Technische Universität Hamburg-Harburg
Arbeitsbereich Softwaresysteme

Diplomarbeit

zur Erlangung des Grades Diplom-Ingenieur

Eine Architektur zur ontologie-gesteuerten Auswertung von Inhalten in Multimedia-Content-Management-Systemen

Keno Selzer
Matrikel-Nr.: 13075
Studiengang Informatik-Ingenieurwesen
Technische Universität Hamburg-Harburg

Betreuung:

Prof. Dr. rer. nat. Ralf Möller

Arbeitsbereich Softwaresysteme
Technische Universität Hamburg-Harburg

Inhaltsverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

ABox	Assertional Box
ASCII	American Standard Code for Information Interchange
Bd.	Band
bzw.	beziehungsweise
CMS	Content Management Systeme
DAML	Darpa Agent Markup Language
d. h.	das heißt
DIG	Description Logic Implementation Group
engl.	englisch
etc.	et cetera
GSB	Geometrische Szenen Beschreibung
http	Hypertext Transfer Protocol
ID	Identifier (engl. für Identifikator)
KRSS	Knowledge Representation System Specification
MPEG	Moving Picture Experts Group
NAOS	Natural language description of Object movements in Street scenes
NASA	National Aeronautics and Space Administration
OIL	Ontology Inference Layer
OWL	Web Ontology Language
PV	Property-Value (engl. für Eigenschaftswert)
QBE	Query By Example
QBIC	Query By Image Content
Racer	Renamed ABox and Concept Expression Reasoner
RDF	Resource Description Framework

S.	Seite
SM	similarity measure (engl. für Ähnlichkeitsmaß)
TBox	Terminological Box
TCP	Transmisson Control Protocol
vgl.	vergleiche
XML	Extensible Markup Language
z. B.	zum Beispiel

Danksagung

Ich bedanke mich bei Herrn Prof. Dr. rer. nat. Ralf Möller für das interessante Thema, die Betreuung dieser Arbeit sowie für seine hervorragende Unterstützung und Geduld.

Ebenfalls bedanke ich mich für die Unterstützung des Arbeitsbereich Kognitive Systeme der Universität Hamburg, insbesondere bei Lothar Hotz, die mir eine sehr gute Einführung in die Materie boten.

Kapitel 1

1.1 Motivation

Durch die wachsende Anzahl an digitalen Dokumenten und Informationen und die Möglichkeit diese auf immer größeren Speichermedien zu hinterlegen, steigt der Bedarf nach Programmen, die eine Suche über die Dokumente erlauben.
So ist es nicht verblüffend, dass die Hersteller der führenden Suchmaschinen von Internetinhalten Google [1] bzw. Microsoft [2] auch ein Programm zur lokalen Suche auf Desktop Rechnern anbieten [3]. Auch wenn diese Produkte bereits relativ ausgefeilt sind, erlauben sie dennoch nur eine Suche über Texte, wie sie in verschiedenen Dokumententypen (z. B. Microsoft Word-Dokumenten oder PDF) oder E-Mails vorliegen.
Multimediale Informationen wie z. B. Bilder oder Filme können mit diesen Programmen, abgesehen von der Suche nach Dateinamen, nicht erfasst werden. Allerdings nehmen gerade diese Datenbestände durch den Einzug der unkompliziert zu handhabenden Digitalfotografie und den digitalen Videoaufnahmen stetig zu.
Nicht nur im privaten Sektor sind Suchen über multimediale Informationen nötig:
Auch Journalisten, Designer und Werbekaufleute sind in ihrem Berufsleben auf der Suche nach dem passenden Bild für einen Artikel oder eine Anzeige. Ebenso werden im wissenschaftlichen Sektor große Datenbestände angesammelt, die von verschiedenen Forschungseinrichtungen ausgewertet werden. Als Beispiel wäre hier das Erdbeobachtungssystem (engl.: *Earth Observing System*) der *National Aeronautics and Space Administration* (NASA) zu erwähnen, welches eine Reihe von künstlichen Satelliten in der Erdorbit positioniert hat und große Datenmengen an Bildern aufnehmen kann [4]. Weitere Beispiele sind im medizinischen oder militärischen Bereich zu finden.
Multimediale Informationen können in so genannten Multimedia Content Management Systemen verwaltet werden (z. B. CoreMedia CMS [5]). Solche Systeme erlauben das Speichern, Verwalten und das Verteilen an verschiedene Ressourcen. Auch hier stellt sich das Problem, wie ein im System abgespeichertes Bild wieder aufgefunden werden kann. In der Vergangenheit gab es hierzu einige Ansätze:
Der einfachste dieser Ansätze ist die Zuordnung von Schlagwörtern zu den entsprechenden Bildern. Dieses Verfahren nennt man Verschlagwortung.

Es bietet jedoch einige Nachteile:

Die Verschlagwortung erfolgt manuell und ist daher recht zeitintensiv und auf größere Datenmengen kaum anwendbar. Außerdem ist die subjektive visuelle Wahrnehmung der Menschen unterschiedlich, so dass gleiche Bilder von verschiedenen Menschen andere Schlagwörter zugeordnet bekommen und damit das Auffinden der Bilder erschwert wird.

Ein weiteres Verfahren ist das Suchen von Bildern anhand ähnlicher Abbildungen (Query By Example, QBE) [6]. Eines der bekanntesten Beispiele ist IBMs QBIC (Query By Image Content) aus dem Jahre 1995 [7]. Allerdings benötigt man hierfür ein Beispielbild, welches man z. B. als Designer eines Werbeplakates nicht unbedingt zur Verfügung hat, sondern gerade sucht. Wünschenswert wäre also ein System, das durch einfache Eingabe eines Wortes ohne vorherige Indizierung durch Verschlagwortung das Auffinden eines Bildes mit betreffendem Inhalt ermöglicht.

Ein solches System wäre dann auch in der Lage bewegte Bilder (Filme) auszuwerten und könnte somit z. B. auch für die Steuerung eines Roboters mit visueller Erkennung dienen oder für automatische Überwachungskameras.

1.2 Ziel der Arbeit

Ziel der vorliegenden Arbeit ist die Entwicklung einer Architektur, die es erlaubt multimediale Inhalte wie Bilder oder Filme mittels einer Ontologie aus Content-Management-Systemen auszuwerten.

Im Gegensatz zum **SCEN**e **I**nterpretation as **C**onfiguration (kurz: SCENIC), das von dem Arbeitsbereich Kognitive Systeme der Universität Hamburg [21] entwickelt wurde und auf Konfigurierungssystem KONWERK basiert [28], wird in dieser Arbeit das wissenverarbeitende System ***R**enamed **A**Box and **C**oncept **E**xpression **R**easoner* (kurz: RACER [8]), der von der Architektur über Beschreibungslogik angesprochen wird, eingesetzt.

Dabei wird ein grundlegender Verarbeitungsschritt, welcher eine Merkmalsextraktion der Bilder bzw. Filme vornimmt, vorausgesetzt. Eine Zwischenrepräsentation aus diesem Verarbeitungsschritt ist z. B. der MPEG-7 Standard der *Moving Picture Experts Group* (MPEG) [9] oder die geometrische Szenenbeschreibung (GSB engl. geomatrical scene description) des *Natural language description of Object movements in Street scenes* (NAOS, engl. für Beschreibung in natürlicher Sprache von Objektbewegungen in Straßenszenen) [10]. Letzteres wird in dieser Arbeit verwendet und in Kapitel 2.4 näher erläutert.

Die Erstellung der Zwischenrepräsentationen ist bereits hinreichend gelöst und wird kein Teil dieser Arbeit sein. Da bei dieser Extraktion der Zwischenrepräsentation keinerlei übergeordnetes Wissen mit einfließt bzw. je nach Bildquelle eine schlechte Vorlage existiert, kann es hierbei durchaus zu Fehlinterpretationen kommen, die mittels dieser Architektur erkannt und korrigiert werden. Nützlich sind Kontext-Informationen, die gegebenenfalls eventuell aus Nutzereingaben oder ähnlichem resultieren. Zur Korrektur der Fehlinterpretation werden diese Informationen herangezogen, aber auch ohne sie wird eine gute Erkennung von Bildinhalten gegeben sein. Gerade übergeordnete Bildinhalte, die sich aus einer Vielzahl von Objekten zusammensetzen und nicht auf dem Bild als einzelnes Objekt repräsentiert sind, werden ermittelt. Ein Beispiel ist ein Tischgedeck, als solches nicht auf einem Bild als Gegenstand vorhanden. Es setzt sich aus verschiedenen anderen Objekten zusammen wie z. B. einem Teller mit zugehörigem Besteck etc.

Weiterhin erfolgt die Bewertung der Interpretationen von multimedialen Informationen, so dass im Kontext des Content-Management-Systems eine Rangfolge der Suchergebnisse erstellt werden kann.

Als Tests werden unterschiedliche GSB-Daten verwendet, die einerseits zwei komplette

Tischgedecke enthalten, bei denen bereits alle Gegenstände in der sensorischen Extraktion korrekt erkannt und andererseits die betreffenden Gegenstände teilweise falsch klassifiziert wurden. Das Ziel ist es auch beim zweiten Datensatz ein gut bewertetes Gedeck zu finden.

In dieser Arbeit wird ein Verfahren gezeigt, dass aus diese Daten sowohl mit Zusatzinformationen, die das Vorhandensein entsprechender Individuen beinhaltet, als auch ohne sie das gewünschte Resultat ermittelt. Getestet werden auch modifizierte Datensätze, in denen ein oder mehrere Objekte fehlen und somit keine kompletten Ergebnisse ermittelt werden können. Dadurch wird auch das Auffinden von entsprechenden Teilen eines Konzeptes nachgewiesen. Auch höhere Konzepte wie z. B. *Dinner-for-two, Lonely-Dinner* etc. werden definiert.

Weiterhin sind auch die betreffende Suchanfragen, die z. B. ein Gedeck definieren, in zahlreichen Variationen modifiziert worden, um die Auswirkungen von falschen Auslegungen zu erforschen. Eine erfolgreiche Bearbeitung dieser Tests zeigt, dass eine Szenen Interpretation mittels einer Beschreibungslogik durchführbar ist. Eine nähere Erklärung der Tests erfolgt im Anhang dieser Arbeit.

1.3 Gliederung der Arbeit

Die vorliegende Arbeit ist wie folgt strukturiert:
In Kapitel 2 wird als erstes eine kurze Einführung in die Beschreibungslogik und den Racer-Server gegeben. Es folgen eine Erläuterung der Szenen Interpretation mittels der Beschreibungslogik und eine Darstellung des Datenformats der geometrischen Szenenbeschreibung.
Kapitel 3 stellt die entwickelte Architektur vor. Dazu wird als erstes die Architektur und der betreffende GSB-Parser beschrieben. Anschließend wird näher auf die für die Architektur relevanten Suchanfragen eingegangen und letztendlich die entstandene *Hallucination Machine* erläutert.
Abschließend enthält das vierte Kapitel eine Zusammenfassung dieser Arbeit und einen Ausblick auf die zukünftige Erweiterung der Architektur.

Kapitel 2

Einführung

2.1 Kurze Einführung in die Beschreibungslogik

Die Beschreibungslogik (engl. d*escription logic*) ist entwickelt worden, um eine Wissensbasis zu repräsentieren. Zur Anwendung kommt hierfür eine formale Semantik [11].
In der tatsächlichen Welt gibt es Individuen z. B. im englischen Königshaus Elizabeth und Charles und Beziehungen zwischen diesen Individuen z. B. *has_child*, also hat Elizabeth ein Kind namens Charles.
Das Wissen wird hierbei von zwei disjunkten Alphabeten repräsentiert, der Rolle (engl. *role*) und dem Konzept (engl. *concept*).
Die Rolle präsentiert die Beziehungen zwischen den Individuen, also z. B. *has_child(x,y)*, während das Konzept eine Beschreibung des Individuums angibt, also z. B.

$$parent(x) \equiv person(x) \wedge \exists y : (has\_child(x, y) \wedge person(y)) \quad .$$

Umgangssprachlich würde das Beispiel ein Elternteil (engl. *parent*) beschreiben, das eine Person *x* ist und ein Kind *y* besitzt, welches auch eine Person sein muss.
Die Standard-Beschreibungslogik heißt $\mathcal{ALC}$. Hier würde das Konzept *parent* als $parent(x) \equiv person \cap \exists has\_child.parent$ geschrieben werden.
Somit können nun alle für das Beispiel benötigten Konzepte beschrieben werden, um eine Familienstruktur zu definieren.
Eine Ansammlung von Konzepten wird eine TBox (***T**erminological **Box***) genannt.
Mit Konzepten kann man also in der Beschreibungslogik die Hintergrundinformationen darstellen.
In Abbildung 2.1 wird dargestellt, welche Mutterkonzepte es gibt. So wird definiert, dass eine weibliche Person, also eine Frau, zur Mutter wird, wenn sie gleichzeitig Elternteil ist usw.
Weiterhin muss nun in der Beschreibungslogik dargestellt werden können, dass ein Individuum einem Konzept zugehört und eine bestimmte Rolle erfüllt, z. B. dass Elizabeth eine Mutter ist. Dieses geschieht mit Hilfe von erklärenden Axiomen (engl. *assertional axioms*), die das Konzept (engl. *concept assertion*) und die Rollen (engl. *role assertion*) definieren.

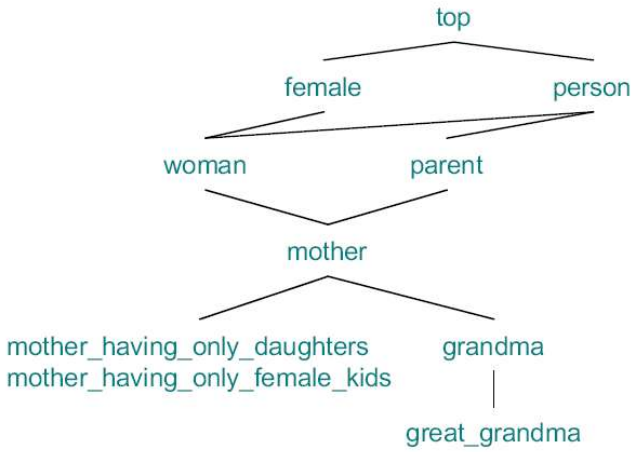

Abbildung 2.1: Beispiel einer TBox (aus [11])

In der *concept assertion* wird einem Individuum ein Konzept zugewiesen, so z. B. Elizabeth dem Mutterkonzept (in $\mathcal{ALC}$: elizabeth:mother).

