur einstellige Attributssymbole für Ontologien zugelassen sind.

Die Möglichkeit zur Definition zusammengesetzter Terme erweist sich als besonders fruchtbar für die Formulierung von *Anfragen* an eine Ontologie-Algebra, die zu einer Ontologie-Signatur gegeben ist. Zusammengesetzte Terme lassen sich dazu verwenden, *Pfadausdrücke* zu formulieren, die auch verschachtelte Anfragen ermöglichen. Pfadausdrücke sind solche

Ausdrücke, die über eine Kette von Attributssymbolen definiert sind.[1)] Für die beispielhaft angegebene Ontologie-Signatur SIG_{OS}-Bsp. und die zugehörige Algebra ALG_{OS}-Bsp. liefert der Ausdruck

tochterunternehmen_von(arbeitet_fuer(verheiratet_mit(x)))

bei Belegung der Variable $x \in VAR_{Person}$ durch $bel_{Person}(x)$=Sabine mit Sabine$\in OB_{person}$ das Objekt Bruederich_AG mit Bruederich_AG$\in OB_{Unternehmen}$. Das Vorgehen zur Ermittlung des Pfades ist durch die folgende Tabelle gegeben:

Rekursions-stufe	**Terme**	**formale Objekte**
Stufe 0	$x \in TERM_{Person}$ mit $x \in VAR_{Person}$ und $VAR_{person} \subseteq TERM_{Person}$	$tausw_{Person}(x)=bel_{Person}(x)$ $=sabine \in OB_{Person}$
Stufe 1	$(Verheiratet\_mit(x)) \in TERM_{Person}$	verheiratet_mit(sabine) $=hans \in OB_{Person}$
Stufe 2	$(Arbeitet\_fuer(y)) \in Term_{Unternehmen}$ mit $(y=Verheiratet\_mit(x)) \in TERM_{Person}$	arbeitet_fuer(hans) $=arnulf\_GmbH \in OB_{Unternehmen}$
Stufe 3	$(Tochterunternehmen\_von(z)) \in TERM_{Unternehmen}$ mit $(z=Arbeitet\_fuer(y)) \in TERM_{Unternehmen}$ und $(y=Verheiratet\_mit(Sabine)) \in TERM_{Person}$	tochterunternehmen_von(arnulf_GmbH) $=Bruederich\_AG \in OB_{Unternehmen}$

Tabelle 3: Rekursive Konstruktion und Auswertung von Termen

In der Tabelle 3 ist exemplarisch aufgezeigt, wie das „endgültige“ Objekt Bruederich_AG aus der konzeptspezifischen Objektmenge $OB_{Unternehmen}$ ermittelt werden kann. Die Unterscheidung zwischen der Spalte $TERM_s$ und $tausw_s$ spiegelt die konsequente Unterscheidung zwischen der Menge aller Terme über der Ontologie-Signatur SIG_{ONT} und der Menge aller formalen Objekte aus der Ontologie-Algebra A_{OS} wider. Während die erstgenannte Menge lediglich Ausdrücke beinhaltet, die aus der Anwendung von Attributssymbolen auf Terme und Variablen bestehen, umfasst die die zweite genannte Menge nur „konkrete“ formale Objekte aus Objektmengen und ebenso „konkrete“ Operationen auf den formalen Objektmengen. Das Komplement zu „konkreten“ formalen Objekten und Operationen in der Ontologie-Algebra sind die Konzepte und Attributssymbole in der zugehörigen Ontologie-Signatur.

1) Vgl. KIFER ET AL. (1995) S. 817 ff.; UPHOFF (1997) S. 47 für die Umsetzung von Pfadausdrücken in der Wissensrepräsentationssprache F-LOGIC. Vgl. KARVOUNARAKIS ET AL. (2002) S. 596 für die Umsetzung von Pfadausdrücken in der Anfragesprache RQL, die für die Wissensrepräsentationssprache RDF definiert ist.

2.2.3.2 Formeln über Ontologie-Signaturen

Die Menge $FORM_{OS}$ der Formeln über einer Ontologie-Signatur SIG_{OS} ist rekursiv definiert als:

(1.) Wenn $t_1 \in TERM_T$ und $t_2 \in TERM_T$ Terme zum Konzept $T \in S_{OS}$ sind, dann ist equal(t_1=t_2) eine (atomare) Formel.

(2.) Wenn $t_1 \in TERM_{s.1}$ und $t_2 \in TERM_{s.2}$ jeweils Terme zu den Konzepen $s_1 \in S_{EW}$ und $s_2 \in S_{MW}$ sind, dann ist element_of(t_1,t_2) eine Formel.

(3.) Wenn $t_1,t_2,t_3 \in TERM_{Real}$ jeweils Terme zum Konzept $Real \in S_D$ sind, dann sind add(t_1,t_2,t_3), subt(t_1,t_2,t_3), multip(t_1,t_2,t_3) und div(t_1,t_2,t_3) Formeln.

(4.) Wenn $t_1,t_2 \in TERM_{Real}$ jeweils Terme zum Konzept $Real \in S_D$ sind, dann sind greater(t_1,t_2) und greater_or_equal(t_1,t_2) Formeln.

(5.) Wenn $t_1,t_2,t_3 \in TERM_{String}$ jeweils Terme zum Konzept $String \in S_D$ sind, dann ist concat(t_1,t_2,t_3) eine Formel.

(6.) Wenn $t_1,t_3 \in TERM_{String}$ und $t_2 \in TERM_{Integer}$ jeweils Terme zu den Konzepten $String \in S_D$ bzw. $Integer \in S_D$ sind, dann cut(t_1,t_2,t_3) eine Formel.

(7.) Wenn G eine Formel ist, dann ist auch $\neg G$ eine (zusammengesetzte) Formel.

(8.) Wenn G und H Formeln sind, dann sind auch $G \wedge H$, $G \vee H$, $G \rightarrow H$ und $G \leftrightarrow H$ (zusammengesetzte) Formeln.

(9.) Wenn G eine Formel ist und $x \in VAR$ eine Variable ist, dann sind sowohl $\exists x:G(x)$ als auch $\forall x:G(x)$ (zusammengesetzte) Formeln.

Dadurch, dass jeder Term $t \in TERM_s$ zu einer Sorte $s \in S_{OS}$ immer eine Term $t \in TERM_T$ zur maximalen Ontologie-Sorte T ist, gilt, dass equal(t_1,t_2) eine Formel ist, wenn t_1 und t_2 Terme zu beliebigen Ontologie-Sorten s_1,s_2 sind. Dies ergibt sich aus der (4.) Eigenschaft von Termen über Ontologie-Signaturen, die im vorherigen Abschnitt aufgeführt wurde in Kombination mit der Eigenschaft von Relationssymbolen nur Terme zu solchen Sorten zu akzeptieren, die in der Relationssymboltypisierung an der entsprechenden Stelle aufgeführt wurden.

2.2.4 Ontologien

Ontologien sind formal definiert als:

$$SPEZ_{OS}=(SIG_{OS},VAR_{OS},REG_{OS}).$$

Eine Ontologie ist die Erweiterung einer Ontologie-Signaturen SIG_{OS} um einer Menge VAR_{OS} von Variablen und eine Menge REG_{OS} von Regeln.[1)] Variablen sind notwendig um bestimmte Regeln auszudrücken. Die Regeln einer Ontologie sind eine echte Teilmenge der Menge $FORM_{OS}$ aller Formeln die über einer Ontologie-Signatur formulierbar sind. Somit gilt $REG \subset FORM_{OS}$.

Für die zuvor aufgeführte Signatur O-SIG-Bsp sind z.B. folgende Regeln definierbar:

$\forall x,y$: equal(verheiratet_mit(x),y) $\leftrightarrow$ equal(verheiratet_mit(y),x).
$\forall x,y,z$: equal(arbeitet_fuer(x),y) $\wedge$ equal(tochterunternehmen_von(y),z)
$\rightarrow$ equal(arbeitet_fuer(x),z).
$\forall x,y,z$: equal(arbeitet_fuer(x),y)$\leftrightarrow$equal(hat_Mitarbeiter(y),z) $\wedge$ element_of(x,z).

Mit der ersten Regel wird ausgedrückt, dass das Attributssymbol verheiratet_mit eine inverse Funktion bezeichnet. Die zweite Regel drückt den Sachverhalt aus, dass jeder Mitarbeiter eines Unternehmens, das Tochterunternehmen eines anderen Unternehmens ist, auch ein Mitarbeiter des Mutterunternehmens ist. Mit der dritten Regel wird ausgedrückt, dass, wenn eine Person für ein Unternehmen arbeitet, diese Person in der Belegschaft des jeweiligen Unternehmens enthalten ist.[2)]

Durch die Erweiterung einer Ontologie-Signatur um Variablen und Regeln wird der Interpretationsraum für eine Ontologie-Signatur eingeschränkt. Bisweilen konnten über die Struktur einer Algebra nur mit Hilfe von Termen Aussagen gemacht werden. Da Grundterme für eine Ontologie-Signatur nicht definiert sind, musste hierfür eine Variablenbelegung vorgenommen werden. Durch die Einführung der Relationssymbole RS_{OS} können *wahrheitsfähige* Aussagen über die Struktur einer Algebra gemacht werden. Dennoch sind der Ausdrucksmächtigkeit von Ontologien beim vorgestellten Ansatz Grenzen gesetzt. Sie ist im Vergleich zur Prädikatenlogik gegeben, die es erlaubt, nach Bedarf Relationssymbole zu definieren um über ihre

---

1) Vgl. BENCH-CAPON ET AL. (2003) S. 705; BENCH-CAPON/MALCOLM (1999) S. 254. Da die Menge der Regeln potenziell leer sein kann, ist auch jede Ontologie-Signatur eine Ontologie (Vgl. EHRIG ET AL. (1989) S. 175 („Jede Signatur ist auch eine Signatur-Spezifikation,...“). Vgl. ebenso EHRICH ET AL. (1999) S. 181 („...jede algebraische Signatur auch eine algebraische Spezifikation ist.“). Schließlich werden für eine Spezifikation lediglich weitere Verfahren zur Beschreibung der zu Grunde liegenden Algebra eingesetzt.

2) Mittels der zuvor aufgeführten Pfadausdrücke lassen sich die zweite und die dritte Regel kompakter formulieren. Die zweite Regel könnte semantisch äquivalent wie folgt formuliert werden:

$$\forall x,y: \text{tochterunternehmen\_von}(\text{arbeitet\_fuer}(x))=y \rightarrow \text{arbeitet\_fuer}(x)=y.$$

Die dritte Regel kann lauten:

$$\forall x,y: \text{arbeitet\_fuer}(x)=y \leftrightarrow x \in \text{hat\_mitarbeiter}(y).$$

Durch den Rückgriff auf Pfadausdrücke kann in diesen Fällen auf die konjunktive Verknüpfung von Teilformeln und die dafür notwendige Quantifizierung über Hilfsvariablen verzichtet werden. Ein Großteil von Formeln, die exemplarisch für Ontologien aufgeführt werden, lassen die Substitution ihrer konjunktiv verknüpften Teilformeln durch Pfadausdrücke zu. Allerdings gilt dies nicht für Formeln, in denen die Konjunktion keinem Pfad durch die Ontologie entspricht.

Extension wahrheitsfähige Aussagen machen zu können. Dieses eingeschränkte Ausdrucksvermögen ist allerdings bereits früher angesprochen und vertreten worden.

Durch die Signatur-bezogene Charakterisierung von Ontologien können *Vergleiche* zwischen zwei Ontologien mit Mitteln des traditionellen Signaturkonzepts durchgeführt werden. Beispielsweise lassen sich mittels *Morphismen* Verhältnisse zwischen Signaturen und Algebren untereinander feststellen. So können die Verfahren zur Überprüfung der *Isomorphie* zwischen zwei Signaturen verwendet werden, um formale Instrumente für den Ontologie-Vergleich zu haben. Die Isomorphie von zwei Ontologien deutet auf ihre gegenseitige partielle Ersetzbarkeit aufgrund ihrer strukturellen Gleichheit hin. Die Algebren zu Ontologien sind sich genau dann ähnlich, wenn die Ontologien isomorph sind.[1)]

2.3 Petri-Netze

2.3.1 Allgemeine Petri-Netze

In allgemeinen Petri-Netzen werden die notwendigen Bestandteile jeder folgenden Petri-Netz-Klassen definiert. Daher werden sie auch als Petri-Netze i.e.S. bezeichnet.[2)] Sowohl elementare als auch höhere Petri-Netze weisen alle Bestandteile von allgemeinen Petri-Netzen auf. Die taxonomischen Beziehungen zwischen den Petri-Netz-Klassen entsprechen der Darstellung in Abbildung 9.