In der *role assertion* geschieht dieses in Bezug von Individuen auf deren Rolle, so z. B. Elizabeth und Charles der Rolle has_child (in $\mathcal{ALC}$: (elizabeth,charles):has_child).

Eine Ansammlung von *assertion axioms* wird ABox (***A**ssertional **Box***) genannt.

Abbildung 2.2 zeigt ein Beispiel einer ABox, die die Beziehungen im englischen Königshaus darstellt.

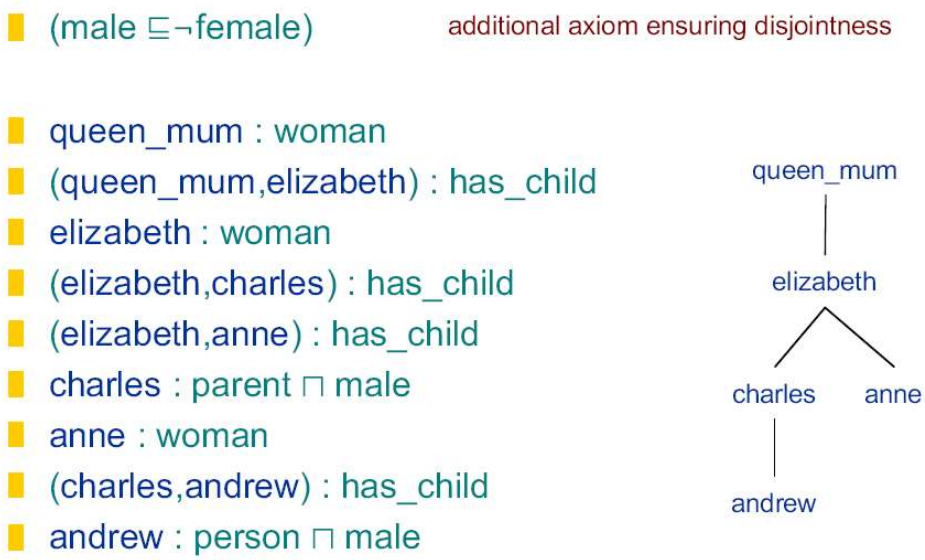

Abbildung 2.2: Beispiel einer ABox (aus [11])

2.2 Einführung in Racer

Racer steht für ***R**enamed **A**Box and **C**oncept **E**xpression **R**easoner* und ist ein wissenverarbeitendes System, dass die Beschreibungslogik $\mathcal{ALCQHI}_{R+}(\mathcal{D}^-)$ implementiert, welches auch unter $\mathcal{SHIQ}(\mathcal{D}_n^-)$ [12] bekannt ist. $\mathcal{SHIQ}$ ist eine Erweiterung von $\mathcal{ALC}$ in Bezug auf die Anzahl der Beschränkungen, der Rollen Hierarchien, der inversen Rollen und der transitiven Rollen [13].
Racer ist somit ein System, das das in der Einleitung beschriebene Problem unter der Voraussetzung, dass die benötigten Hintergrundinformationen vorhanden sind, lösen kann. Es benutzt dabei T- sowie A-Boxen.
Racer ist also eine Umsetzung einer Folgerungsmaschine auf dem semantischen Netz, welches Ontologien entwickeln kann und Suchanfragen über RDF Dokumente stellen kann, wobei RDF (engl. für **R**esource **D**escription **F**ramework) ein universelles Format für Daten im Netz darstellt [14]. Außerdem wird neben RDF noch DAML-OIL [15] und OWL (Web Ontology Language) [16] unterstützt.
Das System implementiert den HTTP-basierenden quasi-standard DIG (Description Logic Implementation Group) für die Kommunikation des Beschreibungslogik-Systems und den Anwendungen, die ein XML-basiertes Protokoll benutzen [17].
Außerdem implementiert Racer die meisten Funktionen, die im älteren KRSS (Knowledge Representation System Specification [18]) spezifiziert wurden [13].
Racer lässt sich neben einem semantischen Netz z. B. noch für Aufgaben aus dem Bereich der medizinischen Informatik, Process-Engineering, Software-Engineering usw. anwenden.
Weitergehende Informationen über das Racer-System werden im Racer Handbuch [13] erläutert.

2.3 Einführung in die Szenen Interpretation mittels Beschreibungslogik

Die visuelle Wahrnehmung beim Menschen und die damit verbundene Interpretation des Gesehenen ist ein äußerst komplexes Verfahren, welches eine Menge Wissen und Erfahrung voraussetzt [19].
Betrachten wir Abbildung 2.3:
Für den Menschen offensichtlich ist hier eine Szene auf einer Straße zu sehen, in der die Müllabfuhr den Müll abholt und die Post ausgetragen wird. Dieses ist aber auch nur offensichtlich, weil Hintergrundwissen über die Kleidung der Müllarbeiter, das Abholen des Mülls mittels speziellen Müllwagen, Aussehen der Mülltonnen etc. vorhanden ist. Ohne dieses Wissen wäre die realistische Interpretation des Bildes sehr schwer bzw. nahezu unmöglich, aber es wäre dennoch eine Deutung möglich.

Abbildung 2.3: Straßenszene (aus [19])

Die Müllarbeiter könnten genauso als Pannenhelfer, die einen Lastkraftwagen reparieren, interpretiert werden, weil der Betrachter nicht über das nötige Wissen über die Müllabfuhr in Deutschland verfügt (z. B. Menschen aus anderen Ländern, in denen die Müllabfuhr andere Kleidung trägt).
Letztendlich können wir auch nicht mit genauer Sicherheit sagen wer mit seiner Interpretation recht behält, weil der Erfahrung nach zwar die Wahrscheinlichkeit hoch liegt, das der Müll abgeholt wird, aber auch uns genau diese Erfahrung täuschen kann. Vielleicht hat der Müllwagen eine Panne und

wird repariert, und wir lassen uns durch unsere Erfahrung zur falschen Interpretation verleiten.
Es gibt also zahlreiche Deutungen eines Bildes und man kann höchstens eine Wahrscheinlichkeit des Interpretationsergebnisses aufgrund seines Wissens und seiner Erfahrung angeben. Ein Raum möglicher Interpretationen wird aufgespannt, der so genannte Halluzinations-Raum, nach dem britischen Wissenschaftler Max Clowes, der 1971 diesen Begriff mit seinem Satz *„Vision is controlled hallucination"* (engl. für „Sehen ist eine kontrollierte Halluzination") prägte.
Eine computer-berechnete Bildinterpretation wird also, genauso wie der Mensch unterbewusst, die wahrscheinlichste Deutung (Halluzination) berechnen. Hierfür wird als erstes eine grundlegende Bildverarbeitung durchgeführt, die diverse Merkmale wie z. B. geometrische Form oder Lage im Bild des gefundenen Objektes extrahiert.
Diese Merkmale werden in einer Zwischenrepräsentation wie z. B. der MPEG-7 Standard der *Moving Picture Experts Group* (MPEG) [9] oder die geometrische Szenenbeschreibung (GSB) des *Natural language description of Object movements in Street scenes* (NAOS) [10] zwischengespeichert. Da die Zwischenrepräsentation weder vollständig noch komplett sein muss, kann durch eine höhere Szenen Interpretation die Verarbeitung unterstützt werden.
Wie in Abbildung 2.4 dargestellt, werden hierfür Bildsequenzen einer Szene oder einzelne Bilder z. B. in GSB-Daten (engl. GSD, geometrical scene description) überführt und mittels höherer Szenen Interpretation, die von der Wissenbasis der Beschreibungslogik und eventuell vorhandener Kontext-Information unterstützt wird, verarbeitet.

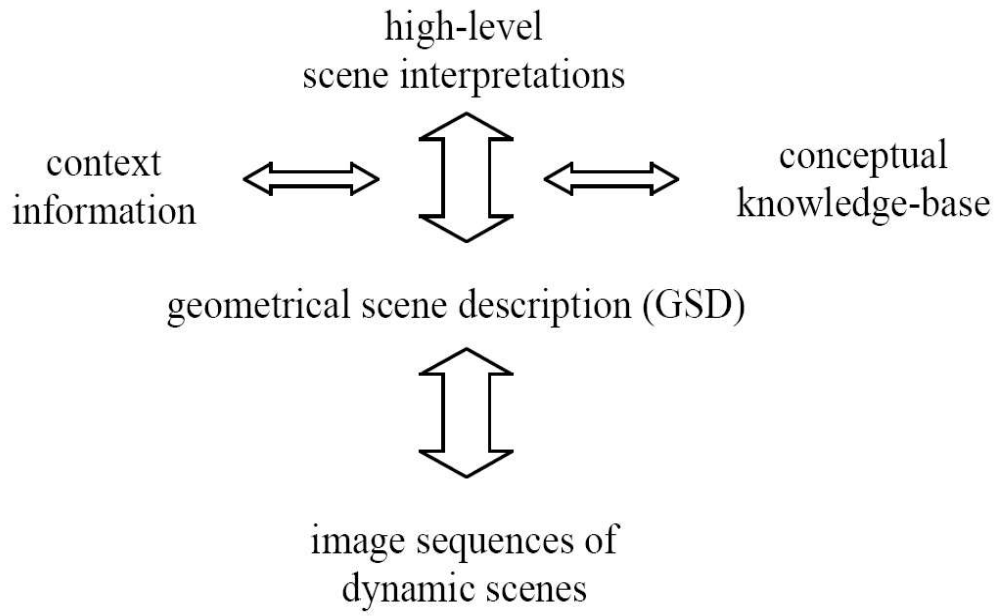

Abbildung 2.4: Architektur für höhere Szenen Interpretation (aus [19])

Dabei kann die Kontext-Information verschiedener Art sein:

Wenn z. B. aus der Szene das Legen eines Tischgedeckes gedeutet wird und es ist durch Zusatzinformation bekannt, dass es am frühen Morgen geschieht, dann kann dadurch auf das Decken eines Frühstückstisches geschlossen werden und das jemand in der nächsten Zeit ein Frühstück zu sich nehmen will [19].

Auch die Informationen über die Ränder des Tisches etc. erleichtern die Bildauswertung. Genauso könnte bereits bekannt sein, z. B. durch Verschlagwortung in einem Multimedia Content Management-System, dass sich auf dem Bild ein Gedeck befindet. In diesem Fall sollte als erstes das Konzept Gedeck gefunden werden, um somit Fehlinterpretationen oder schlechte Deutungen zu vermeiden. Allerdings ist hierbei auch zu beachten, dass die Verschlagwortung, wie bereits weiter oben erwähnt, die Gefahr der Doppeldeutung oder der unterschiedlichen Einschätzung anderer Personen in sich birgt. Ein Beispiel bietet hierfür Abbildung 2.5, die je im Auge des Betrachters ein halb volles oder ein halb leeres Glas darstellt.

Deshalb ist eine abschließende Bewertung des gefundenen Konzeptes unabdingbar.

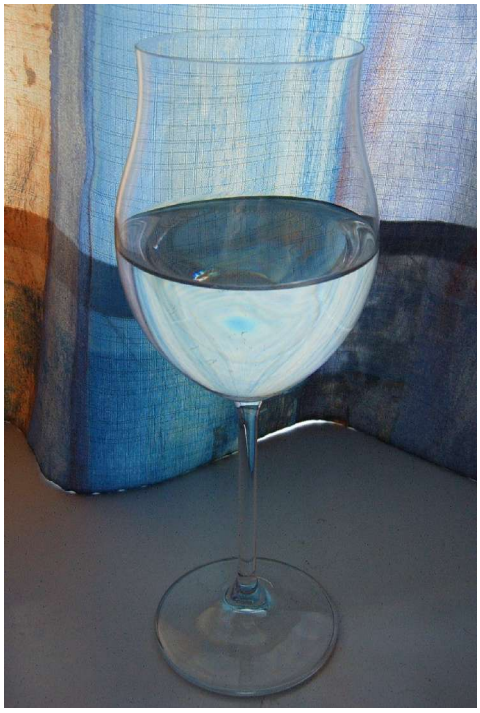

Abbildung 2.5: Abbildung eines Glases

Eine weitere Beschränkung des Halluzinations-Raumes wird, neben der Kontext-Information, durch eine Interpretationsanfrage ermöglicht.

Diese Anfrage, wie z. B. in einem Multimedia Content Management-System eine Suche von Bildern mit gewissen Inhalten, erlaubt ein zielgesteuertes Auswerten des Bildes, die die Möglichkeiten des gegebenen Halluzinations-Raumes erheblich reduzieren. Für diesen Fall wären alternative bzw. weitere Halluzinationen nicht von Interesse und können somit vernachlässigt werden.

Es ist leicht nachvollziehbar, dass theoretisch jedes Konzept in jedem Bild gefunden werden könnte. So könnte z. B. ein Tischgedeck, wie in Abbildung 2.6 dargestellt, auch in einem Foto der Fassade des Hamburger Rathauses gefunden werden, wie in Abbildung 2.7 hervorgehoben. Natürlich macht so eine Interpretation für einen Menschen mit entsprechender Erfahrung wenig Sinn, aber da diese Möglichkeit durchaus in Betracht gezogen werden kann, muss die Bewertungsfunktion hierfür schlechte Werte berechnen. Damit kann ein System das Bild vom Hamburger Rathaus für die Interpretationsanfrage Gedeck gar nicht oder ziemlich weit hinten in der Ergebnisliste der Suchanfrage im Vergleich zu Abbildung 2.6 ausgeben.

Aus diesen beiden Bildern wird auch ersichtlich, dass nicht nur eine geometrische Form, Lage im Bild etc. von Bedeutung sind, sondern auch Beziehungen zwischen den Objekten. Solche Beziehungen werden von Menschen unterbewusst anhand ihres Wissens verarbeitet. Bei einem Gedeck ist es durchaus üblich das die Gabel links und das Messer rechts vom Teller liegt. Dieses hilft die Interpretationen zu finden.