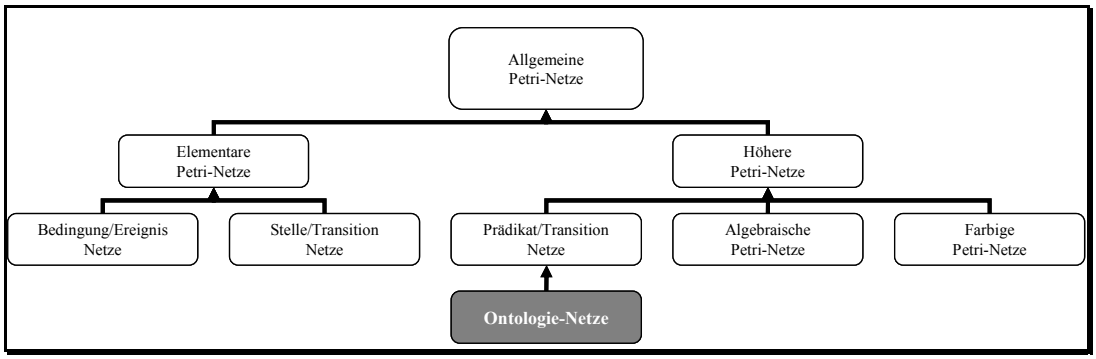

Abbildung 9: Petri-Netz Taxonomie

Die gerichteten Kanten in der Abbildung sind als Generalisierungsbeziehung zu interpretieren. Im vorliegenden Projektbericht wird eine Klasse höherer Petri-Netze vorgestellt, die sich

1) Für eine Vertiefung von Morphismen vgl. EHRIG ET AL. (1999) S. 133 ff.; EHRICH ET AL. (1989) S. 23.

2) Vgl. ZELEWSKI (1995) Bd. 3 S. 25.

als Unterklasse von Prädikat/Transition-Netzen einordnen lässt. Die Ausprägungen der neu vorzustellenden Petri-Netz-Klasse werden als *Ontologie-Netze* bezeichnet.

Ein allgemeines Petri-Netz PN wird definiert durch das Drei-Tupel[1)]

$$PN=(ST,TR,F)$$

Die Komponente eines allgemeinen Petri-Netzes PN sind:

(1.) eine Menge ST von *Stellen* $ST=\{st_m|m\in\mathcal{N}\}$,

(2.) eine Menge TR von *Transitionen* $TR=\{tr_n|n\in\mathcal{N}\}$ und

(3.) eine *Flussrelation* F mit $F\subseteq((ST\times TR)\cup(TR\times ST))$.

Die Stellen- und Transitionenmengen sind *disjunkt* zueinander ($ST\cap TR=\varnothing$). Somit sind Petri-Netze *bipartite Graphen*[2)]. Zudem wird vorausgesetzt, dass sowohl die Stellen- als auch die Transitionenmenge *nicht-leer* sind:

$$S\neq\varnothing,\ T\neq\varnothing$$

und somit

$$S\cup T\neq\varnothing.$$

Schließlich muss das Petri-Netz *verknüpft* sein. Die Verknüpftheit eines Netzes ist dann gegeben, wenn jeder Knoten in mindestens einem Element der Flussrelation vorkommt.

Für jeden Knoten $x\in(ST\cup TR)$ ist sein *Vorbereich* VB(x) gegeben durch

$$VB(x)=\{y|(y,x)\in F\}.$$

Die Menge VB(x) der Knoten, die im Vorbereich eines Knotens vorkommen, wird als dessen *Eingangsknotenmenge* bezeichnet. Wenn in der Eingangsknotenmenge eines Knotens mindestens zwei weitere Knoten enthalten sind, so wird der Knoten als *rückwärtsverzweigt* bezeichnet[3)].

Der *Nachbereich* NB(x) von $x\in(S\cup T)$ ist gegeben durch

$$NB(x)=\{y|(x,y)\in F\}.$$

---

1) Vgl. BAUMGARTEN (1996) S. 50; ROZENBERG/ENGELFRIET (1998) S. 18; REISIG (1991A) S. 17; ZELEWSKI (1995) Bd. 3 S. 25.

2) Bipartite Graphen zeichnen sich dadurch aus, dass die Menge ihrer Knoten so in zwei disjunkte ud exhaustive Teilmengen unterteilt werden kann, dass jede Kante eine Verbindung zwischen zwei Knoten aus den beiden Knotenmengen darstellt (vgl. BAUMGARTEN (1996) S. 42).

3) Vgl. BAUMGARTEN (1996) S. 51.

Die Menge der Knoten, die im Nachbereich eines anderen Knotens vorkommt, wird als dessen *Ausgangsknotenmenge* bezeichnet. Wenn in der Ausgangsknotenmenge eines Knotens mindestens zwei weitere Knoten enthalten sind, so wird der Knoten als *vorwärtsverzweigt* bezeichnet[1)].

Ein Petri-Netz wird als *schlicht* bezeichnet, wenn es im Netz keine zwei Knoten gibt, deren Eingangs- und Ausgangsknotenmengen identisch sind[2)]:

$$\forall x,y \in ST \cup TR: VB(x)=VB(y) \wedge NB(x)=NB(y) \rightarrow x=y$$

Graphisch lassen sich allgemeine Petri-Netze jeweils durch einen Graphen visualisieren, der für jede Stelle einen Kreis und für jede Transition ein Rechteck enthält. Die Elemente der Flussrelation werden durch gerichtete Kanten dargestellt.

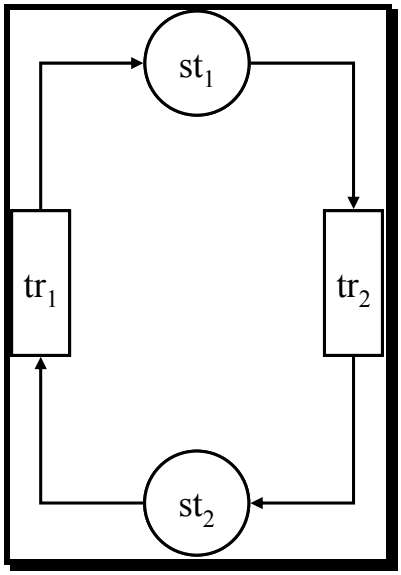

Abbildung 10: Allgemeines Petri-Netz

Abbildung 10 enthält die Visualisierung eines Petri-Netzes

$PN_{PN}=(\{st_1,st_2\},\{tr_1,tr_2\},\{(st_1,tr_2),(st_2,tr_1),(tr_1,st_1),(tr_2,st_2)\})$

mit den Mengen

$ST=\{st_1,st_2\}$,

$TR=\{tr_1,tr_2\}$ und

$F=\{(st_1,tr_2),(st_2,tr_1),(tr_1,st_1),(tr_2,st_2)\}$

1) Vgl. BAUMGARTEN (1996) S. 51.

2) Vgl. BAUMGARTEN (1996) S. 51.

durch einen bipartiten Graphen G(KN,KA) mit der Knotenmenge KN=(ST∪TR) und der gerichteten Kantenmenge KA=F.

2.3.2 Stelle/Transition-Netze

Mit Stelle-Transition-Netzen (ST/TR-Netze) wird die Ausdrucksmächtigkeit des Petri-Netz-Konzeptes um das „Markenspiel“ erweitert.

Ein ST/TR-Netz ist definiert als[1)]:

$$STN=(ST,TR,F,K,W,M_0)$$

mit

- der Stellenmenge $ST=\{st_m|m\in\mathcal{N}\}$,
- der Transitionenmenge $TR=\{tr_n|n\in\mathcal{N}\}$,
- der Flussrelation $F\subseteq((ST\times TR)\cup(TR\times ST))$,
- der Kapazitätsfunktion $K: ST \rightarrow \mathcal{N}\cup\{\propto\}$,
- der Gewichtsfunktion $W: F \rightarrow \mathcal{N}$ und
- der Markierungsfunktion $M_0: ST \rightarrow \mathcal{N}$.

Durch die *Kapazitätsfunktion* K wird festgelegt, wie viele Marken[2)] sich höchstens auf einer Stelle befinden können. Der Zielbereich der Kapazitätsfunktion ist die Vereinigung der Menge $\mathcal{N}$aller natürlichen Zahlen (ohne Null) mit der unendlichen Zahl $\propto$. Es deutet an, dass auch unendlich große Markenkapazitäten zugelassen sein können, so dass sich auf einer Stelle beliebig viele Marken befinden dürfen.

Die *Gewichtungsfunktion* ordnet jeder Kante im ST/TR-Netz eine natürliche Zahl (ohne Null) zu. Der Gewichtungswert für eine Kante gibt an, wie viele Marken von der Stelle durch das Schalten einer adjazenten Transition aus dem Nachbereich der Stelle abgezogen oder durch das Schalten einer adjazenten Transition aus dem Vorbereich der Stelle dort abgelegt werden.

---

1) Vgl. BAUMGARTEN (1996) S. 79; DESEL/REISIG (1998) S. 129; REISIG (1991A) S. 71; ZELEWSKI (1995) Bd. 3 S. 30.

2) Die hier verwendete Darstellung orientiert sich an der üblichen Darstellungen in der Literatur. Streng genommen weist die Markierungsfunktion jeder Stelle eine Zahl zu, die angibt, wie viele miteinander identische *Kopien* von einer *Basismarke* auf einer Stelle liegen. Im Allgemeinen wird der Unterschied zwischen Marken und ihren Kopien aber nicht durchgehalten (vgl. Reisig (1991A) S. 152 für eine Ausnahme).

Wenn keine explizite Gewichtung für eine Kante angegeben ist, wird für eine Kante $(x,y)\in F$ die implizite Gewichtung $K(x,y)=1$ angenommen.

Die *Anfangsmarkierung* ist eine mit den Kapazitäten der Stellen verträgliche Markierung mittels der Anfangsmarkierungsfunktion M_0. M_0 stellt den anfänglichen Zustand dar, von dem ausgegangen wird. Die Menge alle Markierungsfunktionen wird durch die Mengenfamilie MF mit $MF=(M_z)_{z\in ZR}$, $ZR=\{0,1,...,Z\}$ aller Markierungsfunktionen erfasst.

$$M_z: S \rightarrow \mathcal{N}.^{1)}$$

Die Menge ZR wird als *Zustandsraum* bezeichnet. Jeder Index $z\in ZR$[2)] deutet auf einen bestimmten *Zustand* z hin, in dem sich das ST/TR-Netz befindet.[3)] Die Anfangsmarkierung wird entsprechend als M_0 für den Anfangszustand $z=0$ des ST/TR-Netzes angesprochen. Für alle Stellenmarkierungen M_z gilt, dass sie keine Kapazität K(s) einer in M_z markierten Stelle s überschreiten dürfen:

$$\forall s\in ST: M_z(s)\leq K(s).$$

Schaltakte sind Ereignisse, die den Übergang von einer (Referenz-)Markierung des Petri-Netzes zu einer (Folge-)Markierung bewirken.[4)] In der ST/TR-Netz-Terminologie entspricht der Eintritt eines Ereignisses dem *Schalten* einer Transition. Um Schalten zu können, muss eine Transition *aktiviert* sein. Eine Transition $tr\in TR$ ist unter einer Markierung M_z genau dann aktiviert, wenn

(1) $\forall st\in VB(tr):M_z(st)\geq W(st,tr)$

(2) $\forall st\in NB(tr):M_z(st)\leq K(st)-W(tr,st)$.

Durch die erste Aktiviertheitsbedingung wird ausgedrückt, dass alle Stellen, die im Vorbereich der Transition liegen, eine aktuelle Markierung aufweisen müssen, die mindestens so hoch ist wie die Summe der Marken, die von der Stelle bei Schalten der Transition abgezogen werden soll. Die zweite Aktiviertheitsbedingung fordert, dass durch den Schaltakt betroffene

---

1) Alternativ zur funktionalen kann auch eine *vektorale* Darstellung der Stellenmarkierungen vorgenommen werden. In diesem Fall wird durch einen Stellenvektor, dessen Komponenten jeweils Bilder der Markierungsfunktion repräsentieren, die Markierung eines Petri-Netzes dargestellt (vgl. STARKE (1990) S. 26; ZELEWSKI (1995) Bd. 3 S. 32).

2) Der Zustandsindex ist für die Kapazitäts- und Gewichtungsfunktion nicht notwendig, da Kapazitäten und Gewichtungen zustandsinvariante Größen sind.

3) Vgl. ZELEWSKI (1995) Bd. 3 S. 33.

4) Vgl. ZELEWSKI (1995) Bd. 3 S. 45.

Stellen im Nachbereich der Transition durch das Schalten keine Markierung aufweisen können, die die Kapazität der Stelle abzüglich der zufließenden Marken übersteigt.

Wenn eine Transition $tr \in TR$ schaltet, wird der Übergang von einem Netzzustand $1 \in ZR$ in einen Netzzustand $2 \in ZR$ bewirkt. Der neue Netzzustand 2 äußert sich einer neuen Markierung M_2 des Petri-Netzes. Sie ist von der Markierung M_1, die vor dem Schalten der Transition tr vorlag und den Gewichtungen W der Kanten, die mit der schaltenden Transition tr verbunden sind abhängig:

$$M_2(st)= \begin{cases} M_1(st)-W(st,tr) & \text{falls } st \in VB(tr) \setminus NB(tr) \\ M_1(st)+W(tr,st) & \text{falls } st \in NB(tr) \setminus VB(tr) \\ M_1(st)-W(st,tr)+W(tr,st) & \text{falls } st \in NB(tr) \cap VB(tr) \\ M_1(st) & \text{sonst} \end{cases} \quad \text{für alle } st \in ST$$

Der Schaltakt-bewirket Übergang von einer Referenzmarkierung M_1 zu einer Folgemarkierung M_2 wird in der Schreibweise

$$M_1[tr>M_2$$

notiert.

Eine *Schaltfolge* sf ist die Aneinanderreihung von Transitionen $sf=<tr_1...tr_I>$ mit $tr_i \in TR$, $i=1,...,I$ und $I \in \mathcal{N}_+$. Als besonderen Fall werden auch leere Schaltfolgen mit $sf=<>$ zugelassen. Eine Markierung M_2 wird als *Folgemarkierung* einer Referenzmarkierung M_1 bezeichnet, wenn

(1) $sf=<> \wedge M_2=M_1$ oder

(2) $\exists M_3 \in MF: M_1[tr_1...t_{n-1}>M_3 \wedge M_3[tr_n>M_2$.

Wenn eine Markierung M_2 aus einer Markierung M_1 durch eine Schaltfolge sf hervorgeht, wird sie als *erreichbar* aus M_1 bezeichnet.

Graphisch lassen sich ST/TR-Netze unter einer Markierung M_z visualisieren, indem die Anzahl $M_z(st)$ der Marken, die durch die Markierungsfunktion M_z für jede Stelle $st \in ST$ definiert ist, als Anzahl von Punkten auf den Kreisen der Stellen dargestellt werden. In der folgenden Abbildung 11 ist ein beispielhaftes ST/TR-Netz gegeben.

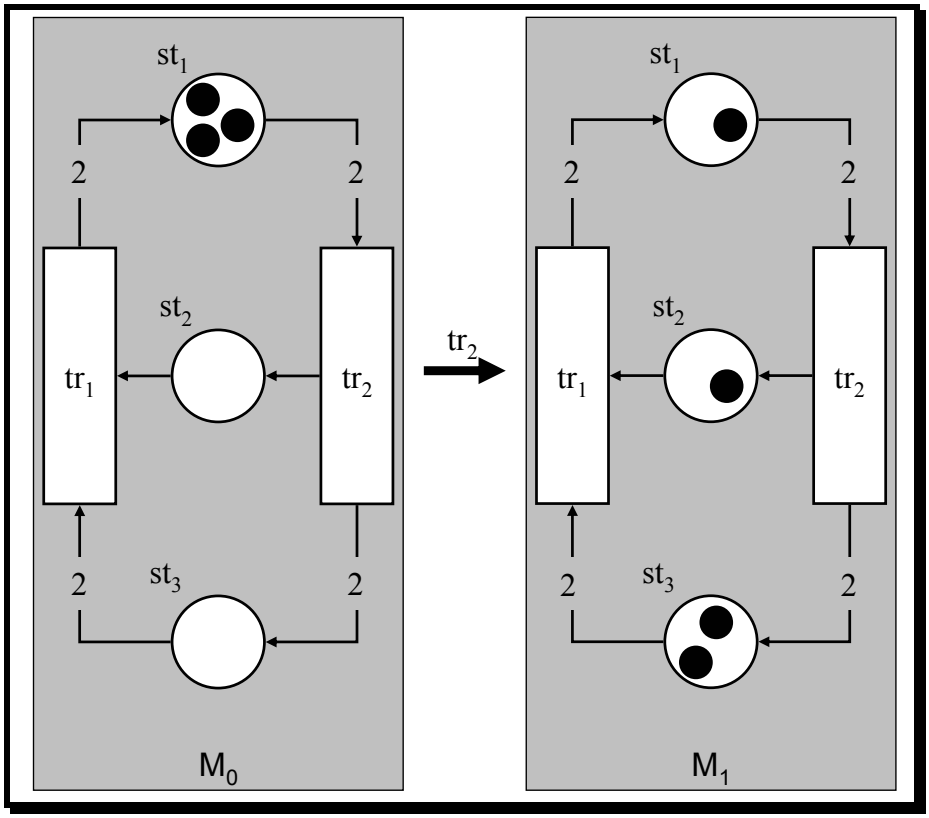

Abbildung 11: Stelle/Transition-Netz und seine Markierungen M_0 und M_1

Die Komponenten des Netzes STN=(S,T,F,K,W,M_0) sind:

ST={st_1,st_2,st_3},

TR={tr_1,tr_2},

F={(st_1,tr_2),(st_2,tr_1),(st_3,tr_1),(tr_1,st_1),(tr_2,st_2),(tr_2,st_2)}

K(st)=∞ für alle st$\in$ST

W(st_1,tr_2)=2, W(st_2,tr_1)=1, W(st_3,tr_1)=2, W(tr_1,st_1)=2, W(tr_2,st_2)=1, W(tr_2,st_2)=1,

M_0(st_1)=3, M_0(st_2)=0, M_0(st_3)=0

Der Übergang M_0[tr_2>M_1 von der Referenzmarkierung M_0 zur Folgemarkierung M_1 mit M_1(st_1)=1, M_1(st_2)=1, M_1(st_3)=1.

Der Übergang M_0[tr_2>M_1 wird in der Abbildung durch den Pfeil zwischen den beiden Graphen angedeutet. Die Transition tr_2 ist im linken Zustand 0 aktiviert, da die einzige Stelle st_1 im Vorbereich von tr_2 mit drei Marken markiert ist und die Flussrelation (st_1,tr_2) ein Gewicht W(st_1,tr_2) von 2 hat. Zudem sind die Stellen st_2,$st_3$$\in$NB($tr_2$) im Nachbereich von tr_2 nicht markiert und die Flussrelation (tr_2,st_3) hat das Gewicht zwei. Schließlich haben beide Stellen st_2,st_3 unbeschränkte Kapazitäten. Somit kann es zum Schaltakt

(st_1,3),(st_2,0),(st_3,0) [tr_2> (st_1,1),(st_2,1)(st_3,2)

kommen. Die Folgemarkierung M_1 ist durch den Graphen auf der rechten Seite der Abbildung 11 angedeutet.

3 Integration von Ontologien und Petri-Netzen

3.1 Überblick über das Vorhaben

Die Petri-Netz-Theorie wird durch den Übergang auf höhere Petri-Netze um die Möglichkeit angereichert, *individualisierte Marken* zu verwenden. Durch diesen Übergang werden die Marken – die bislang in niedrigen Petri-Netzen lediglich atomare Objekte darstellten – mit einer inneren Struktur versehen. Die innere Struktur jeder Marke trägt dann dazu bei, dass alle Marken voneinander individuell und somit voneinander unterscheidbar sind. Mit dieser Individualisierung von Marken ist in erster Linie der Vorteil verbunden, Petri-Netz-basierte Modelle kompakter darzustellen.[1)] Während in ST/TR-Netzen für jedes individuell abzubildende Objekt eine eigene Stelle konzipiert werden muss, können in höheren Petri-Netzen unterschiedliche Objekte auf der selben platziert werden. Die Netz-Dynamik wird dann in Abhängigkeit von den Objekteigenschaften formuliert.

Ausgehend von dem Vorhaben, individuelle Marken zuzulassen, wurden verschiedene Klassen höherer Petri-Netze entwickelt[2)]. Grundlegend sind dabei die algebraischen Petri-Netze, farbige Petri-Netze und Prädikat/Transition-Netze. In algebraischen Petri-Netzen werden die Stellen eines Petri-Netzes mit Sorten einer algebraischen Spezifikation, die Flussrelationen mit Multimengen von Termen über einer Signatur und die Transitionen mit Gleichungen pber einer algebraischen Signatur beschriftet.[3)] In farbigen Petri-Netzen wird jede Stelle im Petri-Netz mit einer „Farbe“, die in der Regel einem Datentyp aus einer zu Grunde gelegten höheren Programmiersprache entspricht beschriftet und die Transitionen mit Formeln beschriftet, die nu für die Datentypen aus den Beschriftungen der adjazenten Stellen definiert sind.[4)]

Für die vorliegende Arbeit werden Prädikat/Transition-Netze (Pr/T-Netze) als formales Gerüst herangezogen. Sie wurden als besonders geeignet für das Integrationskonzept beurteilt, da für Ontologien bereits ein formaler Rahmen gewählt wurde, der sich mit Spezifikationen über

---

1) Vgl. GENRICH (1986) S. 208 ff.; ZELEWSKI (1995) Bd. 5.1 S.8 ff.

2) Für einen umfassenden Vergleich über die verschiedenen Klassen höherer Petri-Netze vgl. GEROGIANNIS ET AL. (1998) S. 139 ff. Für eine generische Beschreibung des Prinzips höherer Petri-Netze vgl. SMITH (1998) S. 181.

3) Vgl. REISIG (1991b) S. 144 ff.

4) Vgl. JENSEN (1996) S. 65 ff.

einer relationalen Signatur deckt. Dieser Rahmen ist reibungslos in eine sortierte Prädikatenlogik überführbar, die als Grundlage für Pr/T-Netze verwendet werden kann.[1)]

Durch die Integration von Ontologien und Pr/T-Netzen wird zum einen erhofft, die fehlende prozedurale Komponente in Ontologien zu überbrücken. Zum anderen wird eine Petri-Netz-Klasse entwickelt, die eine sprachadäquate Modellierung durch ihre Ontologie-Komponente erlaubt. Dieser Zusammenhang wird durch Tabelle 4 verdeutlicht.

	sprachadäquate Modellierung	algebraisch-prädikatenlogische Basis	Prozedurale Semantik
ONTOLOGIEN	✓	✓	
PRÄDIKAT/TRANSITION-NETZE		✓	✓

Tabelle 4: Potenzial von Ontologien und Petri-Netzen zur Integration

Eine weitere Motivation zur Integration von Petri-Netzen und Ontologien ergibt sich aus dem Bedarf nach Instrumenten, die die Modellierung von Prozessen erlauben, die auf verteilte Datenbestände zugreifen. Solche Datenbestände sind typischerweise im Internet vorzufinden. Dabei wird Ontologien im Kontext des *Semantic Web*[2)] ein besonders hohes Leistungspotenzial eingeräumt. Es handelt sich dabei um ein Szenario, das eine Erweiterung des derzeitigen Internets um semantisch ausgezeichnete Informationen beinhalt. Die Auszeichnung der Informationen könnte mittels Begriffssystemen erfolgen, die von Ontologien zur Verfügung gestellt werden.

Allerdings wird in jüngster Zeit auch die Forderung geäußert, Modelle auch mit einer prozeduralen Komponente für das zukünftige Internet bereitzustellen. MARINESCU räumt z.B. Petri-Netzen eine hohe Güte für Prozessmodelle im Semantic Web ein. Allerdings beschränken sich seine Vorschläge auf ST/TR-Netze mit anonymen Marken. Gleichzeitig wird von ihm aber die Bedeutung von Ontologien im Semantic Web hervorgehoben. Insbesondere sei eine gemeinsame Begriffsbasis notwendig, um bei der Verwendung von Prozessmodellen ein gemeinsames Verständnis aller beteiligten Akteure zu sichern.[3)] In einer zweiten Arbeit werden

1) Vgl. HE ET AL. (2004) S. 10 ff.; TACKEN (1994) S. 10 ff.; KORCZYNSKI ET AL. (1990) S. 40 ff.

2) Vgl. CROW/SHADBOLDT (2001) S. 158 ff.; DING ET AL. (2002) S. 210 ff.; ERDMANN/STUDER (2001) S. 324 ff.; HEFLIN (2001) S. 24 ff.; HESSE (2002) S. 478; KIM (2002) S. 52; MAEDCHE/STAAB (2001) S. 72 ff.; MOTTA ET AL. (2000) S. 1072 ff.

3) Vgl. MARINESCU (2002) S. 135 u. 435.

von NARAYANAN/MCILRAITH ST/TR-Netze verwendet, um die Sprache DAML mit einer „operationalen Semantik“ zu versehen.[1)]

Ein viertes Anwendungsszenario hat seinen Ursprung in *Multiagentensystemen* (MAS). Ontologien entstammen traditionell dem Vorhaben, MAS mit einer gemeinsamen Sprachbasis zu versehen.[2)] Diese gemeinsame Sprachbasis wird notwendig, um Kommunikationsbarrieren bei der Koordination zwischen den Agenten desselben MAS zu überwinden. Dabei lassen sich grundsätzlich zwei Strategien unterscheiden. Bei der ersten Strategie wird für alle beteiligten Agenten des MAS *eine* Ontologie vorausgesetzt. Dieser ex-ante-Harmonisierung der Sprachwelten steht eine ex-post-Harmonisierung gegenüber. Bei dieser zweiten Strategie werden Ausdrücke aus den unterschiedlichen agentenspezifischen Ontologien mittels Übersetzungsverfahren aufeinander abgebildet.

Parallel zur Sprachproblematik bei MAS dazu werden Petri-Netze als Instrumente zur Spezifikation der Prozesse in MAS diskutiert.[3)] Ein besonderes Interesse liegt hierbei in der Spezifikation von Prozessen für Agenten, die über betriebliche Grenzen hinaus agieren. Sowohl für traditionelle Kooperationsformen als auch für moderne Supply-Chain-Management-Lösungen und für Konzepte für virtuelle Unternehmen werden MAS diskutiert.[4)] Die „agentengerechte“ Auszeichnung von Informationen mittels Ontologien einerseits und die Spezifikation von Prozessen in MAS andererseits lassen sich miteinander vereinbaren, wenn Ontologie-Netze herangezogen werden.

Das Potenzial höherer Petri-Netze für das Integrationskonzept ist zudem durch ihre nachgewiesene Kompatibilität mit traditionellen Schemata gegeben. Es wurde bisweilen bereits bei der Spezifikation prozessbedingter Operationen auf Datenbanken ausgeschöpft, die auf relationalen-[5)], Nf^2-[6)], SGML-[7)] oder XML-Schemata[8)] basieren. Auch dies ist ein Indiz dafür, dass das Petri-Netz-Konzept für die Integration von Ontologien geeignet ist. Ontologien werden nämlich des Öfteren auch als formale Vorstufe für einige der o.a. Schemata diskutiert.