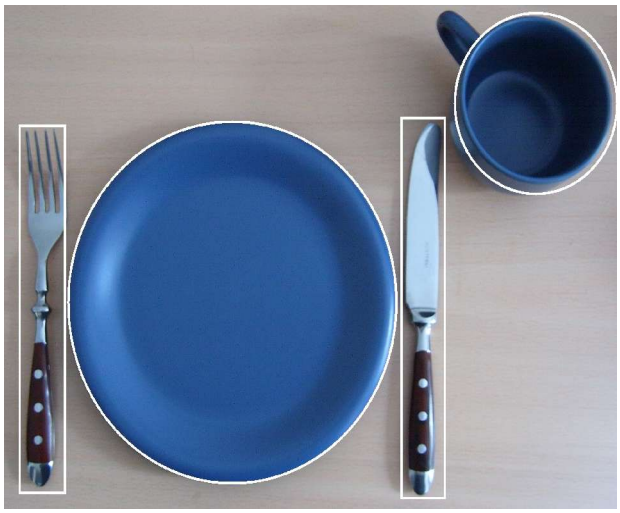

Abbildung 2.6: Tischgedeck mit Markierungen

Abbildung 2.7: Hamburger Rathaus mit Markierungen

Deshalb sollten diese Beziehungen auch in der Beschreibungslogik mittels Rollen verankert werden. Konzeptuale Modelle werden *Aggregates* (engl. für Aggregat) genannt.

Ein *Aggregate* besteht aus einem Namen, einzelnen Teilen, die entweder ein phsyikalisches Objekt oder wiederum ein *Aggregate* sind, Attribute (wie z. B. Zeitstempel etc.), entsprechende Eltern-*Aggregate* und Randbedingungen, die über die Attribute der einzelnen Teile gelten.

Ein Beispiel wäre das Auftreten des *Aggregates* Aufdecken eines Gedeckes (engl. *place-cover)*, wie es als konzeptuales Modell in Abbildung 2.8 dargestellt wird. Dieses *Aggregate* beinhaltet ein *Table Top* (engl. für Tischplatte), drei Transporte von Gegenständen und ein weiteres *Aggregate cover* (engl. für Gedeck).

Weiterhin werden Zeitstempel (engl. *time marks*) für den Anfang bzw. das Ende des *place-cover Aggregate* und Randbedingungen wie z. B. pc-tp3.tp-te ≥ pc-tp2.tp-te die angeben, dass der Transport der Tasse (engl. *cup*) nach dem Transport der Untertasse (engl. *saucer*) enden muss, definiert.

name:	place-cover
parents:	:is-a agent-activity
parts:	pc-tt :is-a table-top
	pc-tp1 :is-a transport with (tp-obj :is-a plate)
	pc-tp2 :is-a transport with (tp-obj :is-a saucer)
	pc-tp3 :is-a transport with (tp-obj :is-a cup)
	pc-cv :is-a cover
time marks:	pc-tb, pc-te :is-a timepoint
constraints:	pc-tp1.tp-ob = pc-cv.cv-pl
	pc-tp2.tp-ob = pc-cv.cv-sc
	pc-tp3.tp-ob = pc-cv.cv-cp
	...
	pc-tp3.tp-te ≥ pc-tp2.tp-te
	pc-tb ≤ pc-tp3.tb
	pc-te ≥ pc-cv.cv-tb

Abbildung 2.8: Konzeptuales Modell einer place-cover scene (aus [19])

```
(equivalent place-cover
        (and agent-activity
                (exactly 1 pc-tp1 (and transport (some tp-obj plate))
                (exactly 1 pc-tp2 (and transport (some tp-obj saucer))
                (exactly 1 pc-tp3 (and transport (some tp-obj cup))
                (<= pc-tp2 ○ tp-end pc-tp3 ○ tp-end)
                (= pc-beg (minim pc-tp1 ○ tp-beg pc-tp2 ○ tp-beg pc-tp3 ○ tp-beg))
                (= pc-end (maxim pc-tp1 ○ tp-end pc-tp2 ○ tp-end pc-tp3 ○ tp-end))
                (<= (- pc-end pc-beg) max-duration))))
```

Abbildung 2.9: Beschreibungslogik-Konzept place-cover mit zeitlichen Beschränkungen aus [19]

Wie in [19] beschrieben, werden die *Aggregates* nun in Beschreibungslogik Konzepte umgeschrieben:

Die Bezeichner unter *parts* (engl. für Bestandteile) wie z. B. pc-tp1 werden in Rollen Namen, die entsprechenden Konzept-Ausdrücke für die Werte werden in Rollenwerte-Restriktionen und das ganze wird zur einer Vereinigung aus Rollen-Restriktionen verändert. Ein Beispiel bietet hierfür Abbildung 2.9, die *place-cover* mit zeitlichen Beschränkungen zeigt.

Zusammenfassend wird in Abbildung 2.10 ein Schema für diese Umwandlung dargestellt. Wie oben beschrieben wird das Konzept aus den einzelnen Bestandteilen zusammengesetzt. Die Elternkonzepte (engl. *parent-concept*) werden aus den *parents* (engl. für Eltern) Zeilen entnommen und der Name ergibt sich aus der *name* (engl. für Name) Zeile.

Im folgendem werden zum besseren Verständnis die höheren Berschreibungslogik Konzepte, die keine physikalisch vorhandenen Gegenstände repräsentieren, sondern auf konzeptualen Modellen basieren, als *Aggregates* bezeichnet.

```
(equivalent <concept-name>
        (and <parent-concept1> ... <parent-conceptN>
                (<number-restriction1> <role.name1> <part-concept1>)
                        ...
                (<number-restrictionK> <role.nameK> <part-conceptK>)
                <constraints between parts>))
```

Abbildung 2.10: Schema für die Erstellung eines Beschreibungslogik-Konzeptes

In [19] bzw. [20] wird eine Möglichkeit aufgezeigt sich durch den Halluzinations-Raum zu bewegen und damit eine Deutung des Bildes zu erlangen.
Hierfür werden vier grundlegende Befehle verwendet:

- *Aggregate Instantiation* (engl. für Aggregat Instantiierung)
- *Instance Expansion* (engl. für Instanz Erweiterung)
- *Instance Specialisation* (engl. für Instanz Spezialisierung)
- *Instance Merging* (engl. für Instanz Zusammenfassung)

Der Befehl *Aggregate Instantiation* bewirkt das Erstellen eines *Aggregates* aus den einzelnen Teilen. Hierfür werden die entsprechenden Teile (nicht unbedingt notwendigerweise alle) unter entsprechenden Randbedingungen gesucht und zu einem *Aggregate* instantiiert.
Im Gegensatz zu dieser von unten nach oben aufbauenden Suche steht der *Instance Expansion* Befehl, der zu einem bereits instantiierten *Aggregate* entsprechende Teile erstellt. Hierbei können z. B. durch *Aggregate Instantiation* noch nicht gefundene Teile nachträglich erstellt werden, also auch hier muss nicht die Erweiterung aller Teile einer Instanz erfolgen. Ein typischer Grund für *Instance Expansion* ist der Bedarf einer Verbindung zwischen einem höheren *Aggregates* und einem visuellen Objekt [19].
Der Schritt *Instance Specialisation* stellt eine Spezialisierung eines bereits erkannten Objektes her. Ein Beispiel wäre die Spezialisierung eines Konzeptes Teller in ein Konzept Teller-aus-Meissner-Porzellan durch vielleicht nicht erkannten oder fehlenden Merkmalen oder Attributen.
Der vierte Befehl *Instance Merging* führt zwei gleiche *Aggregate* zusammen. Es kann durchaus vorkommen, dass bei der Erarbeitung einer Deutung eines Bildes mehrere Aggregate genau das gleiche beschreiben. Tritt dieses auf, dann müssen beide zusammengeführt werden.
Ein Beispiel hierfür ist ein Element wie z. B. Gedeck welches durch Kontext-Information entstanden ist und nun mit dem eigentlich schon erkannten Gedeck zusammengefasst wird.

2.4 Datenformat der Geometrischen Szenen Beschreibung

Wie erwähnt ist die geometrische Szenenbeschreibung (GSB) des NAOS [10] eine Zwischenrepräsentation, die aus der sensorischen Verarbeitung einer bewegten Szene entsteht. Eine weitere Zwischenrepräsentation wäre der MPEG-7 Standard der *Moving Picture Experts Group* (MPEG) [9].

In dieser Arbeit wurde die GSB als Grundlage verwendet, da mir durch die bisherige Forschungsarbeit des Arbeitsbereiches Kognitive Systeme der Universität Hamburg [21] hervorragendes Datenmaterial zur Verfügung gestellt wurde.

Die Merkmalsextraktion der Bilder bzw. bewegten Szenen ist hinreichend in anderen Arbeiten behandelt worden und wird kein Thema dieser Arbeit darstellen.

Die geometrische Szenenbeschreibung besteht aus einem Strom von ASCII-Zeichen und bietet dadurch Systemunabhängigkeit [22]. Dabei ist die GSB als eine Folge von beliebig vielen Bildbeschreibungen aufgebaut. Dadurch ist es auch möglich ein einzelnes Bild, welches im Prinzip als erster Teil einer bewegter Szene angesehen werden könnte, im GSB-Format darzustellen. Eine Bildbeschreibung in der GSB ist jeweils durch ein Klammerpaar umschlossen und beginnt neben dem Schlüsselwort „FR“ für *Frame* (engl. für Vollbild) mit einer fortlaufenden Nummer des entsprechenden *Frames.*

In Abbildung 2.11 ist ein Beispiel zweier Bildbeschreibungen einer Filmsequenz dargestellt und Abbildung 2.12 zeigt das Format der GSB in einer erweiterten Backus-Naur-Form und dient zum besseren Verständnis des folgenden:

In dieser Klammerung sind die einzelnen gefundenen Objekte, wiederum geklammert, beinhaltet.

```
(FR 38 (ID 1 (PV TYPE SAUCER) (PV CENTER (423 265)) (PV BOX (393 235 455 298)) (PV
SM(14 5 11 0 2 98 )) (PV MOVE 1)) (ID 2 (PV TYPE PLATE) (PV CENTER (107 353)) (PV
BOX (60 312 155 388)) (PV SM(4 0 4 70 0 15 )) (PV MOVE 1)))

(FR 40 (ID 1 (PV TYPE SAUCER) (PV CENTER (425 256)) (PV BOX (394 226 456 289)) (PV
SM(14 5 12 0 2 98 )) (PV MOVE 1)) (ID 2 (PV TYPE PLATE) (PV CENTER (107 349)) (PV
BOX (60 307 154 387)) (PV SM(3 0 4 78 0 8 )) (PV MOVE 1)))
```

Abbildung 2.11: Ausschnitt eines Beispieles eines GSB-Datenbestandes

GSB	::=	(FR Frame-Nr. {Object})
Object	::=	(ID Object-Nr. {Property})
Property	::=	(PV Property-Name Property-Value)
Property-Name	::=	TYPE | CENTER | BOX | SM | MOVE

Abbildung 2.12: Format der GSB in erweiterter Backus-Naur-Form (aus [21])

Objekte beginnen mit dem Kürzel „ID“ für *Identifier* (engl. für Identifikator) gefolgt von einer Nummer, die das Objekt auch in anderen *Frames* identifiziert. Dabei besitzen diese, ebenfalls durch Klammerung umschlossenen, verschiedene Eigenschaften gekennzeichnet durch „PV“ für *Property-Value* (engl. für Eigenschaftswert).
Es gibt verschiedene Typen von Eigenschaften, die nach dem Kürzel „PV“ angegeben werden:

- *Type* (engl. für Typ)
- *Center* (engl. für Mitte)
- *Box* (engl. für Kasten)
- *SM (*Abkürzung für *similarity measure* engl. für Ähnlichkeitsmaß)
- *Move* (engl. für Bewegung)

Theoretisch könnten noch weitere *Property-Value*-Paare wie z. B. die Farbe des Objektes oder die Orientierung bzw. Ausrichtung verwendet werden. Allerdings werden sie nicht in den vorliegenden Daten benutzt und sind für die verwendeten Beispiele nicht von Relevanz.
Die Eigenschaft *Type* gibt den erkannten Typen des Objektes an. Also z. B. *Saucer* (engl. für Untertasse) für das Objekt mit der „ID“ 1 im Beispiel in Abbildung 2.11. Diese Typangabe ist eine erste Klassifizierung der Merkmalsextraktion und kann sich im Laufe einer bewegten Szene durch z. B. bessere Sicht auf das Objekt o.ä. ändern.
Center gibt den Mittelpunkt des Objektes an und dient zusammen mit der *Box*, die die Koordinaten zweier Eckpunkte eines virtuelles Rechteck wiederspiegelt, welches den Gegenstand umschließt, der Orientierung im Bild und Abschätzung der Größe.
Die Eigenschaft *similarity measure* beinhaltet mehrere Werte, die umklammert sind, und der Wahrscheinlichkeit des Objektes eines gewissen Types entsprechen.
So ist im obigen Beispiel in der *Frame*-Nummer 40 der letzte Wert des ersten Gegenstandes mit 98 angegeben und drückt somit die 98% Wahrscheinlichkeit aus, dass dieses Objekt eine Untertasse

(engl. *saucer*) ist. Die höchste Bewertung entspricht immer den vorklassifizierten Typen.
Der Wert der Eigenschaft *Move* ist ein binärer Wert, der lediglich anzeigt, ob das Objekt in dem betreffenden Bild bewegt wird oder sich in einer Ruheposition befindet.

Kapitel 3

Vorstellung der Architektur

3.1 Beschreibung der Architektur

Im Rahmen dieser Arbeit ist eine Architektur zur ontologie-gesteuerten Auswertung von Inhalten in Multimedia-Content-Management-Systemen entstanden.
Eine schematische Darstellung dieser Architektur stellt Abbildung 3.1 dar.
Die entwickelte „Hallucination Machine“ kommuniziert mit dem als Basis dienenden, bereits vorgestellten Racer-Server über eine TCP-Schnittstelle. Der Name „Hallucination Machine“ ergibt sich durch die Eigenschaft des Aufspannens und Duchsuchens des in Kapitel 2.3 erwähnten Halluzinations-Raumes. Das Programm wurde in Java [23] entwickelt, um eine weitgehende Plattformunabhängigkeit zu bieten.
Generell werden dem Programm, wie bereits in Kapitel 2.3 angedacht, Kontext-Informationen und Interpretationsanfragen übergeben, unterstützt von Suchanfragen, deren nähere Bedeutung noch in Kapitel 3.3 erläutert wird.
Für diese Arbeit wurde die Racer-Version Racer Pro 1.8.2 verwendet, die eine Wissensbasis mittels einer TBox übertragen bekommt.
Die vorliegenden GSB-Daten werden mit einem ebenfalls erstellten GSB-Parser, der in Kapitel 3.2 beschrieben wird, in eine entsprechende $ABox_{(einfach)}$ überführt, die ebenfalls Racer übergeben wird. Nach der Verarbeitung in der „Hallucination Machine“ bzw. Racer-Server wird diese $ABox_{(einfach)}$ in die $ABox_{(Ergebnis)}$ überführt, die alle gefundenen Konzepte (z. B. Tischgedeck) und alle gefundenen bzw. durch höheres Wissen korrigierte Individuen enthält. Die $ABox_{(Ergebnis)}$ kann entweder ausgegeben oder sinnvollerweise zur weiteren Analyse und Bearbeitung im Racer-Server gelassen werden.

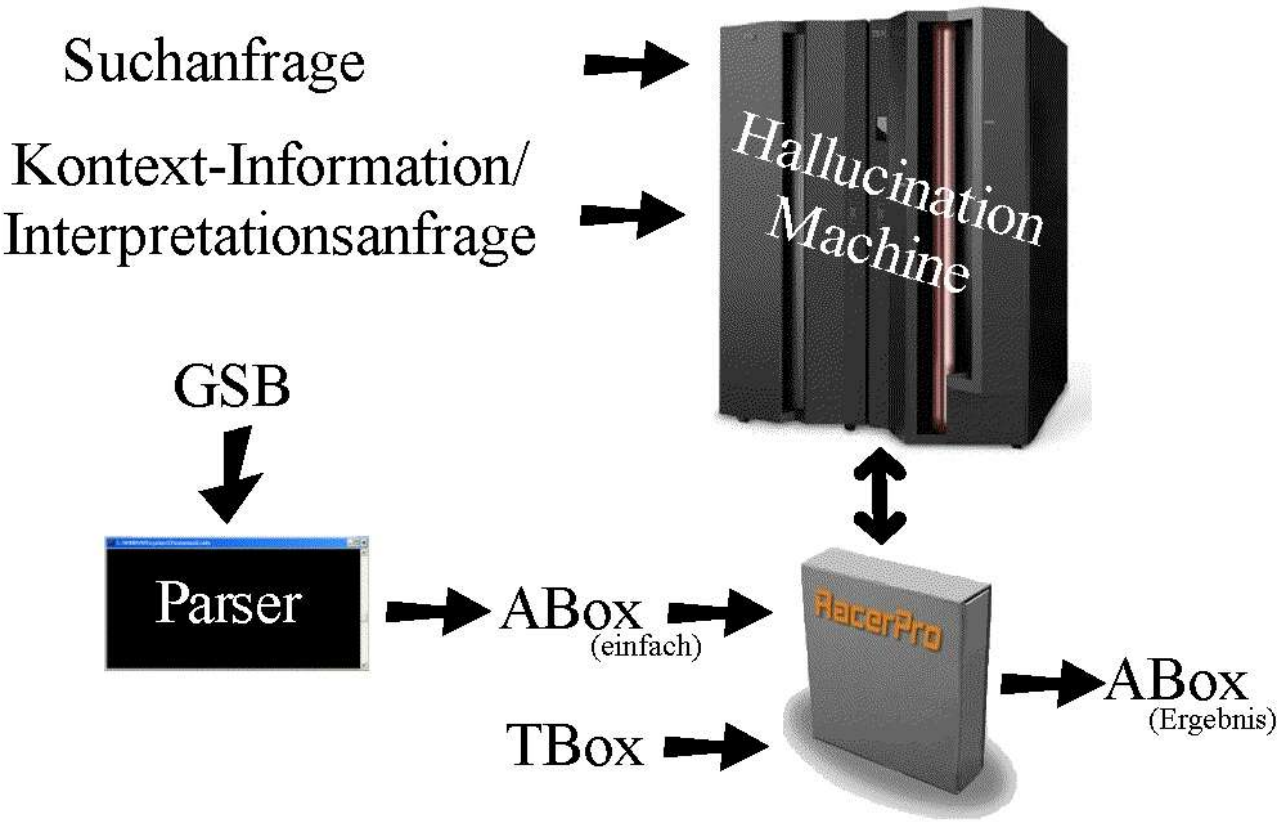

Abbildung 3.1: Darstellung der entwickelten Architektur

3.2 Beschreibung des GSB-Parsers

Wie bereits erwähnt wird eine Bildsequenz mittels Merkmalsextraktion in eine geometrische Szenenbeschreibung (GSB) überführt. Der Aufbau des Datenformats der GSB wurde bereits in Kapitel 2.4 behandelt. Für diese Arbeit wird die Merkmalsextraktion als gegeben angenommen, so dass bereits die GSB vorliegt.

Der GSB-Parser hat nun die Aufgabe diese GSB-Daten in eine entsprechende ABox zu überführen. Natürlich müssen entsprechende Konzepte, die in der ABox bzw. in der GSB auftauchen, auch in der TBox definiert sein. Diese entsprechende Definition mit entsprechenden Attributen wird vorausgesetzt. Die sich ergebende ABox wird zwischengespeichert und steht so für wiederholte Verarbeitung zur Verfügung.

Bei genauerer Betrachtung der GSB zeigt sich, dass für jeden Zeitpunkt bzw. *Frame* jeweils alle sichtbaren Gegenstände beschrieben werden mit jeweiliger Klassifikation, um welches Objekt es sich mit welcher Wahrscheinlichkeit, die nicht notwendig über die Zeit kontinuierlich sein muss, handelt. Z. B. bei dem Objekt mit dem Identifikator 1 handelt es sich zu 98% Wahrscheinlichkeit um eine Untertasse. Diese Klassifikation kann sich aber im Laufe der Zeit z. B. durch Verdeckung von Teilen oder des ganzen Gegenstandes verändern. Genauso gibt es weitere veränderliche Eigenschaften, wie z. B. ob das Objekt bewegt wird oder ruht etc.

Gerade bei der Wahrscheinlichkeit bzw. der *similarity measure* Eigenschaft stellt sich aber die Frage, ob und wie sinnvoll es ist, diese Informationen auch über die Zeit auszuwerten. Einerseits könnte ein über die Zeit maximal auftretender Wert in die ABox übernommen werden, anderseits würde auch durchaus ein Durchschnittswert Sinn machen. Beides ist aber unter Umständen relativ problematisch anzusehen.

Nehmen wir ein Beispiel, in dem ein Tisch nach und nach gedeckt wird:

Logischerweise wird erst die Untertasse hingestellt, bevor die Tasse aufgedeckt wird. Aber genau in diesem Anwendungsfall zeigen sich die Schwächen einer sensorischen Bildverarbeitung ohne eine höhere Interpretation. Beim Blick durch die Kameralinse erkennt man beim Bewegen der Tasse anfänglich noch korrekt bzw. mit einer hohen Wahrscheinlichkeit eine Tasse. Beim Positionieren der Tasse auf die Untertasse erfolgt dagegen nur mit einer sehr kleinen Wahrscheinlichkeit eine Erkennung für das Konzept Tasse. Die sensorische Bildverarbeitung kann in diesem Fall (so hat die Auswertung der GSB-Daten ergeben) nicht die Untertasse von der Tasse unterscheiden und klassifiziert deshalb beide als Untertasse.

```
(FR 1246 (ID 1 (PV TYPE SAUCER)(PV CENTER (435 191))(PV BOX (404 160 467 224))(PV
SM(20 10 17 0 3 98 ))(PV MOVE 0)))
```

Abbildung 3.2: Beispiel eines GSB-Frames

Bei einer maximalen Auswertung könnte es passieren, dass die Untertasse vielleicht um nur 1% besser erkannt wurde als die Tasse und wird somit auf jeden Fall falsch klassifiziert.
Bei der Bestimmung der Wahrscheinlichkeit anhand eines Durchschnittswertes würde der Zeitpunkt, an dem die Tasse auf die Untertasse gestellt wird und wie viele Bilder anschließend noch vorhanden sind, diesen Wert erheblich beeinflussen und damit die Gewichtung der korrekt erkannten Tasse in Bezug auf den Durchschnittswert.
Es kann auch sein, dass ein komplett falsches Objekt erkannt wird, weil am Anfang beim Hereinführen des Gegenstandes ins Bild der komplette Gegenstand nicht sichtbar ist. Dieses könnte in einer sehr kurzen Bildsequenz eine starke Gewichtung erhalten.
Aufgrund dieser Tatsachen habe ich mich entschlossen, nur die Wahrscheinlichkeitswerte des aktuell ausgewerteten Bildes in Betracht zu ziehen.
Für Daten wie die Koordinaten der Mitte des Gegenstandes bzw. des virtuellen umschließenden Kastens wird natürlich nur der aktuelle *Frame* ausgewertet. Der GSB-Parser betrachtet dabei immer das letzte Bild, welches in den Daten angegeben wird. Um diese Verfahren näher zu erläutern, schauen wir uns Abbildung 3.2 bis 3.4 an. In Abbildung 3.2 ist das zu verarbeitende Bild in GSB dargestellt. Dieses wird jetzt durch den Parser in die Form der Abbildung 3.3 überführt.
Es werden nur solche Attribute in die ABox geschrieben über die auch eine logische Schlussfolgerung (engl. *reasoning*) in Racer durchgeführt wird bzw. unter Umständen durchgeführt werden kann. Es macht daher für die in dieser Arbeit beschriebene Anwendung wenig Sinn z. B. die verschiedenen Wahrscheinlichkeiten für die einzelnen Gegenstände in die ABox zu schreiben, da sie lediglich den Racer-Server unnötig belasten würden. Deshalb werden diese Informationen in einer extra Datei gespeichert, um diese später wieder in der *Hallucination Machine* verfügbar zu machen.
Dieses verdeutlicht die Abbildung 3.4 in der neben dem Schlüsselwort *Object* (engl. für Objekt) der Identifikator und, ähnlich der GSB-Syntax geklammert, die möglichen Gegenstände inklusive der betreffenden Wahrscheinlichkeit enthalten sind.

```
(instance Object-1 SAUCER)
(constrained Object-1 Object-1-x-coordinate x-coordinate)
(constraints (= Object-1-x-coordinate 435))
(constrained Object-1 Object-1-y-coordinate y-coordinate)
(constraints (= Object-1-y-coordinate 191))
(constrained Object-1 Object-1-x-Corner1-BoundingBox x-Corner1-BoundingBox)
(constraints (= Object-1-x-Corner1-BoundingBox 404))
(constrained Object-1 Object-1-y-Corner1-BoundingBox y-Corner1-BoundingBox)
(constraints (= Object-1-y-Corner1-BoundingBox 160))
(constrained Object-1 Object-1-x-Corner2-BoundingBox x-Corner2-BoundingBox)
(constraints (= Object-1-x-Corner2-BoundingBox 467))
(constrained Object-1 Object-1-y-Corner2-BoundingBox y-Corner2-BoundingBox)
(constraints (= Object-1-y-Corner2-BoundingBox 224))
```

Abbildung 3.3: Entsprechender ABox-Ausschnitt zu Abbildung 3.2

Es gibt aber durchaus zeitliche Informationen, die nicht vernachlässigbar sind. So werden Bewegungen einzelner Objekte festgehalten und in die ABox übernommen. Entscheidend hierfür ist die Eigenschaft *Move* die, solange sie den binären Wert „1“ aufweist, auf eine Bewegung hindeutet. Es hat sich anhand der Probedaten gezeigt, dass es für ein bis zwei Bilder aufgrund von sehr kleinen Bewegungen zu einer Fehlinterpretation der *Move* Eigenschaft kommen kann. Deshalb hat sich eine gewisse Toleranz gegenüber dieser Fehleinschätzung bewährt, und es werden so eigentlich zwei oder mehr Bewegungen zu einer zusammengefügt.
Das Konzept *Movement* (engl. für Bewegung) beinhaltet dabei die Start- und Endzeit der Bewegungen, deren Angabe in den *Frame*-Nummern erfolgt und die x- und y-Koordinaten der Start- und Endpunkte, wobei die entsprechenden Mittelpunkte des Objektes gespeichert werden. Das sich bewegende Objekt wird mittels der Rolle *moved* (engl. für bewegt) an die Bewegung gebunden.

```
(Object 1 (Fork 20) (Spoon 10) (Knife 17) (Plate 0) (Cup 3) (Saucer 98))
```

Abbildung 3.4: Entsprechender Zusatzeintrag zu Abbildung 3.2

Weiterhin werden die Berührungen einzelner Gegenstände protokolliert, wie z. B. eine Hand beim Decken eines Tisches einen Teller berührt, um ihn zu seiner entsprechenden Endposition zu führen. Hierfür werden die betreffenden umschließenden Kästen der Objekte über die Zeit hin verglichen und beim Auftreten einer Überschneidung wird von einer Berührung ausgegangen. Das entsprechende Konzept *Touch* (engl. für Berührung) besitzt lediglich die Start- und Endzeit als Attribut.

Die Gegenstände, die sich berühren, werden mittels der Rolle *touched* (engl. für berührt) an das Individuum für das Konzept *Touch* gebunden, die Bindung untereinander erfolgt mit der Rolle *touched-by* (engl. für berührt durch).

Alle Attribute werden vom GSB-Parser als so genannte *Concrete Domain* (engl. für konkrete Domäne) gespeichert. *Concrete Domains* wurden mit der Sprache $\mathcal{ALC}(\mathcal{D})$ [24] eingeführt [19]. Dieses erlaubt das Einbinden von Prädikaten, die außerhalb des Beschreibungslogik-Beweisers ausgewertet werden können [19]. Ein Beispiel wäre z. B. das Vergleichen von reellen Zahlen auf Gleichheit, Ungleichheit etc. Mehr Informationen über *Concrete Domain* siehe [25].

Es wären durchaus noch weitere interessante Informationen denkbar, die in die ABox geschrieben werden könnten, aber für diese Beispiel-Anwendung sind diese Daten durchaus ausreichend. Für andere Fälle wäre der Parser durchaus erweiter- bzw. veränderbar.

Der Parser erhebt keinen Anspruch auf Allgemeinheit, da die GSB-Daten in diesem Fall auch sehr speziell auf dieses Szenario zugeschnitten sind (z. B. keine Verwendung der Farbbeschreibung oder des Blickwinkels etc.)

3.3 Beschreibung der Suchanfragen

Als Basis für die *Hallucination Machine* wird Racer benutzt, der die Sprache $\mathcal{SHIQ}(\mathcal{D}_n^-)$ [12] implementiert. Die Sprache hat aber nur eine beschränkte Mächtigkeit in Bezug auf so genannte *feature chain agreements*, die nicht erlaubt werden ([19] und [26]).
Ein *feature chain agreement* ist

(same-as F_1 F_2),

wobei F_1 bzw. F_2 ein *feature* (engl. für Eigenschaft) ist, welches auch als Attribut bezeichnet werden kann.
Eine *feature chain* ist eine Komposition aus verschiedenen *features*, also

(compose F_1 ... F_n).