---

1) Vgl. NARAYANAN/MCILRAITH (2002) S. 81 ff.

2) Vgl. HE/LEUNG (2002) S. 272 f.

3) Vgl. MUSCHOLL (2001) S. 15 ff.; XU ET AL. (2002) S. 194 ff.

4) Vgl. OUZOUNIS (2001) S. 81 ff.

5) Vgl. REISIG (1991) S. 132. ff.

6) Vgl. OBERWEIS (1996) S. 98 ff.

7) Vgl. WEITZ (2000) S. 113 ff.

8) Vgl. LENZ (2003) S. 170 ff.;

3.2 Erweiterung von Ontologien

3.2.1 Erweiterte Ontologie-Signaturen SIG_{EOS}

Die modulare Struktur von Signaturen ermöglicht ihren einseitigen Aufruf. Der Aufruf einer Signatur in einer anderen Signatur ist als Teilmengendeklaration für ihre Sorten- und Operationssymbolmengen zu interpretieren. Die „importierte“ Signatur ist dann eine *Untersignatur* der „importierenden“ Signatur.[1)] Wenn zwei relationale Signaturen $SIG_1=(S_1,OPS_1,RS_1)$ und $SIG_2=(S_2,OPS_2,RS_2)$ mit den Eigenschaften $S_1 \subseteq S_2$, $OPSF_1 \subseteq OPSF_2$ und $RS_1 \subseteq RS_2$ gegeben sind, dann kann die Signatur SIG_1 eine Untersignatur von SIG_2 sein.[2)] Analog wird eine Spezifikation $SPEZ_1=(SIG_1,VAR_1,ANF_1)$ als Unterspezifikation einer anderen Spezifikation $SPEZ_2=(SIG_2,VAR_2,ANF_2)$ bezeichnet, wenn ihre Signatur SIG_1 eine Untersignatur der Signatur SIG_2 von $SPEZ_2$ ist und zusätzlich alle ihre Variablen aus VAR_1 und Anforderungen aus ANF_1 auch in VAR_2 bzw. ANF_2 enthalten sind.

In einer erweiterten Ontologie-Signatur wird neben den Komponenten S_{OS},OSP_{OS} und RS_{OS} , die bereits für jede Ontologie-Signatur festgelegt sind, auch eine weitere Komponente zugelassen. Es handelt sich hierbei um *Relationssymbole mit variablen Extensionen*. Sie werden zur Beschriftung von Stellen in Ontologie-Netzen genutzt. Durch diese Erweiterung von Ontologie-Signaturen um solche Relationssymbole, werden Formeln zugelassen, wie sie bereits über jeder relationalen Signatur in Abschnitt 2.1.4.2 definiert wurden. Dadurch wird das gesamte Ausdruckspotenzial der Prädikatenlogik erschlossen.

Eine erweiterte Ontologie-Signatur ist definiert als:

$$SIG_{EOS}=(SIG_{OS},RS_{EOS})$$

Die Komponenten einer erweiterten Ontologie-Signatur sind:

(1.) eine (konventionelle) Ontologie-Signatur

$SIG_{OS}=(S_{OS},men,\leq_S,=_S,||_S,bez_{lan},def_{lan},OPS_{OS},typ_{OPS},min,max,RS_{OS},inv_{OPS})$ und

(2.) eine Menge RS_{EOS} von Relationssymbolen mit variablen Extensionen.

Die Relationssymbole aus der Menge RS_{EOS} aus eine erweiterten Ontologie-Signatur können in zwei Varianten vorkommen. In der ersten Variante handelt es sich um Relationssymbole

---

1) Vgl. EHRIG ET AL. (1998) S. 124

2) Es gilt dabei zu beachten, dass nicht jedes Tupel (S,OPS), dessen Komponenten in der Form $S \subseteq S_2$ und $OPS \subseteq OPS_2$ gegeben sind, auch eine Signatur darstellt. Voraussetzung hierfür ist, dass alle Elemente der Operationssymbolmenge OPS in ihren Vor- und Nachbereichen nur Sorten enthalten, die Elemente der Sortenmenge S sind.

mit einer Stelligkeit n von genau n=1. Die Extensionen solcher Relationssymbole sind dann solche formalen Objekte aus einer Objektmenge zu einer Sorte, mit der das Relationssymbol typisiert ist. In der zweiten Variante haben Relationssymbole eine Stelligkeit von n≥2. Die Extension eines Relationssymbols $R_i \in R_V$ ist dann die Menge aller n-Tupel $(ob_1,...,ob_n)$ aus formalen Objekten ob_i mit i=1,...,n und $ob_i \in OB_{s.i}$, die als Argumente für das Relationssymbol R_i mit der Typisierung $typ_{RS}(R_i)=s_1...s_n$ aus der zu Grunde liegenden Signatur SIG_{EOS} in Betracht kommen und die die Relation $r_j(ob_1,...,ob_n)$ in der erweiterten Ontologie-Algebra A_{EOS} erfüllen. Relationssymbole mit einem leeren Argumentbereich werden ausgeschlossen.

Die Relationssymbole aus RS_{EOS} werden mit der Typisierungsfunktion

$$typ_{RS}: RS_{EOS} \rightarrow S_{OS}^*$$

typisiert. Die Typisierungsfunktion $typ_{RS\text{-}EOS}$ weist jedem Relationssymbol aus RS_{EOS} eine Verkettung von Konzepten aus der Menge S_{OS} zu. Durch die Typisierung der Relationssymbole aus R_{EOS} werden die Terme festgelegt, die im Argumentbereich der Relationssymbole aus RS_{EOS} vorkommen dürfen.

3.2.2 Erweiterte Ontologie-Algebren A_{EOS}

Eine erweiterte Ontologie-Algebra A_{EOS} ist definiert als:

$$A_{EOS}=(OBF_{OS},OPF_{OS},RF_{OS},RF_{EOS}).$$

Die Komponenten einer erweiterten Ontologie-Algebra A_{EOS} sind:

(1.) eine Familie $OBF_{OS}=(OB_s)_{s\in S}$ von Objektmengen,

(2.) eine Familie $OPF_{OS}=(o_i)_{i=1...I,\ I\in\mathcal{N}}$ von Operationsmengen,

(3.) eine Familie RF_{OS} von Relationsmengen und

(4.) eine Familie RF_{EOS} von Relationsmengen.

Für die Mengenfamilien OBF_{OS}, OPF_{OS} und RF_{OS} gelten die Eigenschaften, die in Abschnitt 2.2.2.1 aufgeführt wurden.

Die Elemente der Mengenfamilie RF_{EOS} interpretieren extensional die Relationssymbole aus der Menge RS_{EOS} die als Komponente erweiterter Ontologie-Signaturen eingeführt wurde. Für jedes Relationssymbol $R_j \in RS_{EOS}$ mit der $typ_{RS\text{-}EOS}(R_j)=s_1...s_n$ enthält die Mengenfamilie RF_{EOS} nämlich eine Relation r_j der Form

$$r_j \subseteq OB_{s.1} \times ... \times OB_{s.n}.$$

Die Relation $r_j \in RF_{EOS}$ ist entsprechend die Extension des Relationssymbols $R_j \in RS_{EOS}$.

3.2.3 Erweiterte Ontologien $SPEZ_{EOS}$

Eine erweiterte Ontologie-Spezifikation ist definiert als:

$$SPEZ_{EOS}=(SIG_{EOS},VAR_{OS},ANF_{OS},ANF_{EOS}).$$

Die Komponenten einer erweiterten Ontologie-Spezifikation sind:

(1.) eine erweitere Ontologie-Signatur SIG_{EOS},

(2.) eine Menge VAR_{OS} von Variablen und

(3.) eine Menge ANF_{OS} von Anforderungen an die Ontologie-Algebra A_{OS} ,

(4.) eine Menge ANF_{EOS} von Anforderungen an die erweiterte Ontologie-Algebra A_{EOS}.

Der erste Unterschied zwischen konventionellen und erweiterten Ontologien liegt darin, dass für erweiterte Ontologien $SPEZ_{EOS}$ wiederum erweiterte Ontologie-Signaturen SIG_{EOS} vorausgesetzt werden, während es sich bei der Signatur-Komponente von konventionellen Ontologien um wiederum konventionelle Ontologie-Signaturen SIG_{OS} handelt. Die Menge der Variablen und die Menge der Anforderungen werden aus der konventionellen Ontologie-Signatur SIG_{OS}, die SIG_{EOS} zu Grunde liegt, übernommen.

Der zweite Unterschied liegt in der Komponente ANF_{EOS}. Es handelt sich hierbei um Anforderungen an die erweiteret Ontologie-Spezifikation. Bei den Elementen der Anfoderungsmenge ANF_{EOS} handelt es sich um Formeln aus der Formelmenge $FORM_{EOS}$, die über der erweiterten Ontologie-Signatur SIG_{EOS} konstruiert werden. Durch sie werden Anfoderungen an erweiterte Ontologie-Algebren ausgedrückt, die mit den Elementen der Anforderungsmenge ANF_{OS} nicht ausgedrückt werden können. Die Elemente der Anforderungsmenge ANF_{EOS} werden verwenden nämlich Relationssymbole RS_{EOS} aus einer erweiterten Ontologie-Signatur SIG_{EOS}.

3.3 Ontologie-Netze

3.3.1 Definition von Ontologie-Netzen

Ein Ontologie-Netz ist definiert als das Tupel:

$$ON=(ST,TR,F,SPEZ_{EOS},BES,A_{EOS},M_0).$$

Die Komponenten eines Ontologie-Netzes sind:

(1.) eine Menge ST von *Stellen* $ST=\{st_m|m\in\mathcal{N}\}$,

(2.) eine Menge TR von *Transitionen* $TR=\{tr_n|n\in\mathcal{N}\}$,

(3.) eine *Flussrelation* F mit $F\subseteq((ST\times TR)\cup(TR\times ST))$,

(4.) eine *erweiterte Ontologie-Spezifikation* $SPEZ_{EOS}$,

(5.) eine Familie von *Beschriftungsfunktionen* $BES=\{BES_{ST},BES_{TR},BES_F\}$,

(6.) eine erweiterte *Ontologie-Algebra* A_{EOS},

(7.) eine *Markierungsfunktion* $M_0= ST \rightarrow \mathcal{MULT}(OB_{BS(ST)})$

Das Drei-Tupel TOP=(ST,TR,F) entspricht der üblichen Netztopologie, die für allgemeine Netze definiert wurde. $ST=\{st_m|m\in\mathcal{N}\}$ ist die Menge der Stellen im Ontologie-Netz. $TR=\{tr_n|n\in\mathcal{N}\}$ ist die Menge der Transitionen. $F\subseteq((ST\times TR)\cup(TR\times ST))$ ist die Menge der Flussrelationen.

Die Knoten und Kanten von Ontologie-Netzen werden mit Komponenten einer erweiterten Ontologie-Spezifikation $SPEZ_{EOS}$ beschriftet.[1)] Die zu Grunde gelegte Spezifikation $SPEZ_{EOS}$ wurde bereits im vorigen Abschnitt als Vier-Tupel

$$SPEZ_{EOS}=(SIG_{EOS},VAR_{OS},ANF_{OS},ANF_{EOS})$$

eingeführt. Es wurde dabei hervorgehoben, dass es sich um eine erweiterte Ontologie-Spezifikation handelt.

BS ist die Menge der *Beschriftungsfunktionen*, die für das Ontologie-Netz definiert sind. Sie ist definiert als Drei-Tupel:

$$BES=(BES_{ST},BES_{TR},BES_F).$$

BES_{ST}: $ST \rightarrow RS_{EOS}$ ist die *Stellen-Beschriftungsfunktion*, die jeder Stelle ein Element aus der Relationssymbolmenge RS_{EOS} zuordnet. Dadurch wird implizit angegeben, welche Art von Marken in einer Stelle zugelassen sind. Die Einschränkung der Menge aller zulässigen Marken ergibt sich später aus einem Zusammenspiel mit der Markierungsfunktion M.

---

1) Somit stehen das algebraische Signaturkonzept und Petri-Netze im hiesigen Konzept in einem *komplementären* Verhältnis. In RITTGEN (1998) S. 97 ff. werden algebraische Spezifiktionen dazu verwendet, die Halbordnungsstruktur von Prozessen zu beschreiben. Dazu werden Operationssymbole eingeführt, die Ordnungsrelationen auf Systemzuständen beschreiben. Der dortige Ansatz setzt daher das algebraische Signaturkonzept in ein *substitutives* Verhältnis mit Petri-Netzen. Diese Vorgehensweise weist eine Verwandtschaft mit *Kripke-Strukturen* auf (vgl. PATIG (2001) S. 62 ff.).

BES_{TR}: TR $\rightarrow$ $FORM_{EOS}$ ist die *Transitions-Beschriftungsfunktion*, die jeder Transition aus der Menge TR ein Element der Formelmenge $FORM_{EOS}$ zuordnet. Die Überlegung dabei ist, Transitionen als Aktivitäten aufzufassen. Die Formelmenge $FORM_{EOS}$ umfasst dabei nur Formeln die mit Hilfe der Relationssymbole aus RS_{EOS} konstruiert werden können

BES_F: F $\rightarrow$ $TERM_{typ(BES(st))}$ ist die *Kanten-Beschriftungsfunktion*, die jedem Element der Flussrelation einen sortenspezifischen Term zuordnet. Die Sorte der Terms ist genau die Sorte, mit das Relationssymbol, mit dem die Stelle aus dem Element der Flussrelation beschriftet ist, typisiert ist. Es kann sich dabei um eine einzelnes Konzept oder auch um eine Verkettung von Konzepten handeln.