Sollen also ein Teller und eine Untertasse die gleiche Farbe haben, kann das als

(same-as (compose has-plate has-colour) (compose has-saucer has-colour))

ausgedrückt werden [19].
Dieses ist aber in der Sprache $\mathcal{SHIQ}(\mathcal{D}_n^-)$ nicht zulässig und damit in dieser Weise auch nicht in Racer darstellbar.
Weiterhin sind auch so genannte *role-value map*, also

(subset R_1 R_2)

(Teilmengen von Rollen) nicht erlaubt. Damit wäre ein Konzept, wie es in Abbildung 3.5 dargestellt ist, aufgrund der Mächtigkeit der Sprache nur zum Teil in Racer zu realisieren.
Problematisch sind die beiden letzten Zeilen, die ausdrücken, dass dieselbe *plate* nahe (engl. *near*) einer *saucer* ist, die in den Rollen *cv-pl* und *cv-sc* referiert werden, bzw. dieselbe *cup* auf (engl. *on*) der *saucer* steht, wie in den Rollen *cv-sc* und *cv-cp*.

```
(equivalent cover
    (and configuration
        (exactly 1 cv-pl plate)
        (exactly 1 cv-sc (and saucer (some near plate)))
        (exactly 1 cv-cp (and cup (some on saucer)))
        (subset cv-pl (compose cv-sc near))
        (subset cv-sc (compose cv-cp on))))
```

Abbildung 3.5: Beschreibungslogik-Konzept für ein einfaches Gedeck aus [19]

Nun sind aber genau diese Beziehungen nicht unwichtig, um ein Gedeck zu beschreiben. Selbst bei einer eventuell möglichen Vernachlässigung dieser Beziehung könnten Beispiele aufgezeigt werden, die bei einer nicht vollständigen Definition zu einer Verfälschung führen.
Daraus folgt, dass die Konzepte also nicht vollständig in der Racer-TBox definiert werden können. Allerdings könnte dieses Konzept als Anfrage in der *New Racer Query Language* [27] (engl. für neue Racer Anfragesprache, kurz: nRQL) formuliert werden, wie in [19] beschrieben:

```
(retrieve (?x ?y ?z)   (and   (?x plate)
                              (?y saucer)
                              (?z cup)
                              (?x ?y near)
                              (?z ?y on)))
```

Racer liefert auf diese Anfrage alle *plate*, *saucer* und *cup* Individuen zurück, die die entsprechenden Rollen *near* und *on* erfüllen und damit alle möglichen *cover* Konzepte, da genau die mittels der Anfrage definiert worden sind.
Es ist also durchaus möglich, für diese Anwendung die Beschränkung der Sprache $\mathcal{SHIQ}(\mathcal{D}_n^-)$, durch entsprechende Suchanfragen zu umgehen. Deshalb müssen alle eventuell auftretenden Konzepte mittels dieser Anfragen definiert und an die *Hallucination Machine* übergeben werden.
Aus verschiedenen Gründen werden diese allerdings nicht in der nRQL-Syntax formuliert, sondern nur in einer nRQL ähnlichen Sprache.

```
(find-new-instance cover ((?x plate (value 1)) (?y saucer (value 0.5)) (?z cup (value 0.1))  (?y ?z on (value 1)) (?x ?z near (value 0.7))) ((?x cv-plate) (?y cv-saucer) (?z cv-cup)))
```

Abbildung 3.6: Anfrage in Hallucination Machine-Syntax

Die *Hallucination Machine* benötigt zu noch zusätzliche Informationen (z. B. die Gewichtung) und analysiert Teile der Anfrage, um sie zu verstehen.
Abbildung 3.6 zeigt eine solche Anfrage. Dabei ähnelt grundsätzlich diese Anfrage dem nRQL-Befehl *Firerule*, der in Abbildung 3.7 dargestellt ist. Dieser Befehl erstellt eine *Rule* (engl. für Regel) und führt diese gleich aus. Der Befehl hat folgendes Aussehen:

(firerule <*rule-antecedence*> <*rule-consequence*>)

Der *rule-antecedence* (engl. für Regel-Bedingungsteil) enthält die eigentlich Anfrage, wie sie oben bereits vorgestellt wurde, während die *rule-consequence* (engl. für Regel-Folgerung) den auszuführenden Teil besitzt. So wird z. B. in Abbildung 3.7 ein *cover* instanziiert und die gefundenen Individuen über Rollen an sich gebunden, wenn der Bedingungsteil erfüllt ist. Die Anfrage für die Hallucination Machine sieht ähnlich aus:

(find-new-instance <*concept*> <*antecedence*> <*consequence*>)

Es fällt auf, dass als erstes ein *concept* (engl. Konzept) mit angegeben werden muss, welches das zu instanzierende Konzept nennt.
Weiterhin beinhaltet *antecedence* noch zusätzlich zu der Anfrage, die allerdings ohne ein *and* (engl. für und) angegeben wird, zu jeder einzelnen Anfrage einen geklammerten *value* (engl. für Wert) Ausdruck.

```
(firerule (and (?x plate) (?y saucer) (?z cup) (?y ?z on)(?x ?z near)) ((instance (new-ind Found-cover-with ?x ?y ?z) cover) (related (new-ind Found-cover-with ?x ?y ?z) ?x cv-plate)(related (new-ind Found-cover-with ?x ?y ?z) ?y cv-saucer)(related (new-ind Found-cover-with ?x ?y ?z) ?z cv-cup)))
```

Abbildung 3.7: Anfrage als nRQL-Firerule formuliert

```
(find-new-instance cover ((?x plate (value 1))) ((?x cv-plate) (constrained NEWINDIVIDUAL
NEWINDIVIDUAL-x-coordinate x-coordinate)(CONSTRAINTS (= NEWINDIVIDUAL-x-
coordinate (TOLD-VALUE (x-coordinate ?x)))))
```

Abbildung 3.8: Anfrage in Hallucination Machine-Syntax mit Attribut Übernahme

Diese Werte geben die einzelne Gewichtung für die spätere Bewertung in der *Hallucination Machine* an. Der Wert darf zwischen null und eins liegen, wobei eins die stärkere Gewichtung angibt.

So ist in Abbildung 3.6 der Teller am stärksten gewichtet und somit auch am wichtigsten für ein Gedeck. Am unwichtigsten ist dagegen die Tasse mit dem Wert 0.1. Auch die Rollen zwischen den zu suchenden Individuen erhalten eine Gewichtung.

Im *consequence*-Teil wird nur die betreffende Variable (z. B. ?x) mit der entsprechenden Rolle angegeben, die das Individuum an das instanziierte Konzept bindet. Da es anhand der Anwendung um das Instanziieren eines Konzeptes geht, sind somit nicht mehr Informationen nötig.

Es sind auch andere Folgerungen möglich, wie z. B. in Abbildung 3.8:

Hierbei wird zusätzlich zu einem einfachen Gedeck, welches nur aus einem Teller besteht, noch die x-Koordinate des Tellers als x-Koordinate des Gedecks übergeben. Da für diesen oder auch andere Befehle, die von der *Hallucination Machine* in eine nRQL-*Firerule* umgeschrieben werden, eigentlich der Name des neuen Individuums bekannt sein müsste oder wie in nRQL mittels Befehl *new-ind* (vgl. Abblidung 3.9) angegeben wird, benutzt die *Hallucination Machine* das Schlüsselwort *NEWINDIVIDUAL*. Dieser Wort dient als Platzhalter und wird von der *Hallucination Machine* entsprechend ersetzt. Ein solche von der *Hallucination Machine* umgeformte *Firerule* ist in Abbildung 3.9 dargestellt.

Ein weiterer Anfrage-Typ ist in Abbildung 3.10 zu sehen. Diese ähnelt eher einer normalen nRQL-Anfrage und dient zum Finden und Setzen der entsprechenden Rollen.

```
(firerule (and (?x plate) ((instance (new-ind Found-cover-with ?x) cover) (related (new-ind Found-
cover-with ?x) ?x cv-plate) (constrained (new-ind Found-cover-with ?x)  (new-ind x-coordinate-
Found-cover-with ?x)  x-coordinate)(CONSTRAINTS (= (new-ind x-coordinate-Found-cover-with
?x)  (TOLD-VALUE (x-coordinate ?x)))))
```

Abbildung 3.9: Anfrage aus Abbildung 3.8 als nRQL-Firerule

```
(find-new-role on (?x ?y) (and  (?x ?y  (constraint (y-coordinate) (y-coordinate) (<  y-coordinate-1
(+ 10  y-coordinate-2) ))) (?x ?y  (constraint (y-coordinate) (y-coordinate) (>  (+ 10 y-coordinate-1)
y-coordinate-2 )))   (?x ?y  (constraint (x-coordinate) (x-coordinate) (<  x-coordinate-1  (+ 10  x-
coordinate-2) )))  (?x ?y  (constraint (x-coordinate) (x-coordinate) (> (+ 10 x-coordinate-1)  x-
coordinate-2 ))) ))
```

Abbildung 3.10: Anfrage für die Hallucination Machine, um Rollen zu finden

So werden neben dem Befehl *find-new-role* der Rollenname, die Suchvariablen und die Suchbedingungen angegeben. Auch dieser Befehl wird wieder in eine nRQL-*Firerule* umgeformt und dient zum Setzen entsprechender Rollen, damit sie in der *find-new-instance*-Anfrage zur Verfügung stehen.

Diese Befehle werden nun in eine Datei geschrieben, wobei pro Befehl eine Zeile verwendet wird. Ähnlich wie in der Programmiersprache Lisp können dabei nicht verwendete Befehle mit einem vorangestellten Semikolon auskommentiert werden.

3.4 Beschreibung der *Hallucination Machine*

Mit der Beschreibung des GSB-Parsers in Kapitel 3.2 und der daraus resultierenden $ABox_{(einfach)}$ sowie der Erläuterung der Suchanfragen in Kapitel 3.3 sind bereits ein Großteil der Eingangsdaten der *Hallucination Machine* abgehandelt worden. Es fehlen lediglich noch die Kontext-Information bzw. Interpretationsanfrage und die TBox. Dabei enthält die Kontext-Informations-Datei in jeder Zeile ein Konzept, das natürlich in der TBox bzw. in den Suchanfragen definiert sein muss. Interpretationsanfragen werden ebenfalls mit in die Kontext-Information hineingeschrieben, da sowohl die Kontext-Information als auch die Interpretationsanfrage bewertet werden und somit praktisch kein Unterschied existiert.

Die TBox enthält entsprechendes Wissen über die Konzepte und wird ansonsten nicht weiterverarbeitet. Allerdings wird vorausgesetzt, dass die TBox ein Konzept *used* (engl. für benutzt) und die Rolle *are* (engl. für sind, wird) enthält. Diese beiden Vorgaben werden benutzt, um bereits in anderen Konzepten verwendete Individuen zu markieren. Dabei wird automatisch vom Programm das Individuum durch die Rolle *are* an ein Individuum des Konzepts *used* gebunden. Daraus ergibt sich der Vorteil das die einzelnen Gegenstände nicht mehrfach gebraucht werden und so z. B. eng beieinander stehende Gedecke oder eine großzügige Formulierung des Begriffs Nähe nicht dazu führen, das mehrere Gedecke die gleichen Gegenstände benutzen, was in der wirklichen Welt auch nicht vorkommt, da z. B. eine Gabel nur von einer Person zur Zeit benutzen kann.

Bevor die einzelnen Schritte, die zur Auswertung des Bildes führen, näher erläutert werden erfolgt ein erster Gedankengang über die Instanziierung von Teilen eines Konzeptes oder anders formuliert, was passiert wenn z. B. kein Gedeck gefunden wird weil z. B. die Tasse fehlt?

Natürlich ist ein Gedeck für einen Menschen immer noch ein Gedeck, auch wenn ein Gegenstand daraus fehlt. Die Schlussfolgerung ist, die *Hallucination Machine* würde das Gedeck ohne eine Tasse ebenfalls instanziieren, da aber ein Teil fehlt fällt die Bewertung schlechter aus als bei einem vollständig gefundenen Gedeck. Da zuerst nach der Kontext-Information gesucht wird, sollte eine solche Information nach Möglichkeit vollständig, mindestens aber teilweise gefunden werden.

Natürlich kann man durch eine schlechte Kontext-Information das Programm irreführen, und es wird z. B. ein Gedeck am Hamburger Rathaus gefunden (vergleiche Beispiel aus Kapitel 2.3), aber dieses würde dann zu einer entsprechend schlechteren Bewertung führen. Eine weitaus wichtigere Frage ist allerdings: Was passiert, wenn man ein Gedeck mit allen Gegenständen findet, aber die betreffenden Rollen werden nicht erfüllt?

Also in diesem konkreten Beispiel: Alles trifft zu, nur die Tasse erfüllt die Rollen *on* und *near* nicht. D.h. die Tasse befindet sich vielleicht nicht in der Nähe des Tellers und steht nicht auf der Untertasse. Auf den ersten Blick sicherlich kein Problem, weil man genauso wie beim Beispiel des Hamburger Rathauses mit einer schlechteren Bewertung argumentieren kann. Aber es stellen sich durchaus noch andere Schwierigkeiten ein.

Eine Tasse, die die Rollen nicht erfüllt, steht womöglich weiter weg und kann durchaus zu diesem Gedeck gehören, unter Umständen aber auch zu einem anderen Gedeck. Es ergibt sich das Problem, das abhängig von der Suchreihenfolge verschiedene Ergebnisse erzielt werden. Einerseits kann ein komplettes Gedeck mit allen Rollen gefunden werden und ein Gedeck ohne Tasse oder andererseits abhängig von der Ergebnisreihenfolge wird erst ein vollständiges Gedeck ohne allerdings entsprechende Rollen für die Tasse und ein Gedeck ohne Tasse erstellt. Diese Tatsache ist nicht unbedingt gewünscht und verfälscht die Bewertung je nach Ausführungsreihenfolge.

Grundsätzlich ist die Vernachlässigung der Rollen immer problematisch. Stellen wir uns vor, es soll ein aus dem Fenster blickender Mann gefunden werden, und auf dem Foto sind tatsächlich ein Mann und ein Fenster zu sehen. Allerdings geht dieser Mann vielleicht nur die Straße entlang und ist nicht in der Nähe des Fensters. Ein Mensch würde nicht auf die Idee kommen, diese Szene als „Mann-sieht-aus-dem-Fenster“-Konzept zu klassifizieren. Er würde wohl auf die Frage, ob ein Mann aus dem Fenster schaut, antworten, dass durchaus ein Fenster zu sehen ist, aber kein Mann, der daraus herausschaut oder dass ein Mann zu sehen ist, aber der schaut nicht aus dem Fenster. Dieses wäre gleichbedeutend mit einem Mann, der nicht die Rolle „schaut aus“ in Bezug Fenster erfüllt und das Konzept Mann-sieht-aus-dem-Fenster besitzt nur ein Individuum Mann.

Aufgrund dieser starken Gewichtung der Rollen in der menschlichen Auswertung des Gesehenen, wird auch in dieser Arbeit den Rollen eine starke Position in der Erkennung von Konzepten beigemessen.

Ein Einwand wäre sicher: Keine gute Halluzination ergibt sich, wenn als Ergebnis nur ein Gedeck ohne Messer zu finden ist, weil das Messer nicht die Rolle *near* in Bezug auf einen Teller erfüllt, da es etwas zu weit davon entfernt liegt. Dieser Einspruch ist durchaus berechtigt, allerdings sollten Rollen, die sowieso nur relativ formuliert sind wie z. B. Nähe nicht zu eng bemessen werden, damit solche Fälle nicht auftreten. Menschen sehen solche Begriffe, wie in der Nähe von etc. auch äußerst subjektiv.

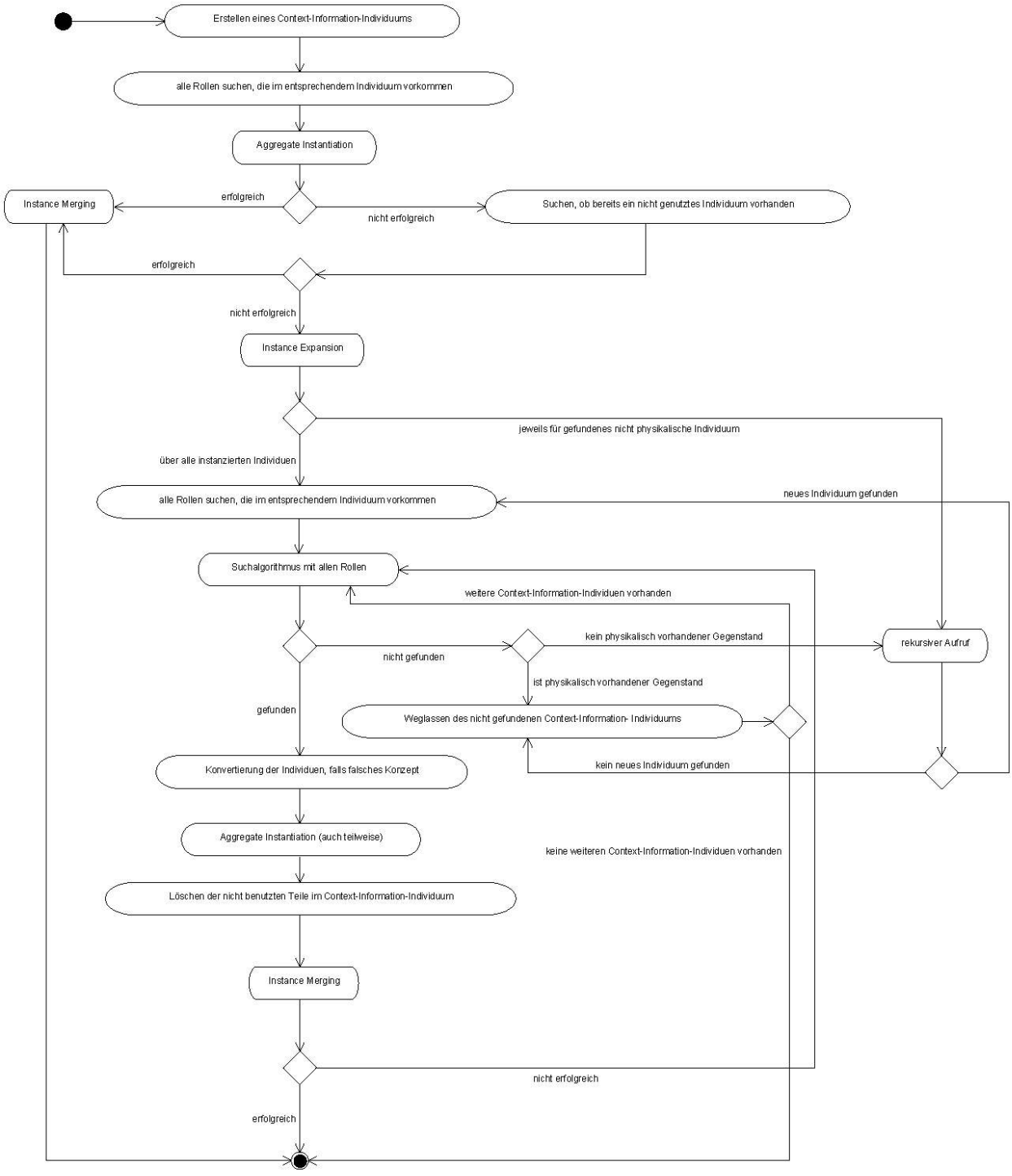

Abbildung 3.11: Darstellung der Funktionsweise der Verarbeitung einer Kontext-Information als Aktivitätsdiagramm

Im weiteren wird die Funktionsweise der *Hallucination Machine* dargestellt:
Die Übergabe der Eingabedateien erfolgt mittels Parameter beim Starten der *Hallucination Machine*, diese leitet die entsprechende $Abox_{(einfach)}$ bzw. TBox entsprechend der Abbildung 3.1 an den Racer-Server weiter. Anschließend werden die Kontext-Informationen bzw. Interpretationsanfragen der Reihe nach in den Hauptspeicher eingelesen. In den Suchanfragen werden die Bedingungsteile und Folgerungen je Konzept analysiert und ebenfalls in den Hauptspeicher geschrieben.
In Abbildung 3.11 ist ein Schema des nun folgenden als Aktivitätsdiagramm dargestellt. Zuerst wird eine Kontext-Information aus dem Hauptspeicher geholt und als Individuum erstellt. Hierfür wird als Name der Prefix „Context-Info-“ zusätzlich zum Konzept-Namen und einer fortlaufenden Nummer gewählt. Also wird ein *cover*, erstellt als Kontext-Information, als „Context-Info-Cover-0“ angelegt. Es soll nun versucht werden, dieses angelegte Individuum zu finden und durch den bereits in Kapitel 2.3 erwähnten Befehl *Instance Merging* wiederum zu löschen bzw. mit dem gefundenen Element zu vereinigen. Hierfür werden zunächst alle Rollen, die die Beziehungen der zu suchenden Individuen untereinander beschreiben (also z. B. die Rolle *on* für die Individuen *cup* und *saucer*), gesucht. Dieses geschieht mittels entsprechenden *find-new-role* Befehlen aus den eingegebenen Suchanfragen.
Anschließend wird das in Kapitel 2.3 erwähnte *Aggregate Instantiation* ausgeführt. Messungen haben dabei ergeben, dass eine Anfrage an Racer schneller verarbeitet werden kann, wenn vorweg die Einzelteile gesucht werden.
Statt z. B.

```
(retrieve (?x ?y ?z) (and (?x plate) (?z saucer) (?y cup) (?z ?y on)))
```

würde man also

```
(retrieve (?x) (?x cup))
(retrieve (?x) (?x saucer))
(retrieve (?x) (?x plate))
(retrieve (?x ?y) (?x ?y on))
(retrieve (?x ?y ?z) (and (?x plate) (?z saucer) (?y cup) (?z ?y on)))
```

schreiben.

Damit erhält Racer die Möglichkeit seinen Cache zu füllen und entsprechend schneller zu antworten. Bedingung ist allerdings die Aktivierung dieser Funktion mit dem Befehl „(ensure-subsumption-based-query-answering)“. Die *Hallucination Machine* nutzt diesen Vorteil bei der *Aggregate Instantiation*, indem vor dem eigentlichen Suchen bzw. Erstellung des Individuums mittels nRQL-Befehl *firerule,* auch einzelne Suchanfragen gestellt werden.

In diesem Schritt werden allerdings nur vollständig gefundene Individuen erstellt, wenn einige Teile fehlen oder nicht die entsprechenden Rollen erfüllen wie z. B. oben beschrieben ein Gedeck ohne Tasse dann wird erst einmal kein Gedeck gefunden. Außerdem dürfen die Individuen noch nicht über die Rolle *are* an ein Individuum des Konzepts *used* gebunden sein, d.h. sie werden noch nicht in einem anderen Zusammenhang verwendet. Diese Bedingung wird mit in den nRQL-Befehl eingebunden.

Es könnte allerdings geschehen, dass z. B. ein *Dinner-for-two* (engl. für Essen für zwei), welches zwei Gedecke enthalten soll, jeweils das Kreuzprodukt findet und erstellt. Es wird also ein *dinner-for-two* mit z. B. *cover-0* und *cover-1* gefunden und ein anderes *dinner-for-two* mit *cover-1* und *cover-0*. Das ist natürlich zweimal dasselbe *dinner-for-two*. Darum vergleicht die *Hallucination Machine*, die erstellten Individuen und löscht alle überzähligen, wenn die gleichen referenziert wurden.

Sollte nun die Suchanfrage bzw. die von dem Programm an Racer gesendete nRQL-*firerule* positiv ausfallen und somit ein Konzept, das der erstellten Kontext-Information entspricht, gefunden werden, dann können diese beiden Individuen im *Instance Merging*-Schritt vereinigt werden. Eine genauere Beschreibung dieses Schrittes erfolgt im Rahmen dieser Arbeit.

Kann kein entsprechendes Individuum erstellt werden, besteht immer noch die Möglichkeit, dass bereits bei einer vorherigen *Aggregate Instantiation* ein entsprechendes Individuum gefunden wurde, ohne allerdings mit einem Kontext-Information-Objekt vereinigt zu sein. Existieren z. B. zweimal die Gegenstände für zwei unterschiedliche Gedecke mit entsprechenden Rollen, werden bereits bei der ersten Suche bzw. *Aggregate Instantiation* beide Gedecke gefunden und entsprechend angelegt. Eins davon wird nun im *Instance Merging*-Schritt mit der betreffenden Kontext-Information vereinigt. Allerdings wird jetzt bei der nächsten *Aggregate Instantiation* kein Gedeck mehr gefunden, obwohl bzw. gerade weil bereits eines vorhanden ist. So wird nun nach einem betreffenden Individuum gesucht, welches nicht mit der Rolle *are* an ein Individuum des Konzepts *used* gebunden ist. Ebenso werden Suchergebnisse ignoriert, die bereits vereinigt wurden. Dieses wird mittels einer intern verwalteten Liste abgeglichen. So würde das zweite Gedeck gefunden und kann entsprechend weiter behandelt werden.

Ist diese Suche positiv verlaufen, geht es ebenfalls wieder zum *Instance Merging*-Schritt, ansonsten zum *Instance Expansion*, deren Inhalt bereits auch in Kapitel 2.3 beschrieben wurde. Hierbei werden alle Individuen als Kontext-Information erstellt, die laut der Suchanfrage mit allen entsprechenden Rollen vorkommen müssen. Dabei werden die Elemente, wie oben beschrieben, wieder mit dem Prefix „Context-Info-“ versehen und weiter nach dem Konzept und einer laufenden Nummer benannt.
Nun gibt es genau zwei Arten von instantiierten Objekten:

- Individuen, die direkt physikalische Gegenstände sind, wie z. B. Teller
- höhere Konzepte, die keine physikalischen Objekte repräsentieren

Ist ein höheres Konzept erstellt worden, dann wird das gerade beschriebene und noch folgende Verfahren rekursiv aufgerufen und es beginnt wieder mit der Suche nach allen entsprechenden Rollen, der *Aggregate Instantiation* etc. Dieser Zusammenhang ist in Abbildung 3.11 als „rekursiver Aufruf“ gekennzeichnet, in der ein höheres Konzept über eine vorhandene Definition in den Suchanfragen ermittelt wird.
Alle erstellten Individuen, die physikalisch und nicht physikalisch vorhandene Gegenstände repräsentieren, werden gemeinsam in einem Suchalgorithmus bearbeitet. Dieser Suchalgorithmus ist als Aktivitätsdiagramm in Abbildung 3.12 dargestellt. Logisch wird in diesem Algorithmus ein Suchbaum aufgestellt und durchsucht, bis ein lokal optimales Ergebnis gefunden wird.
Zuerst erfolgt die Sortierung der erstellten Kontext-Informations-Individuen absteigend nach der Anzahl der Rollen. Hierfür werden in der entsprechenden Suchanfrage, aus der die Individuen entstanden sind, die jeweiligen Rollen untereinander ausgewertet und zusammengezählt. Dies garantiert eine Beschränkung des Suchraumes, da das Individuum mit den meisten Rollenverbindungen, die auch als Randbedingungen angesehen werden können, auch eine kleine Ergebnismenge liefert. Weiterhin besteht der Vorteil, dass die Suche schnell abgebrochen werden kann, wenn keins gefunden wird.
Von dieser sortierten Liste kommt das oberste Individum mit den meisten Nebenbedingungen zur Anwendung, um es in eine nRQL-Anfrage zu formulieren. In der Anfrage wird nach einem passenden Individuum gesucht, um es mit dem betreffenden Kontext-Information-Individuum zu vereinigen. Es erfolgt das Einbinden aller Rollen, aber nicht der Konzepte, in diese Anfrage, da die betreffenden Individuen bei der sensorischen Verarbeitung falsch erkannt worden sein könnten. Wiederum werden alle Individuen, die bereits mit der Rolle *are* an ein Individuum des Konzepts