$M=(M_z)_{z=0,...,Z}$ mt $Z \in \mathcal{N}$ ist eine Familie von Markierungsfunktionen. M_0= ST $\rightarrow$ $\mathcal{MULT}(OB_{typ(BES(ST))})$ ist die Funktionsvorschrift, die die *Anfangsmarkierung* des Ontologie-Netzes angibt. Während in ST/TR-Netzen die Markierung einer Stelle über eine Funktionsvorschrift definiert ist, die jeder Stelle eine natürliche Zahl zuordnet, wird in Ontologie-Netzen jeder Stelle eine Multimenge über einer sortenspezifischen Objektmenge zugeordnet. Die Sorte der Objektmenge ist genau die Sorte, mit der das Relationssymbol, mit dem die Stelle beschriftet ist, typisiert ist. Wenn die Typisierung $typ_{RS\text{-}EOS}$ des Relationssymbol, mit dem die Stelle beschriftet ist, *ein* Konzept ist, handelt es sich bei der Objektmenge um eine „herkömmliche“ Objektmenge. Wenn die Typisierung eine Verkettung von mindestens zwei Konzepten ist, handelt es sich bei der Objektmenge um ein Kreuzprodukt „herkömmlicher“ Objektmengen.

Eine Transition tr$\in$TR unter einer Markierung M_z und einer Belegung ihrer Variablen durch bel_S ist genau dann aktiviert wenn:

(1.) die Formel G=BES_{TR}(tr), mit der die Transition tr beschriftet ist, zu fausw(F)=*w* ausgewertet wird und

(2.) die Stellen $st_1,...,st_n \in ST$ in den Vorbereichen von tr entsprechend der Beschriftung der Elemente $(st_1,tr),...,(st_n,...,tr) \in F$ der Flussrelation markiert sind:

$$\forall i \in I: tausw_s(BES(st_i,tr)) \leq M_z(st).$$

Es ist dabei zu beachten, dass sowohl bei der Auswertung der Formel F=BES_{TR}(tr), mit der die Transition tr beschriftet ist, als auch bei der Auswertung der Terme $t_1,...,t_n$, mit dem die Kanten $f_1,...,f_n \in F$, die zu tr führen beschriftet sind, die gleichen Variablenbelegungen bel_S vorausgesetzt werden.

Wenn die Transition

3.3.2 Beispiel für Ontologie-Netze

Im Folgenden wird ein Ontologie-Netz entsprechend den vorherigen Abschnitten konstruiert. Bei dem Beispiel handelt es sich um eine Scheduling-Szenario. Aus einer Menge von offenen Aufgaben, werden genau jene entnommen, die von freien Mitarbeitern erledigt werden können. Bei der Zuweiseung von Aufgaben werden zwei Kriterien berücksichtigt. Zum einen muss die zur Erledigung der Aufgabe erforderliche Menge von Kompetenzen beim Mitarbeiter vorhanden sein. Zum anderen muss der Mitarbeiter über mindestens so viel Zeit verfügen, wie sie für die Erledigung der Aufgabe notwendig ist.

$ST = \{st_1, st_2\};$
$TR = \{tr_1\};$
$F = \{(st_1, tr_1), (st_2, tr_1), (tr_1, st_1)\};$

$SPEZ_{EOS}$
S_{OS}
S_{EW}:
{Person, Kompetenz, Aufgabe, Int, Int_pos};
S_{MW}:
{men(Kompetenz)};
OPS_{OS}
OPS_{ST}:
{Hat_Kompetenz Person → men(Kompetenz),
Hat_Alter: Person → Int_pos,
Hat_Zeitbudget: Person → Int,
Erfordert_Kompetenz Aufgabe → men(Kompetenz),
Erfordert_Zeitbudget Aufgabe → Int};
RS_{EOS}:
{Freie_Mitarbeiter: <Person men(Kompetenz) Int>,
Offene_Aufgaben: <Aufgabe men_Kompetenz Int>};
VAR_{OS}:
{{$VAR_{Person} = P_1$},
{$VAR_{Aufgabe} = A_1$},
{$VAR_{INT} = ZB_1, ZB_2, ZB_3$},
{$VAR_{men(Kompetenz)} = K_1, K_2$}};

BES
BES_{ST}
$BES(st_1)$ =Freier_Mitarbeiter,
$BES(st_2)$ =Offene_Aufgaben;
BES_{TR}
$BES(tr_1)$ =(subset_of(K_1,K_2) ∧ lower_or_equal(ZB_2,ZB_1);
BES_F
$BES(st_1, tr_1) = <P_1, K_1, ZB_1>$,
$BES(st_2, tr_1) = <A_1, K_2, ZB_2>$,
$BES(tr_1, st_1) = <P_1, K_1, ZB_3>$,

A_{EOS}
OBF_{OS}
{$OB_{Person} = \{heinz, uwe, werner\}$,
$OB_{Kompetenz} = \{java, sql, c++, englisch\}$,
$OB_{Aufgabe} = \{a_1, a_2, a_3\}$};
OPF_{OS}
{hat_Kompetenz(heinz) ={java},
hat_Kompetenz(uwe) ={java,sql},
hat_Kompetenz(werner) ={sql,c++,englisch}
hat_Alter(heinz) =42,
hat_Alter(uwe) =35,
hat_Alter(werner) =38,
hat_Zeitbudget(heinz) =5,

hat_Zeitbudget(uwe) =6,
hat_Zeitbudget(werner) =3,
erfordert_Kompetenz(a_1) ={java,sql},
erfordert_Kompetenz(a_2) ={java,englisch},
erfordert_Kompetenz(a_3) ={sql,englisch}
erfordert_Zeitbudget(a_1) =4,
erfordert_Zeitbudget(a_2) =2,
erfordert_Zeitbudget(a_3) =5};

RF_{OS}

{freier_Mitarbeiter: OB_{Person} × $OB_{men(Kompetenz)}$ × OB_{Int},
offene_Aufgaben: $OB_{Aufgabe}$ × $OB_{men(Kompetenz)}$ × OB_{Int}};

M_0

$M_0(st_1)$=<heinz,{java},5>+<uwe,{java,sql},6>+<werner,{sql,c++,englisch},3>
$M_0(st_2)$=<a_1,{java,sql},4>+<a_2,{java,englisch},2>+<a_3,{sql,englisch},5>.

Das Ontologie-Netz mit oben angegebenen Komponenten kann mittels eines Graphen wie in Abbildung 12 visualisiert werden.

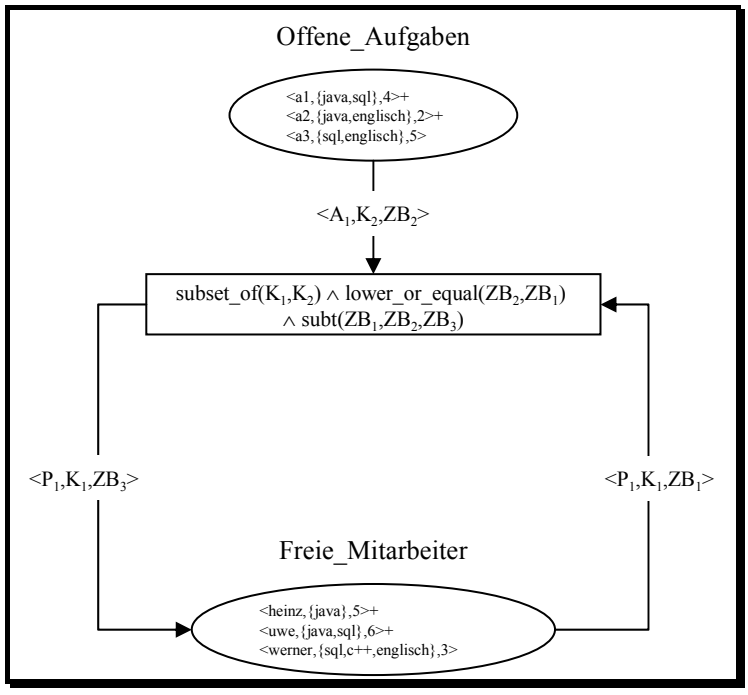

Abbildung 12: Beispielhaftes Ontologie-Netz

In dem Beispiel-Ontologie-Netz ist die einzige Transition tr_1 ist unter der angegebenen Markierung M_0 bei nur einer Variablenbelegungen aktiviert. Sie kann zum einen bei der Variablenbelegung (P_1→uwe; K_1→{java,sql}; ZB_1→6; A_1→a_1; K_2→{java,sql}; ZB_2→4; ZB_3→2) schalten. In diesem Fall würden die Marke zum Mitarbeiter uwe und die Marke zur Aufgabe a_1 von ihren Stellen entnommen werden. Zeitgleich wird eine marke <uwe,{java,sql},2> auf der Stelle st_1 abgelegt.

Die Zuweisung irgendeiner Aufgabe an den Mitarbeiter heinz scheitert an den fehlenden Kompetenzen von heinz. Die Zuweisung von a_1 oder a_2 an werner scheitert ebenso an den fehlenden Kompetenzen von heinz. Die Zuweisung von Aufgabe a_3 an heinz scheitert hingegen an dem zu geringen Zeitbudget von heinz.

3.4 Dynamische Objekterzeugung und -löschung

Im Projektverlauf wurden seitens der Partner des Projektes KOWIEN Anwendungsfälle benannt, die mit den bekannten Methoden zur Konstruktion von Ontologien nicht umgesetzt werden konnten. In erster Linie handelt es sich dabei um Verfahren, die es erlauben, die ontologiegestützte Wissensbasis zur Laufzeit zu modifizieren.[1)] Ein solcher Anwendungsfall beinhaltet z.B. die Erzeugung eines neuen formalen Objektes zu dem Konzept Kompetenzaussage, in dem alle Instanzen aggregiert werden, die Informationen zu den Kompetenzen von Akteuren inklusive der Kompetenzausprägungen beinhalten.

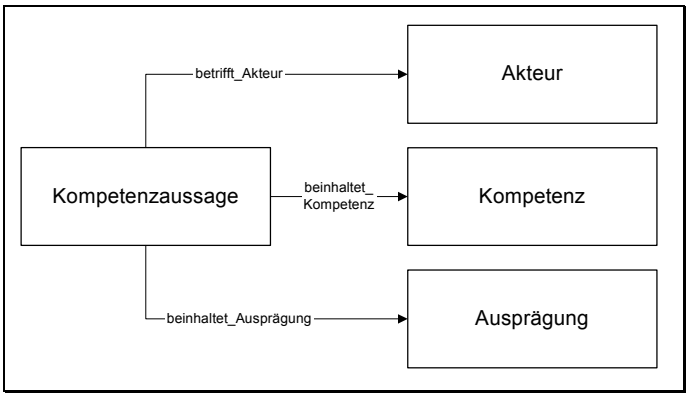

Abbildung 13: Ausschnitt aus der KOWIEN-Ontologie

In Abbildung 13 ist Auszug aus der KOWIEN-Ontologie gegegeben. Die exemplarische Instanziierung des Konzeptes „Kompetenzaussage“ könnte in F-Logic wie folgt lauten:

```
Komptenzaussage_3_Akteur_Mueller[
        betrifft_Akteur->           Mueller;
        beinhaltet_Kompetenz->      Java;
        beinhaltet_Ausprägung->     Experte].
```

Wenn eine Ontologie-Signatur zu Grunde gelegt wird, kann der Ausdruck in einer prädikatenlogischen Notation äquivalent wie folgt formuliert werden:

1) Der Begriff *Modifikation* wird im Folgenden als Oberbegriff für *Löschen*, *Erzeugen* und *Ändern* verwendet.

betrifft_Akteur(Kompetenzaussage_3_Akteur_Mueller) =Mueller ∧
beinhaltet_Kompetenz(Kompetenzaussage_3_Akteur_Mueller) =Java ∧
beinhaltet_Ausprägung(Kompetenzaussage_3_Akteur_Mueller) =Experte

Die oben angegebene Aussage beinhaltet die Information, dass ein Akteur mit Namen Mueller die Kompetenz Java in der Ausprägung Experte hat. Wenn nun aufgrund eines Ereignisses diese Aussage *modifiziert* werden sollte, reichten die bisher vorgestellten Mechanismen hierzu nicht aus. Ein solches Geschehnis kann z.B. die Teilnahme des Mitarbeiters an einer Weiterbildungsmaßnahme sein. Daraufhin müsste eine neue Instanz zum Konzept „Kompetenzaussage" erzeugt werden. Eine solche Regel ist bei einer deklarativen Semantik lediglich bedingt ausdrückbar. Sie übersteigt allerdings die Grenzen der Horn-Logik[1)], die für Ontologie-gestützten Systemen meist gesetzt sind. Es kann nämlich lediglich die Aussage getätigt werden, dass „irgendeine" Kompetenzaussage *existiert*, die Informationen zum Kompetenzprofil des Mitarbeiters beinhaltet. Somit ist es eine Regel, die im Subjugat einen Existenzquantor voraussetzt. Sie könnte beispielhaft in einer prädikatenlogischen Notation wie folgt aussehen:

$\forall$X,Y besuchte_Weiterbildung(X,Y) $\wedge$ hat_Thema (Y,Z) $\rightarrow$
$\exists$K: betrifft_Mitarbeiter(K,X) $\wedge$
beinhaltet_Kompetenz(K,Z) $\wedge$
beinhaltet_Ausprägung(K,Experte)

In Systemen, die auf Horn-Logik beschränkt sind, sind aber Existenzquantoren im Subjugat von allquantifizierten Formeln untersagt. Darüber hinaus ist die „neue" Information, die durch obige Formel generiert wird, von geringer Qualität, da keine nähere Spezifikation erfolgt, um *welche* Kompetenzaussage es sich handelt. Es wird lediglich ausgesagt, dass bei Erfüllung der Bedingungen der Regel mindestens eine Kompetenzaussage K mit spezifizierten Eigenschaften existiert. Der Existenzquantor liesse sich vermeiden, wenn das formale Objekt, das mit K intendiert wird, bezeichnet werden könnte. Da Variablen allerdings aus einer Signatur und formale Objekte aus einer Algebra stammen, herrscht eine Diskrepanz vor, die sich nicht ohne weiteres überbrücken lässt.

Die prozedurale Regel

1) Die Horn-Logik zeichnet sich dadurch aus, dass nur Formel zugelassen werden, die
(1.) in konjunktiver Normalform vorliegen und
(2.) und nur Disjunktionsglieder mit höchstens einem positiven Literal aufweisen.

„Wenn
ein Mitarbeiter an einer Weiterbildungsmaßnahme
zu einem Themenfeld teilgenommen hat und
keine Kompetenzaussage existiert,
die dem Mitarbeiter dieses Themenfeld als Kompetenz zuweist,
dann
erzeuge eine Kompetenzaussage,
die ihm diese Kompetenz zuweist“

kann daher in keiner Ontologie-Regel umgesetzt werden.[1)]

Die prozessabhängige Modifikation von Modellen ist hingegen ein Verfahren, das mittels Petri-Netzen vielfach umgesetzt wurde.[2)] Schließlich entspricht die Anschrit für jede Kante aus der Flussrelationvon Petri-Netzen, die von einer Stelle zu einer Transition führt, einer Löschvorschrift für Tupel aus der Multimenge, mit der die Stelle markiert ist. Analog entspricht jede Flussrelation, die von einer Transition zu einer Stelle führt, einer Einfügevorschrift.

Mit Ontologie-Netzen wird es nunmehr möglich, auch diese Anforderung an ein Ontologiegestütztes System umzusetzen. Es können nunmehr Einfüge- und Löschoperationen für Prädikate definiert werden, deren Extensionen Objektzuständen entsprechen. Aufgrund des bidirektionalen Zusammenhangs zwischen Objektzuständen und den Attributen, die für ein Objekt definiert sind, wird die ursprüngliche Algebra als Modell unzulässig sein. Aus der Klasse aller Algebren, die für die Signatur grundsätzlich zulässig sind, wird nun eine alternative Algebra als Modell interpretiert. Dadurch werden Ontologie-Netze ihrem Anspruch gerecht, Ontologie-gestützte Wissensbasen um dynamische Aspekte zu erweitern.

Um die dynamische Objekterzeugung und -löschung in ein Ontologie-Netz einbinden zu können, ist es notwendig das Netz um „Sammelstellen“ für Objektidentifikatoren zu erweitern. Dabei handelt es sich um einstellige Prädikate, deren Extensionen Konstantensymbole aus der zu Grunde gelegten Zustandsspezifikation sind. Ihre Typisierung ist mit der Ontologie-Sorte T angegeben, so dass jeder Term zu jeder Ontologie-Sorte als Extension angenommen werden

---

1) Es könnte wohl eine Umsetzung erfolgen, wenn die Struktur der Ontologie sich nicht auf reifizierte Kompetenzaussagen stützen würde. Wenn z.B. die Spezifikation Mitarbeiter[hat_Java_Kompetenz=>>Auspraegung] gegeben wäre, so könnte eine Regel der Form
FORALL Mitarbeiter1,Schulung1
Mitarbeiter1[Teilnehmer_an_Schulung->>Schulung1] AND Schulung1[hat_Thema->>Java] →
Mitarbeiter1[hat_Java_Kompetenz->>Experte]
verwendet werden.
Allerdings bleibt auch dabei das Problem erhalten, dass eventuell zuvor vorhandene Relationen zwischen einem Mitarbeiter und einer Kompetenzausprägung „gelöscht“ werden müssten, wenn die neu geschaffene Relation sich nicht mit dieser verträgt.

2) Vgl. AOUMEUR (2001) S. 52 ff.; AOUMEUR/SAAKE (2002) S. 153 ff.; LENZ (2003) S. 175 ff.; OBERWEIS (1996) S. 141 ff.; ZELEWSKI (1995) Bd. 5 S. 38 f. In der letzten Quelle werden Transitionen, die Erzeugungs- oder Löschoperationen eines Modells bewirken, als „Generator-“ bzw. „Absorbertransition“ klassifiziert.

kann. Um eine übersichtlichere Modellierung zu gewährleisten, bietet es sich an, für jede Objekterzeugende bzw. -löschende Transition zwei Stellen im Vorbereich zu definieren, die für verwendete und verfügbare Identifikatoren stehen.

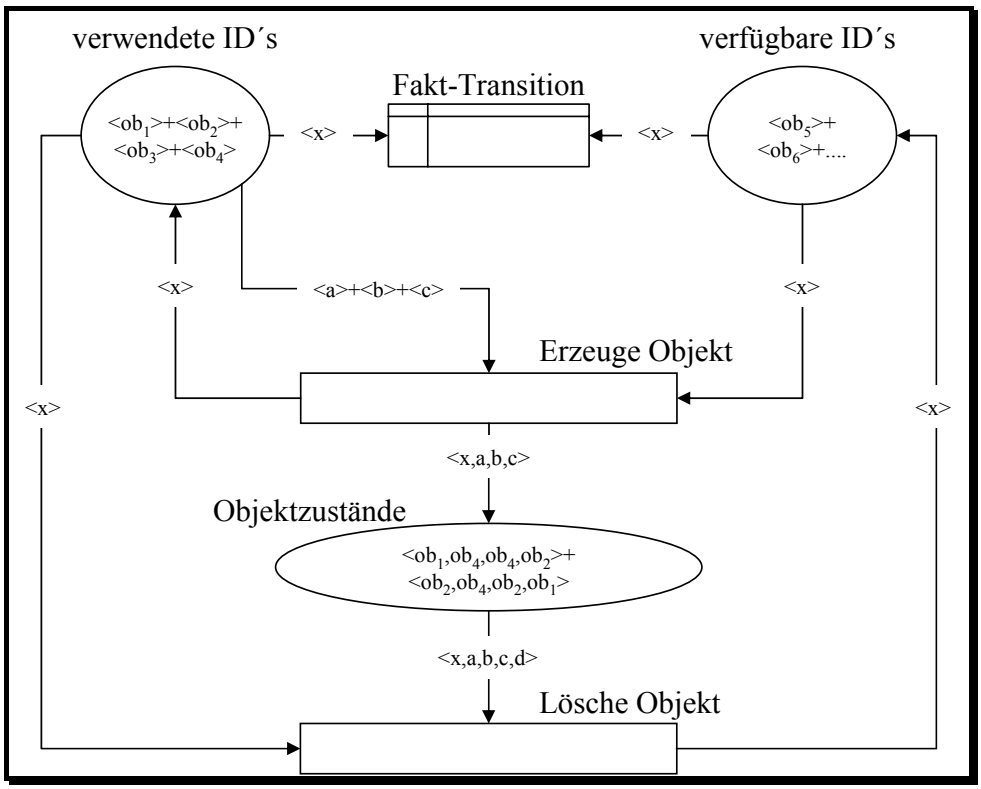

Abbildung 14: Objekterzeugung- und Vernichtung

In Abbildung 14 ist ein exemplarisches Ontologie-Netz aufgezeigt, in dem auch Stellen berücksichtigt sind, die für Identifikatoren vorgesehen sind. Die Stelle „Objektzustände" hat in ihrer aktuellen Markierung bereits eine formale Summe, die die Zustände für die beiden Objekte ob_1 und ob_2 enthält. Die beiden Transitionen „Erzeuge Objekt" und „Lösche Objekt" seien für das Beispiel beide mit tautologischen Formeln beschriftet, so dass ihre Aktiviertheit nur von den Markierungen der Stellen in ihrem Vor- und Nachbereich abhängen.

Als neues Ausdruckmittel ist nun eine *Fakt-Transition*[1)] hinzugekommen. Es handelt sich dabei um einen Transitionstyp, dessen Aktiviertheit auf einen Fehler in der aktuellen Markierung des Netz-Zustandes hinweist. Im obigen Ontologie-Netz wurde eine Fakt-Transition eingefügt, um auszuschließen, dass ein Identifikator sowohl als verfügbar als auch als bereits

1) Vgl. OBERWEIS (1996) S. 159 u. 216 ff.

verwendet angezeigt wird.[1)] Zudem wird bei der objekterzeugenden Transition davon ausgegangen, dass die Objekte, die als Attributwerte angegeben werden sollen (a,b,c), bereits existieren und somit ihre Identifikatoren in der Stelle „verwendete ID´s“ enthalten sind. Dies entspricht auch dem üblichen Vorgehen bei Software-Tools zur Konstruktion von Ontologien. Es lassen sich keine Objekte im Moment der Attributsdefinition von anderen Objekten erzeugen. Jedes Objekt muss zunächst eigenständig erzeugt werden, um nachher als Attributswert für ein anderes Objekt zur Verfügung zu stehen.

4 Kritische Würdigung

Mit dem vorliegenden Projektbericht wurde der erste Versuch unternommen, Ontologien in ein integratives Konzept einzubinden, das über statische Aspekte hinaus auch die Modellierung dynamischer Aspekte erlaubt. Die Dynamik des Konzeptes wird durch Ontologie-Netze gewährleistet.

Durch das integrative Konzept wird es nunmehr möglich, der Ontologie-gestützten Modellierung eine prozedurale Komponente zu verleihen. Durch diese Bereicherung ergeben sich vielfache Anwendungsmöglichkeiten wie z.B. die Modellierung von Workflows im Semantic Web. Hierfür wird es notwendig, Applikationen zu erstellen, die die Erstellung und Wartung von Ontologie-Netzen ermöglichen. Eine prototypische Implementierung scheitert allerdings im KOWIEN-Projekt an zwei Gründen: Zum einen entspricht die hiesige Thematik nicht dem Kerngebiet des Forschungsprojektes KOWIEN. Eine Erweiterung von Ontologien um dynamische Aspekte wurde im Projekt nicht als Ziel definiert. Teilergebnisse der Bemühungen um Ontologie-Netze sind allerdings auch für das KOWIEN-Projekt von Relevanz. So ist die durchgehend formale Argumentation auch für Analysen verwertbar, die sich im Kontext von KOWIEN befinden. Zum anderen werden die Modellierungsprimitive, die für Ontologien hier vorausgesetzt wurden, von keiner Programmiersprache unterstützt. Erst bei einer Reduktion auf die Subsortenrelation kann auf die Programmiersprache OBJ3 zurückgegriffen werden. Sie unterstützt geordnete sortierte Algebren und Termersetzungsregeln, die auf dieser Basis definiert sind.[2)]

---

1) Die gleiche Funktionalität ließe sich für das hiesige Problem auch durch *Inhibitorkanten* umsetzen. In AOUMEUR (2001) S. 52 f. werden solche Inhibitorkanten verwendet, um zu überprüfen, ob der zu Vergebende Identifikator bereits in der Idenitifkator-Sammelstelle enthalten ist. Für eine prototypische Implementierung von Ontologie-Netzen bieten sich eher Fakt-Transitionen an, da Inhibitorkanten lediglich eine Auswirkung auf unzulässige *Schaltakte* haben. Fakt-Transitionen haben hingegen Auswirkungen hinsichtlich unzulässiger *Markierungen*.

2) Vgl. HAMEL/GOGUEN (1994)

Die vorgestellte Konzeption weist allerdings noch „Schwachstellen“ auf, die demnächst angegangen werden. So ist es z.B. bei der derzeitigen Formalisierung schwierig, Prädikatsextensionen anzugeben, wenn für ein Objekt dessen Attributswert nicht definiert ist. Bei einer Erweiterung der Halbordnungsstruktur von Ontologie-Signatur mit einer „entarteten“ Sorte, die Subsorte aller anderen Sorten ist und einer Instanziierung der sortenspezifischen Objektmenge zu genau dieser Sorte mit einem wiederum „entarteten“ Objekt, könnte als Default-Regel formuliert werden, dass bei Nicht-Gegebenheit eines Attributswerts dieses Objekt als Wert angenommen wird. Dies ist problemlos, da das entsprechende Objekt in jeder Objektmenge enthalten wäre, weil die entsprechende Objektmenge Teilmenge jeder anderen Objektmenge wäre. Da die epistemische Qualität einer solchen entarteten Sorte allerdings bisweilen unklar ist, wurde für den vorliegenden Projektbericht hiervon abgesehen.

Darüber hinaus ist es dringend erforderlich für Ontologie-Netze ein Vorgehensmodell zu entwickeln. Ein solches Vorgehensmodell würde wohl als Teilkomponente ein Vorgehensmodell zur Konstruktion von Ontologien voraussetzen. Um jedoch dem ganzheitlichen Anspruch von Ontologie-Netzen gerecht werden zu können, bedarf es eines Vorgehensmodells, dass auch die Erweiterung um dynamische Aspekte berücksichtigt.