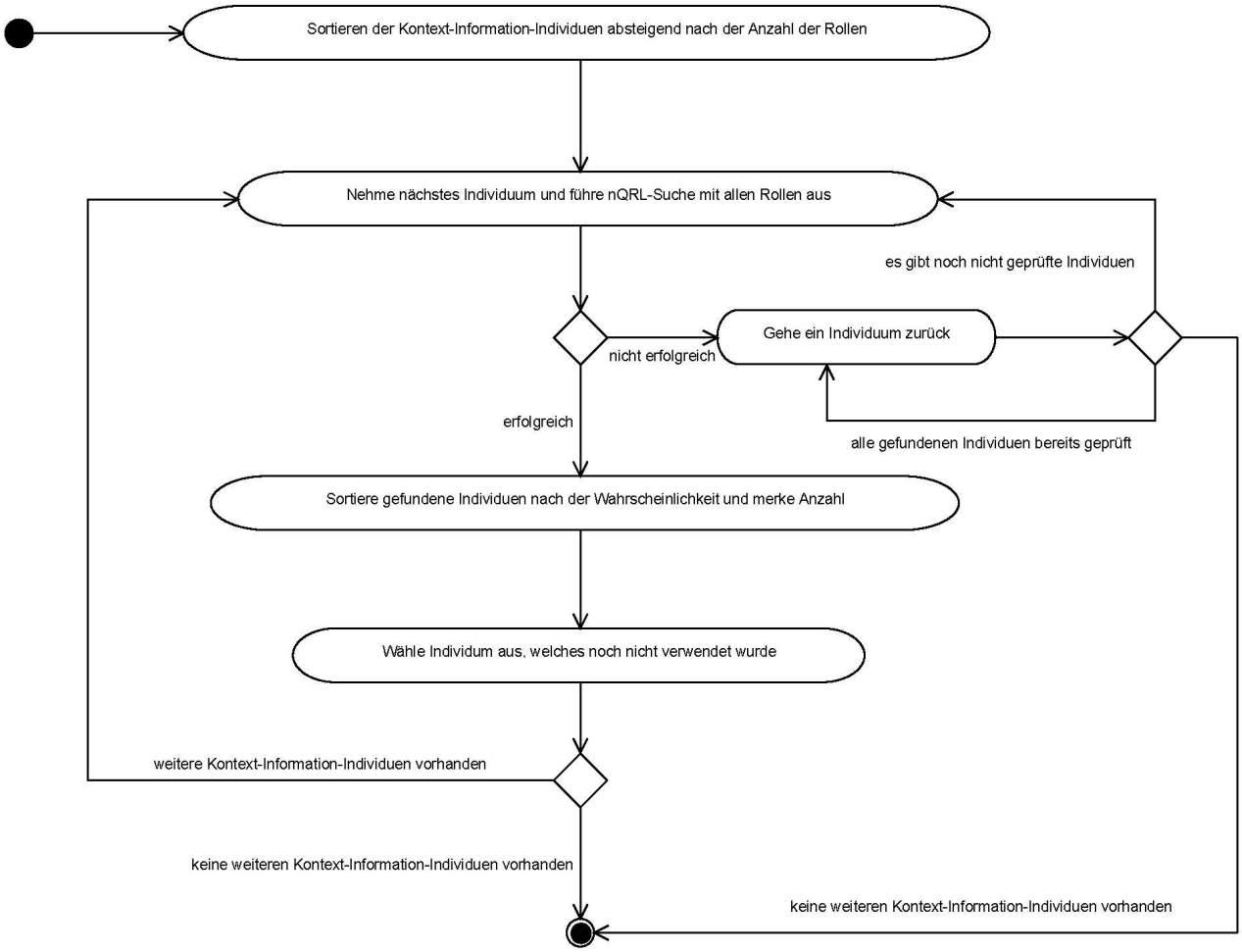

Abbildung 3.12: Darstellung der Funktionsweise des Suchalgorithmus als Aktivitätsdiagramm

```
(retrieve (?z) (and (nil ?z cv-cup) (nil ?z are) (?y ?z on) (nil ?y are) (OBJECT-2 ?z near) (nil
OBJECT-2 are) (?z ?c near) (nil ?c are)))
```

Abbildung 3.13: nRQL-Beispielanfrage für den Suchalgorithmus

used gebunden sind ignoriert und fließen auch in die Anfrage mit ein.

Ein Beispiel zeigt Abbildung 3.13. Es verdeutlicht auch, dass bereits gefundene Individuen (hier: *OBJECT-2*) in die Anfrage ebenfalls mit einfließen, die bereits weiter oben in der sortierten Liste gefunden und als korrekt angenommen wurden. Sollte nun kein Individuum als Ergebnis zurückgeliefert werden, ist entweder das gefundene *OBJECT-2* ein falsch gewähltes oder aber es existiert wirklich kein betreffendes Individuum mit diesen Randbedingungen.

Bei Erfolg der formulierten nRQL-Anfrage, d.h. es werden ein oder mehrere Individuen gefunden, wird das Ergebnis absteigend nach der Wahrscheinlichkeit, die die sensorische Verarbeitung ermittelt hat, sortiert. Individuen, deren Wahrscheinlichkeit in Bezug auf das gesuchte Konzept null beträgt, und bereits erstellte Individuen (gekennzeichnet durch das Prefix „Found-“), sowie Kontext-Information werden nicht als Ergebnis berücksichtigt. Es erfolgt die interne Speicherung der ungeprüften Ergebnis-Individuen.

Zum Einsatz kommt das Individuum, welches die größte Wahrscheinlichkeit hat und noch nicht in einem möglichen vorherigen Durchlauf ausgewählt worden ist. Außerdem werden eventuell vorherige gewählte Individuen wie z. B. für Abbildung 3.13 *Object-2* aussortiert, falls diese ebenfalls als Ergebnis zurückgeliefert werden. Gibt es nun noch weitere Kontext-Information-Individuen, die hier durchsucht werden sollen, dann wird wiederum die nächste nRQL-Anfrage erstellt, wie in Abbildung 3.12 zu sehen ist. Der Suchalgorithmus ist mit einer lokal optimalen Lösung erfolgreich beendet, wenn es kein weiteres Kontext-Individuum gibt.

Wird allerdings in der nRQL-Anfrage kein Individuum gefunden, geht es in der Kontext-Informations-Liste einen Schritt zurück und es erfolgt eine Prüfung, ob noch ungeklärte Ergebnisse vorliegen. Gibt es weitere Individuen, die noch nicht in Bezug auf dieses Kontext-Informations-Element getestet wurden, dann wird mit diesen wiederum eine nRQL-Anfrage erstellt und wie beschrieben bzw. in Abbildung 3.12 dargestellt weiterverfahren. Liegen keine ungeprüften Ergebnisse vor, dann wird noch ein weiterer Schritt in der Kontext-Information-Liste zurückgegangen bis es ein noch nicht durchlaufendes Individuum gibt oder man das oberste Element in der Liste erreicht hat und dort ebenfalls keine weitere Suchmöglichkeit besteht. Dann wird der Algorithmus als erfolglos abgebrochen.

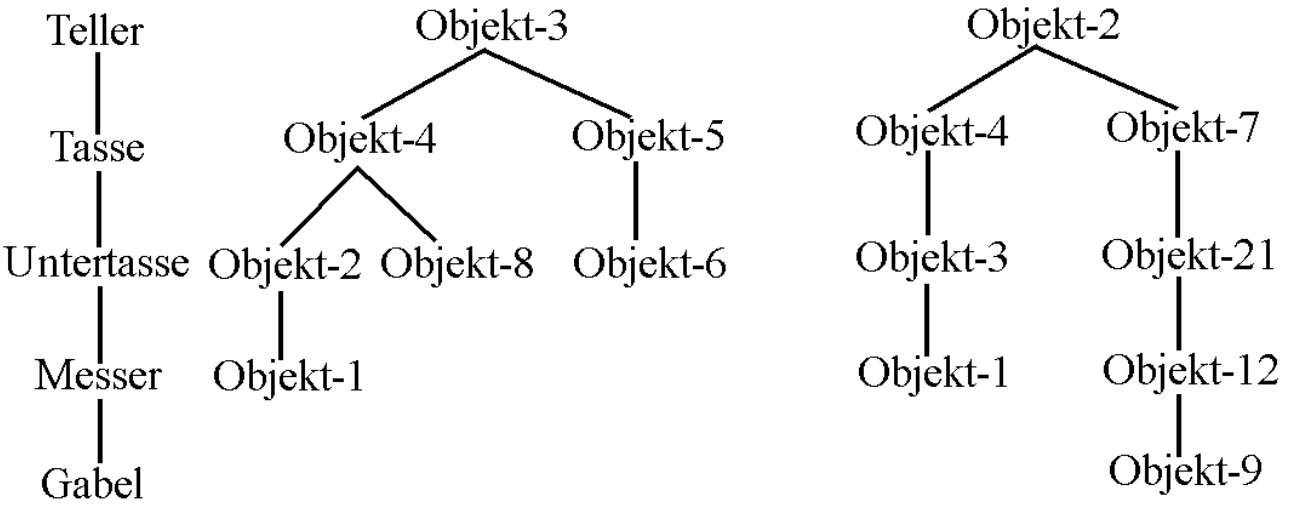

Abbildung 3.14: Beispiel eines Suchbaumes für das Konzept Tischgedeck

Ein Beispiel ist in Abbildung 3.14 zu sehen. Links gibt es eine Darstellung der zu suchenden Konzepte, während rechts daneben jeweils die betreffenden Suchbäume vorhanden sind. Für das Konzept Teller werden zwei Individuen (Objekt-3 und Objekt-2) gefunden und deshalb zwei Suchbäume aufgespannt. Der Teller besitzt dabei die meisten Randbedingungen (Rollen) und wird deshalb zuerst gesucht. Es wird dabei angenommen, dass diese Suchbäume so sortiert sind, dass die Individuen mit der besseren Bewertung weiter links stehen. Nun wird für das Konzept Teller Objekt-3 als Lösung angenommen und die Tasse wird mit der Bedingung, dass Objekt-3 ein Teller ist, gesucht. Es werden wiederum zwei Individuen gefunden. Anschließend wird Objekt-4 als Tasse gewählt und so weiterverfahren bis das Objekt-1 dem Konzept Messer zugewiesen und damit keine Gabel gefunden wurde. Es wird bis zur Untertasse zurückgegangen und Objekt-8 anstatt Objekt-2 gewählt, was wiederum in eine Sackgasse führt. Deshalb wird als Tasse Objekt-5 gewählt. Diese Möglichkeit führt im weiteren auch zu keiner Lösung. Erst die Wahl des Individuums Objekt-2 führt nach dem erfolglosen Durchlauf des linken Zweiges zur Lösung im rechten Zweig.

Dieser Suchalgorithmus ist der in Abbildung 3.11 genannte Suchalgorithmus mit allen Rollen. Ist die Suche erfolglos, dann wird das nicht entdeckte Kontext-Informations-Individuum aus dem Suchraum entfernt.

Dabei wird im Suchalgorithmus mitprotokolliert bei welchem Individuum der Algorithmus am weitesten durch den Suchraum gelangt, und welches Individuum dann nicht gefunden wurde. Um eine eventuell erfolgreiche Ausführung der Suche zu erlangen, wird dieses dann entfernt. Das Schema wird so lange wiederholt bis entweder der Suchalgorithmus erfolgreich beendet wird oder

der Suchraum auf keine Individuen beschränkt wurde, was zum direkten Abbruch des Algorithmus bzw. der entsprechenden Rekursionen führt. Eine Ausnahme ist, wenn das betreffende Individuum ein nicht physikalisch vorhandenen Gegenstand repräsentieren soll, dann wird erst durch einen erneuten rekursiven Aufruf des Verfahrens (welches Abbildung 3.11 wiederspiegelt) ein möglich neues Individuum gesucht und der Suchalgorithmus wiederholt. Erst wenn das rekursive Verfahren ergebnislos ist, wird diese Kontext-Information entfernt.

Diese Wiederholung des rekursiven Verfahrens wird nötig, da zwar der oben beschriebene Suchalgorithmus sicherstellt das ein entsprechendes *Aggregate* gefunden wird, aber es kann zu Problemen kommen, wenn z. B. ein Gedeck in der Nähe eines anderen Gedeckes liegen soll, um z. B. dem Konzept Essen-für-zwei-Personen zu entsprechen. Der Suchalgorithmus stellt sicher das in den jeweiligen Rekursionen jeweils ein Gedeck gefunden wird, aber nicht, dass diese beiden auch in der Nähe liegen. Es ist äußerst komplex diese Information in den Suchalgorithmus mit einfließen zu lassen, da nicht nur die Rolle in der Nähe verstanden werden muss, sondern auch noch z. B. die Definition der Koordinaten, die für die Nähe der Gedecke entscheidend sind, bzw. von welchen Gegenständen des Gedeckes diese übernommen werden. Eine weitere Komplexität ergibt sich, weil darin auch Ausdrücke wie Minimum oder Maximum von diversen Gegenständen erlaubt sind.

Gäbe es den rechten Suchbaum in Abbildung 3.14 nicht, weil dass Objekt-2 nicht vorhanden wäre oder schon verwendet, dann würde das Konzept Gabel entfernt und nach erneutem Durchlauf des Suchalgorithmus ein Tischgedeck ohne Gabel gefunden werden.

Bei erfolgreicher Suche wird, wie nach einem positiven Ergebnis beim Suchalgorithmus für ein vollständiges Konzept, eine Konvertierung der Individuen vorgenommen, die ein falsches Konzept besitzen. Z. B. wurde *Object-1* als Konzept *Plate* von der sensorischen Vorklassifizierung erkannt und nun mittels Suchalgorithmus ermittelt, dass es mit höherem Wissen als *Saucer* interpretiert werden sollte. Deshalb wird im nächsten Schritt *Object-1* als Konzept *Saucer* deklariert und gleichzeitig gelöscht, dass *Object-1* ein Konzept *Plate* ist. Die *Hallucination Machine* arbeitet dabei immer mit der Annahme, dass ein Individuum nur ein Konzept besitzen kann, so dass wie in der Realität ein Gegenstand entweder ein Teller oder eine Untertasse sein kann, aber auf keinen Fall beides.

Dieser Schritt macht die bereits erwähnte *Instance Specialisation* überflüssig, da in diesem Schritt wenn nötig eine Spezialisierung als Konvertierung vollzogen wird. Ein Teller, der aber in einem Konzept gutes-Gedeck ein Teller-aus-Meissner-Porzellan darstellt, wird auch entsprechend konvertiert, so dass eine explizite Spezialisierung unnötig ist.