Literaturverzeichnis

ALAN (2003)
Alan, Y.: Konstruktion der KOWIEN-Ontologie. Projektbericht 5/2003, Projekt KOWIEN, Institut für Produktion und Industrielles Informationsmanagement, Universität Duisburg-Essen. Essen 2003.

AITKEN/CURTIS (2002)
Aitken, S.; Curtis, J.: A Process Ontology. In: Gómez-Pérez, A.; Benjamins, R. (Hrsg.): Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web, 13th International Conference, EKAW 2002, Siguenza, Spain, October 1-4, 2002, Proceedings. Berlin et al. (Springer) 2002. S. 108-113.

AMBLE (1987)
Amble, T.: Logic Programming and Knowledge Engineering. Wokingham et al. (Addison-Wesley) 1987.

AOUMEUR (2001)
Aoumeur, N.: Specifying and Validating Consistent and Dynamically Evolving Concurrent Information Systems: An Object Petri-net Based Approach. Dissertation, Universität Magdeburg. Magdeburg 2001.

AOUMEUR/SAAKE (2002B)
Aoumeur, N.; Saake, G.: A component-based Petri net model for specifying and validating cooperative information systems. In: Data & Knowledge Engineering, 42. Jg. (2002), H. 2, S. 143-187.

BAUMGARTEN (1996)
Baumgarten, B.: Petri-Netze. 2. Aufl., Mannheim et al. (BI Wissenschaftsverlag) 1996.

BRANDSTÄDT (1994)
Brandstädt, A.: Graphen und Algorithmen. Stuttgart (Teubner) 1994.

BEIERLE ET AL. (1993)
Beierle, C.; Hedtstück, U.; Pletat, U.; Siekmann, J.: An Order-sorted Predicate Logic with Closely Coupled Taxonomic Information. In: Meinke, K.; Tucker, J.V. (Hrsg.): Many-sorted Logic and its Applications. Chichester et al. (John Wiley & Sons) 1993. S. 179-211.

BENCH-CAPON ET AL. (2003)
Bench-Capon, T.; Malcolm, G.; Shave, M.: Semantics for Interoperability: Relating Ontologies and Schemata. In: Maník, V.; Retschitzegger, W.; Stepánková, O. (Hrsg.): Database and Expert Systems Applications. 14th International Conference, DEXA 2003, Prague, Czech Republic, September 1-5, 2003, Proceedings. Berlin et al. (Springer) 2003. S. 703-712.

BENCH-CAPON/MALCOLM (1999)
Bench-Capon, T.; Malcolm, G.: Formalising Ontologies and Their Relations. In: Bench-Capon, T.; Soda, G.; Min Tjoa, A. (Hrsg.): Database and Expert Systems Applications. 10th International Conference, DEXA '99, Florence, Italy, August 30 - September 3, 1999, Proceedings. Berlin et al. (Springer) 1999. S. 250-259.

CARDELLI/WEGNER (1985)
Cardelli, L.; Wegner, P.: On Understanding Types, Data Abtstraction, and Polymorphism. In: Computing Surveys, 17. Jg. (1985), H. 4, S. 471-522.

CARSTENSEN ET AL. (2001)
Carstensen, K.-U.; Ebert, C.; Endriss, C.; Jekat, S.; Klabunde, R.; Langer, H.: Computerlinguistik und Sprachtechnologie. Heidelberg (Spektrum) 2001.

CRANEFIELD/PURVIS (2002)
Cranefield, S.; Purvis, M.: A UML profile and mapping for the generation of ontology-specific content languages. In: The Knowledge Engineering Review, 17. Jg. (2002), H. 1, S. 21-39.

CROW/SHADBOLT (2001)
Crow, L.; Shadbolt, N.: Extracting focused knowledge from the semantic web. In: International Journal of Human-Computer Studies, 54. Jg. (2001), H. 1, S. 155-184.

DESEL/REISIG (1998)
Desel, J.; Reisig, W.: Place/Transition Petri Nets. In: Reisig, W.; Rozenberg, G. (Hrsg.): Lectures on Petri Nets I: Basic Models. Berlin et al. (Springer) 1998. S. 122-173.

DING ET AL. (2002)
Ding, Y.; Fensel, D.; Klein, M.; Omelayenko, B.: The semantic web: yet another hip? In: Data & Knowledge Engineering, 41. Jg. (2002), H. 2-3, S. 205-227.

EBBINGHAUS ET AL. (1992)
Ebbingahaus, H.D.; Flum, J.; Thomas, W.: Einführung in die mathematische Logik. 3. Aufl., Mannheim et al. (BI Wissenschaftsverlag) 1992.

EHRICH ET AL. (1989)
Ehrich, H.-D.; Gogolla, M.; Lipeck, U. W.: Algebraische Spezifikation abstrakter Datentypen. Stuttgart (Teubner) 1989.

EHRIG ET AL. (1999)
Ehrig, H.; Mahr, B.; Cornelius, F.; Große-Rhode, M.; Zeitz, P.: Mathematisch-strukturelle Grundlagen der Informatik. Berlin et al. (Springer) 1999.

ERDMANN/STUDER (2001)
Erdmann, M.; Studer, R.: How to structure and access XML documents with ontologies. In: Data & Knowledge Engineering, 36. Jg. (2001), H. 3, S. 317-335.

ERK/PRIESE (2002)
Erk, K.; Priese, L.: Theoretische Informatik. 2. Aufl., Berlin et al. (Springer) 2002.

FERSTL/SINZ (1998)
Ferstl, O.K.; Sinz, E.J.: Grundlagen der Wirtschaftsinformatik. München - Wien (R. Oldenbourg) 1998.

FREGE (1966)
Frege, G.: Logische Untersuchungen. Göttingen (Vandehoeck&Ruprecht) 1966.

GENRICH (1986)
Genrich, H.J.: Predicate/Transition Nets. In: Brauer, W.; Reisig, W.; Rozenberg, G. (Hrsg.): Petri nets: central models and their properties. Berlin et al. (Springer) 1986. S. 207-247.

GEROGIANNIS ET AL. (1998)
Gerogiannis, V.C.; Kameas, A.D.; Pintelas, P.E.: Comparative Study and categorization of high-level petri-nets. In: The Journal of Systems and Software, 43. Jg. (1998), H. 2, S. 133-160.

GIRAULT/VALK (2003)
Girault, C.; Valk, R.: Petri Nets for Systems Engineering. Berlin et al. (Springer) 2003.

GOGUEN/MESEGUER (1992)
Goguen, J.A.; Meseguer, J.: Order-Sorted Algebra I: Equational Deduction for Multiple Inheritance, Overloading, Exceptions and Partial Operations. In: Theoretical Computer Science, 105. Jg. (1992), H. 2, S. 217-273.

GRUBER (1993)
Gruber, T.: A translation approach to portable ontology specifications. In: Knowledge Acquisition, 5. Jg. (1993), H. 2, S. 199-220.

GUARINO/WELTY (2000)
Guarino, N.; Welty, C.: A Formal Ontology of Properties. In: Dieng, R.; Corby, O. (Hrsg.): Knowledge Acquisition, Modeling and Management. 12th International Conference, EKAW 2000, Juan-les-Pins, France, October 2-6, 2000. Berlin et al. (Springer) 2000. S. 97-112.

GUESSERIAN (1993)
Guesserian, I.: Many-sorted Logic and Algebraic Semantics. In: Meinke, K.; Tucker, J.V. (Hrsg.): Many-sorted Logic and its Applications. Chichester et al. (John Wiley & Sons) 1993. S. 123-134.

HÄSSIG (1979)
Hässig, K.: Graphentheoretische Methoden des Operations Research. Stuttgart (Teubner) 1979.

HAMEL/GOGUEN (1994)
Hamel, L.H.; Goguen, J.A.: Towards a Provably Correct Compiler for OBJ3. In: Hermenegildo, M.V.; Penjam, J. (Hrsg.): Programming Language Implementation and Logic Programming, 6th International Symposium, PLILP'94, Madrid, Spain, September 14-16, 1994, Proceedings. Berlin et al. (Springer) 1994. S. 132-146.

HANSEN/NEUMANN (2001)
Hansen, H.R.; Neumann, G.: Wirtschaftsinformatik I. 8. Aufl., Stuttgart (Lucius & Lucius) 2001.

HARRAS ET AL. (1997)
Harras, G.; Hermann, T.; Grabowski, J.: Aliquid stat pro aliqui - aber wie? In: Grabowski, J.; Harras, G.; Herrmann, T. (Hrsg.): Bedeutung - Konzepte - Bedeutungskonzepte. Opladen (Westdeutscher Verlag) 1997. S. 9-19.

HATZILIYGEROUDIS/REICHGELT (1997)
Hatzilyeroudis, I.; Reichgelt, H.: Handling inheritance in asystem integrating logic in objects. In: Data & Knowledge Engineering, 21. Jg. (1997), H. 3, S. 253-280.

HE ET AL. (2004)
He, X.; Yu, H.; Shi, T.; Ding, J.; Deng, Y.: Formally analyzing software architectural specifications using SAM. In: The Journal of Systems and Software, 71. Jg. (2004), H. 1, S. 11-29.

HE/LEUNG (2002)
He, M.; Leung, H.: Agents in E-Commerce: State of the Art. In: Knowledge and Information Systems, 4. Jg. (2002), H. 4, S. 257-282.

HEFLIN (2001)
Heflin, J.: Towards the Semantic Web - Knowledge Representation in a Dynamic, Distributed Environment. Dissertation, University of Maryland. o.O. 2001.

HEISE (2001)
Heise, A.: Grundlagen strukturierten Spezifizierens mit höheren Netzen. Dissertation, Universität München. München 2001.

HESSE (2002)
Hesse, W.: Ontologie(n). In: Informatik Spektrum, 25. Jg. (2002), H. 6, S. 477-480.

JENSEN (1996)
Jensen, K.: Coloured Petri Nets. Volume 1 - Basic Concepts. 2. Aufl., Berlin et al. (Springer) 1996.

KANEIWA (2001)
Kaneiwa, K.: An Order-sorted Logic with Predicate-Hierarchy, Eventuality and Implicit Negation. Dissertation, School of Information Science, Ishikawa - Japan. Ishikawa 2001.

KARVOUNARAKIS ET AL. (2002)
Karvounarakis, G.; Alexaki, S.; Christophides, V.; Plexousakis, D.; Scholl, M.: RQL: a declarative query language for RDF. In: o. Hrsg. (Hrsg.): WWW 2002. Proceedings of the Eleventh International World Wide Web Conference, WWW2002, Honolulu, Hawaii, USA, 7-11 May 2002. New York (ACM Press) 2002. S. 592 - 603.

KIFER ET AL. (1995)
Kifer, M.; Lausen, G.; Wu. J.: Logical Foundations of Object-Oriented and Frame-Based Languages. In: Journal of the ACM, 42. Jg. (1995), H. 4, S. 741-843.

KIM (2002)
Kim, H.: Predicting how ontologies for the semantic web will evolve. In: Communications of the ACM, 45. Jg. (2002), H. 2, S. 48-54.

KLAPSING (2003)
Klapsing, R.: Beschreibung von Web-basierten Informationssystemen mittels RDF-Metadaten. Dissertation, Universität Essen. Essen 2003.

KOHLHASE (1992)
Kohlhase, M.: Beweissysteme mit Logiken höherer Stufe. In: Bläsius, K.H.; Bürckert, H.-J. (Hrsg.): Deduktionssysteme. 2. Aufl., München - Wien (Oldenbourg) 1992. S. 213-238.

KORCZYNSKI ET AL. (1990)
Korczynski, W.; Düpmeier, C.; Süss, W.: Eine Einführung in die Grundlagen der Theorie der Höheren Petri-Netze. Forschungsberichte Kernforschungs-Zentrum Karlsruhe No. 4636. Karlsruhe 1990.

KREOWSKI (1991)
Kreowski, H.-J.: Logische Grundlagen der Informatik. München - Wien (Oldenbourg) 1991.

LAUSEN/VOSSEN (1996)
Lausen, G.; Vossen, G.: Objekt-orientierte Datenbanken: Modelle und Sprachen. München - Wien (Oldenbourg) 1996.

LEHNER ET AL. (1995)
Lehner, F.; Hildebrand, K.; Maier, R.: Wirtschaftsinformatik. München - Wien (Carl Hanser) 1995.

LOECKX (1996)
Loeckx, J.; Ehrich, H.D.; Wolf, M.: Specification of abstract data types. Chichester et al. (Wiley/Teubner) 1996.

LENZ (2003)
Lenz, K.: Modellierung und Ausführung von E-Business-Prozessen mit XML-Netzen. Dissertation, Universität Frankfurt a.M.. Berlin (VWF) 2003.

MAEDCHE ET AL. (2003)
Maedche, A.; Motik, B.; Stojanovic, L.: Managing multiple and distributed ontologies on the Semantic Web. In: The VLDB Journal, 12. Jg. (2003), H. 4, S. 286-302.

MAIER (1993)
Maier, A.: Einbettung von Konzepthierarchien in ein deduktives Datenbanksystem. Dissertation, Universität Trier. Sankt Augustin (infix) 1993.

MANZANO (1993)
Manzano, M.: Introduction to Many-sorted Logic. In: Meinke, K.; Tucker, J.V. (Hrsg.): Many-sorted Logic and its Applications. Chichester et al. (John Wiley & Sons) 1993. S. 3-86.

MARINESCU (2002)
Marinescu, D.C.: Internet-Based Workflow Management - Toward a Semantic Web. New York (John Wiley & Sons) 2002.

MCGUINESS/VAN HARMELEN (2004)
McGuiness, D.; van Harmelen, F.: OWL Web Ontology Language - Overview. Im Internet unter der URL: http://www.w3.org/TR/2004/REC-owl-features-20040210/. Aufruf 13.05.2004 o.O. 2004.

MONIN (2003)
Monin, J. F.: Understanding Formal Methods. London et al. (Springer) 2003.

MOTTA ET AL. (2000)
Motta, E.; Shum, S.B.; Domingue, J.: Ontology-driven document enrichment: principles, tools and applications. In: International Journal of Human-Computer Studies, 52. Jg. (2000), H. 6, S. 1071-1109.

MUSCHOLL (2001)
Muscholl, K.M.: Interaktion und Koordination in Multiagentensystemen. Dissertation, Universität München. München 2001.

MÜLLER (1999)
Müller, M.: Eine Methode zur automatischen Herleitung oder Widerlegung einfacher Programmeigenschaften durch Generierung neuer Termersetzungsregeln. Dissertation an der Philipps-Universität Marburg. Marburg 1999.

NARAYANAN/MCILRAITH (2002)
Narayanan, S.; McIlraith, S.A.: Simulation, verification and automated composition of web services. In: o. Hrsg. (Hrsg.): WWW 2002. Proceedings of the Eleventh International World Wide Web Conference, WWW2002, Honolulu, Hawaii, USA, 7-11 May 2002. New York (ACM Press) 2002. S. 77-88.

OBERSCHELP (1962)
Oberschelp, A.: Untersuchungen zur mehrsortigen Quantorenlogik. In: Mathematische Annalen, 145. Jg. (1962), S. 297-333.

OBERWEIS (1996)
Oberweis, A.: Modellierung und Ausführung von Workflows mit Petri-Netzen. Stuttgart - Leipzig (Teubner) 1996.

OUZOUNIS (2001)
Ouzounis, E.K.: An Agent-Based Platform for the Management of Dynamic Virtual Enterprises. Dissertation, Technische Universität Berlin. Berlin 2001.

PATIG (2001)
Patig, S.: Flexible Produktionsfeinplanung mit Hilfe von Planungsschritten. Dissertation, Universität Magdeburg. Magdeburg 2001.

RAUH/STICKEL (1997)
Rauh, O.; Stickel, E.: Konzeptuelle Datenmodellierung. Stuttgart (Teubner) 1997.

REISIG (1991A)
Reisig, W.: Petrinetze. 2. Aufl., Berlin et al. (Springer) 1991.

REISIG (1991B)
Reisig, W.: Petri Nets and Algebraic Specifications. In: Theoretical Computer Science, 80. Jg. (1991), H. 1, S. 1-34.

RITTGEN (1998)
Rittgen, P.: Prozeßtheorie der Ablaufplanung. Stuttgart - Leipzig (Teubner) 1998.

ROZENBERG/ENGELFRIET (1998)
Rozenberg, G.; Engelfriet, J.: Elementary Net Systems. In: Reisig, W.; Rozenberg, G. (Hrsg.): Lectures on Petri Nets I: Basic Models. Berlin et al. (Springer) 1998. S. 12-121.

RUPPRECHT (2002)
Rupprecht, C.: Ein Konzept zur projektspezifischen Individualisierung von Prozessmodellen. Dissertation, Universität Karlsruhe. Karlsruhe 2002.

SCHÜTTE (1997)
Schütte, R.: Basispositionen in der Wirtschaftsinformatik - ein gemäßigt-konstruktivistisches Programm. In: Becker, J.; König, W.; Schütte, R.; Wendt, O.; Zelewski, S. (Hrsg.): Wirtschaftsinformatik und Wissenschaftstheorie. Wiesbaden (Gabler) 1997. S. 211-241..

SCHÜTTE (1998)
Schütte, R.: Grundsätze ordnungsmäßiger Referenzmodellierung. Dissertation, Universität Münster. Wiesbaden (Gabler) 1998.

SEIBT (2001)
Seibt, J.: Formal process ontology. In: Welty, C.; Smith, B. (Hrsg.): Formal Ontology in Information Systems. 2nd International Conference on Formal Ontology in Information Systems, FOIS 2001, Ogunquit, Maine, USA, October 17-19, 2001, Proceedings. New York (ACM Press) 2001. S. 333-345.

SIEFKES (1990)
Siefkes, D.: Formalisieren und Beweisen. Braunschweig - Wiesbaden (Vieweg) 1990.

SOWA (1984)
Sowa, J.: Conceptual Structures. Reading et al. (Addison-Wesley) 1984.

SOWA (2000)
Sowa, J.: Knowledge Representation. Pacific Grove et al. (Brooks/Cole) 2000.

SPERSCHNEIDER/HAMMER (1996)
Sperschneider, V.; Hammer, B.: Theoretische Informatik. Berlin et al. (Springer) 1996.

STARKE (1990)
Starke, P.H.: Analyse von Petri-Netz-Modellen. Stuttgart et al. (Teubner) 1990.

SURE (2003)
Sure, Y.: Metodology, Tools & Case Studies for Ontology based Knowledge Management, Dissertation, Universität Karlsruhe. Karlsruhe 2003.

TACKEN (1997)
Tacken, J.: Erweiterte Prädikat/Transitions-Netze - Formale Grundlagen. C-Lab Dokumentation zum Software-Tool SEA. Paderborn 1997.
Im Internet unter der URL:
http://www.c-lab.de/sea/Download/Dokumentation/Postscript/Formal.ps.gz
Aufruf am 12.04.2004.

UPHOFF (1997)
Uphoff, H.: Ein Ansatz zur Integration von Pfadausdrücken in Frame-Logik. Dissertation, Universität Freiburg im Breisgau. Freiburg im Breisgau 1997.

USCHOLD ET AL. (1998)
Uschold, M.; King, M.; Moralee, S.; Zorgios, Y.: The Enterprise Ontology. In: The Knowledge Engineering Review, 13. Jg. (1998), H. 1, S. 31-89.

VOSSEN (1994)
Vossen, G.: Datenmodelle, Datenbanksprachen und Datenbank-Management-Systeme. 2. Aufl., Bonn et al. (Addison-Wesley) 1994.

WEITZ (2000)
Weitz, W.: Integrierte Dokumenten- und Ablaufmodellierung im Electronic Commerce. Dissertation, Universität Karlsruhe. Aachen (Shaker) 2000.

WOLF (2001)
Wolf, S..: Wissenschaftstheoretische und fachmethodische Grundlagen der Konstruktion von generischen Referenzmodellen betrieblicher Systeme. Dissertation, Universität Bamberg. Aachen (Shaker) 2001.

XU ET AL. (2002)
Xu, D.; Volz, R.A.; Ioerger, T.R.; Yen, J.: Modeling and verifying multi-agent behaviors using predicate/transition nets. In: o. Hrsg. (Hrsg.): Proceedings of the 14th international conference on Software engineering and knowledge engineering, July 15-19, 2002, Ischia, Italy. New York (ACM Press) 2002. S. 193-200.

ZELEWSKI (1995)
Zelewski, S.: Petrinetzbasierte Modellierung komplexer Produktionssysteme (Projekt PEMOPS), Bd. 1-10. Arbeitsberichte des Instituts für Produktionswirtschaft und Industrielle Informationswirtschaft (Nr. 5-15), Universität Leipzig. Leipzig 1995.

Institut für Produktion und Industrielles Informationsmanagement Universität Duisburg-Essen / Campus Essen

---

Verzeichnis der KOWIEN-Projektberichte

Nr. 1: ALPARSLAN, A.: Ablauforganisation des Wissensmanagements. Projektbericht 1/2002, Projekt KOWIEN, Institut für Produktion und Industrielles Informationsmanagement, Universität Essen, Essen 2002.

Nr. 2: ALAN, Y.: Methoden zur Akquisition von Wissen über Kompetenzen. Projektbericht 2/2002, Projekt KOWIEN, Institut für Produktion und Industrielles Informationsmanagement, Universität Essen, Essen 2002.

Nr. 3: DITTMANN, L.: Sprachen zur Repräsentation von Wissen - eine untersuchende Darstellung. Projektbericht 3/2002, Projekt KOWIEN, Institut für Produktion und Industrielles Informationsmanagement, Universität Essen, Essen 2002.

Nr. 4: DITTMANN, L.: Zwecke und Sprachen des Wissensmanagements zum Managen von Kompetenzen. Projektbericht 4/2002, Projekt KOWIEN, Institut für Produktion und Industrielles Informationsmanagement, Universität Essen, Essen 2002.

Nr. 5: ALAN, Y.; BÄUMGEN, C.: Anforderungen an den KOWIEN-Prototypen. Projektbericht 5/2002, Projekt KOWIEN, Institut für Produktion und Industrielles Informationsmanagement, Universität Essen, Essen 2002.

Nr. 6: ALPARSLAN, A.: Wissensanalyse und Wissensstrukturierung. Projektbericht 6/2002, Projekt KOWIEN, Institut für Produktion und Industrielles Informationsmanagement, Universität Essen, Essen 2002.

Nr. 7: ALAN, Y.: Evaluation der KOWIEN-Zwischenergebnisse. Projektbericht 7/2002, Projekt KOWIEN, Institut für Produktion und Industrielles Informationsmanagement, Universität Essen, Essen 2002.

Nr. 8: ZUG, S.; KLUMPP, M.; KROL, B.: Wissensmanagement im Gesundheitswesen, Arbeitsbericht Nr. 16, Institut für Produktion und Industrielles Informationsmanagement, Universität Duisburg-Essen (Campus Essen), Essen 2003.

Nr. 9: APKE, S.; DITTMANN, L.: Analyse von Vorgehensmodellen aus dem Software, Knowledge und Ontologies Engineering. Projektbericht 1/2003, Projekt KOWIEN, Institut für Produktion und Industrielles Informationsmanagement, Universität Duisburg-Essen (Campus Essen), Essen 2003.

Nr. 10: ALAN, Y.: Konstruktion der KOWIEN-Ontologie. Projektbericht 2/2003, Projekt KOWIEN, Institut für Produktion und Industrielles Informationsmanagement, Universität Duisburg-Essen (Campus Essen), Essen 2003.

Nr. 11: ALAN, Y.: Ontologiebasierte Wissensräume. Projektbericht 3/2003, Projekt KOWIEN, Institut für Produktion und Industrielles Informationsmanagement, Universität Duisburg-Essen (Campus Essen), Essen 2003.

Nr. 12: APKE, S.; DITTMANN, L.: Generisches Vorgehensmodell KOWIEN Version 1.0. Projektbericht 4/2003, Projekt KOWIEN, Institut für Produktion und Industrielles Informationsmanagement, Universität Duisburg-Essen (Campus Essen), Essen 2003.

Nr. 13: ALAN, Y.: Modifikation der KOWIEN-Ontologie. Projektbericht 5/2003, Projekt KOWIEN, Institut für Produktion und Industrielles Informationsmanagement, Universität Duisburg-Essen (Campus Essen), Essen 2003.

Nr. 14: ALAN, Y.; ALPARSLAN, A.; DITTMANN, L.: Werkzeuge zur Sicherstellung der Adaptibilität des KOWIEN-Vorgehensmodells. Projektbericht 6/2003, Projekt KOWIEN, Institut für Produktion und Industrielles Informationsmanagement, Universität Duisburg-Essen (Campus Essen), Essen 2003.

Nr. 15: ENGELMANN, K.; ALAN, Y.: KOWIEN Fallstudie - Gebert GmbH. Projektbericht 7/2003, Projekt KOWIEN, Institut für Produktion und Industrielles Informationsmanagement, Universität Duisburg-Essen (Campus Essen), Essen 2003.

Nr. 16: DITTMANN, L.: Towards Ontology-based Skills Management. Projektbericht 8/2003, Projekt KOWIEN, Institut für Produktion und Industrielles Informationsmanagement, Universität Duisburg-Essen (Campus Essen), Essen 2003.

Nr. 17: ALPARSLAN, A.: Evaluation des KOWIEN-Vorgehensmodells, Projektbericht 1/2004, Projekt KOWIEN, Institut für Produktion und Industrielles Informationsmanagement, Universität Duisburg-Essen (Campus Essen), Essen 2004.

Nr. 18: APKE, S.; BÄUMGEN, C.; BREMER, A.; DITTMANN, L.: Anforderungsspezifikation für die Entwicklung einer Kompetenz-Ontologie für die Deutsche Montan Technologie GmbH. Projektbericht 2/2004, Projekt KOWIEN, Universität Duisburg-Essen (Campus Essen), Essen 2004.

Nr. 19: HÜGENS, T.: Inferenzregeln des „plausiblen Schließens“ zur Explizierung von implizitem Wissen über Kompetenzen. Projektbericht 3/2004, Projekt KOWIEN, Universität Duisburg-Essen (Campus Essen), Essen 2004.

Nr. 20: ALAN, Y.: Erweiterung von Ontologien um dynamische Aspekte. Projektbericht 4/2004, Projekt KOWIEN, Institut für Produktion und Industrielles Informationsmanagement, Universität Duisburg-Essen (Campus Essen), Essen 2004.

Nr. 21: WEICHELT, T.: Entwicklung einer E-Learning-Anwendung zum kompetenzprofil- und ontologiebasierten Wissensmanagement – Modul 1: Grundlagen. Projektbericht 5/2004, Projekt KOWIEN, Universität Duisburg-Essen (Campus Essen), Essen 2004.