Als nächstes wird wieder eine *Aggregate Instantiation* ausgeführt. Die Ausführung wurde bereits erläutert, sie lässt allerdings diesmal auch ein zum Teil gefundenes Individuum zu. Es erfolgt nur eine Berücksichtigung der nicht weggefallenen Individuen in der nRQL-*Firerule* aus dem obigen Suchalgorithmus. Aber nicht nur die Konzepte entfallen, auch die eventuellen Rollen wie z. B. Nähe die dieses Individuum referenzieren. Soll z. B. ein Teller in der Nähe einer Untertasse sein und der Teller wird nicht gefunden, dann wird auch die betreffende Rolle Nähe nicht verwendet.
Werden sämtliche Rollen über eine Individuum definiert und dieses nicht gefunden, besteht natürlich die Gefahr, dass keine weiteren Nebenbedingungen bei der Suche existieren. Ein Beispiel wäre auch hier der Teller, der in der Nähe einer Tasse und eines Messers etc. sein soll und somit ein zentrales Objekt darstellt. Wird dieser Teller nicht gefunden, werden alle anderen Rollen auch nicht berücksichtigt. Deshalb sollten auch z. B. die Tasse und das Messer mittels einer Nähe Rolle in Beziehungen gesetzt werden. Das verhindert eine starke Verallgemeinerung des *Aggregates* durch Wegfall eines zentralen Objektes.
Sollte nun das *Aggregate* nur teilweise gefunden werden, stimmt dieses *Aggregate* nicht mehr mit dem entsprechenden Kontext-Informations-*Aggregate* in den nicht instantiierten Individuen überein. Deshalb erfolgt die Löschung der betreffenden Individuen aus dem Kontext-Informations-*Aggregate*, da das Verfahren des *Instance Merging* die Gleichheit der beiden Individuen voraussetzt. Zur Überprüfung dieser Gleichheit wird im *Instance Merging*-Schritt zuerst entsprechend den erwünschten Kontext-Information-*Aggregate* ein passendes Individuum gesucht, welches alle referenzierten Konzepte mit entsprechenden Rollen besitzt. Hierzu wird eine nRQL-Anfrage an den Racer-Server gestellt. Bei Rücklieferung eines Elementes erfolgt noch die Überprüfung der über die Rollen verbundenen Individuen bzw. deren Konzepte auf Gleichheit. Das Kontext-Informations-Element wird dabei als Vergleichsmenge genutzt, d.h. wenn ein über die nRQL-Anfrage gefundenes Individuum mindestens die gleichen Rollen mit den gleichen referenzierten Konzepten besitzt, dann können diese vereinigt werden. Das zu vereinigende Individuum kann weitere Rollen besitzen, wie z. B. weitere *near* Rollen zu anderen Individuen. Diese Rollen sind nicht weiter von Interesse und behindern daher nicht den *Instance Merging*-Schritt. Es gibt einen rekursiven Vergleich der referenzierten Individuen untereinander, um auch deren Vereinigung, bzw. die von denen wiederum referenzierten, zu prüfen. Alle diese zu vereinigenden Individuen werden in einer internen Liste abgespeichert, um eine Schleifenbildung bei der rekursiven Überprüfung zu unterbinden und um aus dieser Liste das Vereinigen auszuführen. Dieser Schritt erfolgt abschließend. Hierbei handelt es sich aber in der Praxis weniger um eine Vereinigung, sondern um eine Löschung der Kontext-Informations-Elemente.

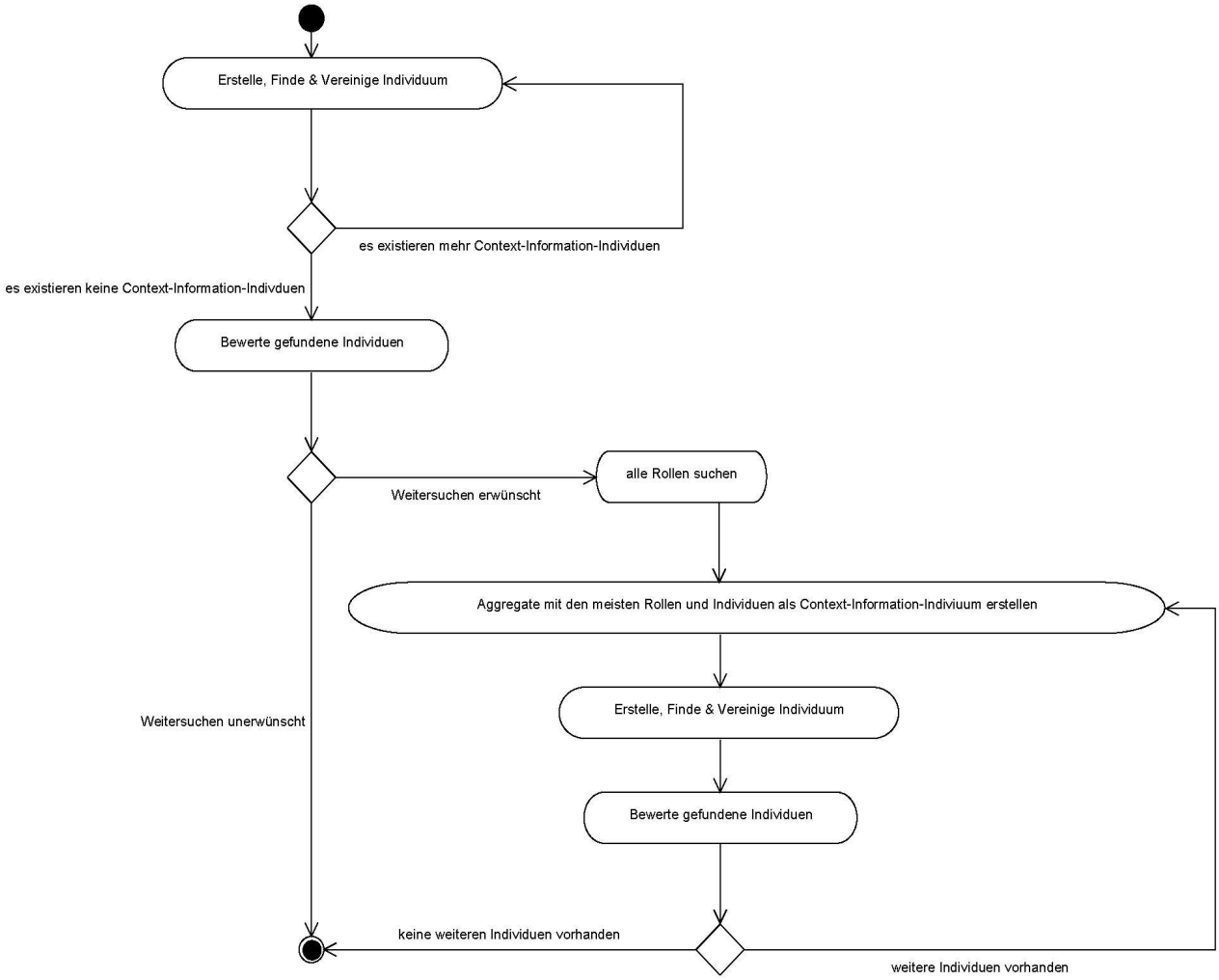

Abbildung 3.15: Darstellung der Funktionsweise der Hallucination Machine als Aktivitätsdiagramm

Zwar wäre eine Vereinigung im Racer-Server mittels des Befehls „same-as“ leicht zu realisieren, allerdings würden die Kontext-Informationen in der ABox bleiben und somit die ABox mit unnötiger Information belasten. Entsprechende Rollen werden ebenfalls gelöscht.

Die im *Instance Merging*-Schritt behandelten Individuen werden nun an ein Individuum des Konzepts *used* über die Rolle *are* gebunden, um erneutes Vereinigen zu verhindern. Ansonsten könnten bei zweifacher Angabe z. B. einer Kontext-Information Teller diese beiden mit demselben, real im Bild existierenden Teller vereinigt werden, und so das Ergebnis verfälschen.

Bei erfolgreichem Abschluss des *Instance Merging* ist der Algorithmus bzw. dieser rekursive Abschnitt mit dem bereits beschriebenen Aufruf beendet.

In Abbildung 3.15 wird nun das gesamte Verfahren der *Hallucination Machine* dargestellt. Zuerst wird ein Kontext-Information-Individuum erstellt bzw. gesucht, wie bereits beschrieben und in Abbildung 3.11 dargestellt. Dieser Vorgang wird für jede Kontext-Information wiederholt. Die Bewertung durch einen Algorithmus drückt aus, wie gut die Kontext-Information ist bzw. wie gut sie erkannt worden ist, und ob oder wie sehr die Interpretationsanfrage, die wie bereits oben beschrieben auch als Kontext-Information behandelt wird, zutrifft. Bei einer schlecht bewerteten Kontext-Information kann ein verarbeitendes System, wie z. B. ein Multimedia Content Management-System, entscheiden, ob diese Information, die scheinbar nicht zutrifft bzw. nicht erkannt wurde ignoriert wird. Dann erfolgt vielleicht wie noch beschrieben eine freie Suche ohne Kontext-Information.

Die Bewertungsfunktion ermittelt für Individuen die mit real existierenden Elementen korrespondieren, also z. B. ein Teller, die entsprechende Bewertung, die die sensorische Verarbeitung ergeben hat. Diese Daten werden aus der in Kapitel 3.2 erwähnten extra gespeicherten Datei entnommen. Für höhere Individuen wie z. B. ein Gedeck gibt es keine entsprechende Bewertung und es erfolgt daher eine Berechnung. Hierfür wird die in den Suchanfragen definierte Gewichtung der Konzepte und Rollen wichtig. Es werden alle entsprechenden Bewertungen der Individuen ermittelt. Vorausgesetzt es sind ebenfalls höhere Individuen wird der Algorithmus rekursiv angewendet und mit der betreffenden Gewichtung aus der entsprechenden Suchanfrage multipliziert. Rollen besitzen keine Bewertung, da diese in der sensorischen Verarbeitung so nicht vorkommen. Die Bewertung erfolgt mit 0 Prozent, wenn sie nicht vorkommen bzw. mit 100 Prozent, wenn sie vorkommen.

Diese Bewertungen multipliziert mit den Gewichtungen der Individuen und der Rollen werden summiert und durch die Summe der Gesamtgewichtungen dividiert. Daraus ergibt sich die Bewertung des Individuums.

Wie bereits erwähnt erfolgt nun eine freie Suche, d.h. eine Suche nach Individuen, die nicht durch Kontext gesteuert werden. Diese Suche ist optional, da sie in einem Multimedia Content Management-System in dem ja nur nach einem betreffenden Individuum gezielt gesucht wird nicht unbedingt sinnvoll ist und nur unnötig Rechenzeit verwenden würde. Bei Wunsch nach dieser freien Suche werden als erstes alle Rollen gesucht, da anschließend eine Bewertung der möglichen Individuen erfolgen soll. Hierfür werden für jede Suchanfrage, die der *Hallucination Machine* übergeben wurde, die entsprechenden Konzepte mit allen Rollen gesucht. Darüber hinaus wird ermittelt, wieviele dieser Konzepte gefunden werden und es erfolgt eine grobe Einschätzung, ob das Konzept vorhanden sein könnte. Auch hier wird wieder in der Anfrage geprüft, ob das gefundene Individuum nicht ein Individuum des Konzepts *used* mit der Rolle *are* verbindet und somit auch noch nicht benutzt wird. Die Suche ist äußerst grob und soll auch nur eine Einschätzung erbringen, ob sich die Anwendung des entsprechenden Algorithmus lohnt.

Der bereits beschriebene bzw. in Abbildung 3.11 dargestellte Algorithmus wird bei einem möglichen Individuum, dessen Suchanfragen am meisten beantwortet oder komplett gefunden wurde, angewendet. Dabei wird dieses Element nun einfach als Kontext-Information angenommen.

Anschließend wird nun dieses gefundene Individuum bewertet, da die obige Suche nur eine grobe Einschätzung darstellt. So kann wie am Beispiel in Kapitel 2.3 dargestellt durchaus ein Tischgedeck am Hamburger Rathaus gefunden werden, allerdings erfolgt eine schlechte Beurteilung. Eine Bewertung, deren Konsequenzen aber hier noch nicht beachtet werden, ist deshalb notwendig, weil ein möglicher Schwellwert, der die schlechte von der guten Klassifikation unterscheidet, etwas zu starr und ungeeignet erscheint.

Die mögliche Einschätzung der Beurteilung wird daher dem darüber liegenden System wie z. B. dem Multimedia Content Management-System überlassen.

Nach dessen Ausführung wird dann wiederum die oben erwähnte Bewertung nach den vorhandenen Individuen durchgeführt und das *Aggregate* gewählt bzw. wenn keines mehr in Frage kommt das Programm beendet.

Kapitel 4

Zusammenfassung und Ausblick

4.1 Zusammenfassung

Im Rahmen dieser Diplomarbeit wurde nach einer kurzen Einführung in die Beschreibungslogik und in den Racer-Server die Szenen Interpretation mittels Beschreibungslogik diskutiert.
Hier wurden die Probleme einer Verarbeitung von Bildern oder Filmsequenzen dargelegt und erklärt, wie eine sensorische Bilderkennung durch höheres Wissen ergänzt werden kann. Es erfolgte die Vorstellung der entwickelten Architektur, die in der Lage ist nach einer Vorverarbeitung in eine geometrische Szenenbeschreibung (GSB) in Bildern oder Filmsequenzen höhere Konzepte wie z. B. ein Tischgedeck zu erkennen und zu bewerten. Dieser Vorgang kann durch zusätzliche Kontext-Informationen unterstützt werden. Weiterhin bietet die sensorische Verarbeitung, die die GSB erstellt, ergänzt durch eine höhere Verarbeitung mit zusätzlichem Wissen (z. B. über die Zusammensetzung und Lage der einzelnen Gegenstände) eine bessere Bilderkennung.
Anschließend folgte die Ausführung wie die Architektur eine freie, nicht durch Kontext-Informationen gestützte Suche durchführen kann, um ohne Zusatzinformationen Konzepte zu erkennen. Dabei sind die in Kapitel 1.2 erwähnten Tests mit erwarteten Ergebnissen absolviert worden. Es ist also möglich mittels der *Hallucination Machine* einfache Szenen (z. B. das Decken eines *Dinner-for-two*) zu interpretieren und auch unter unvollständigen Datensätzen entsprechende Teilkonzepte aufzufinden. Bei den Testdaten führte sowohl die Suche anhand der Kontext-Information sowie die freie Suche zu den gewünschten Resultaten.
Diese Arbeit hat gezeigt, dass bei Kontext-Informations-Individuen, die sehr tief verzweigte Strukturen aufweisen, der Vorgang der Suche unter Umständen komplex wird. Gerade wenn die höheren Konzepte noch Rollenbeziehungen untereinander besitzen, können wie beschrieben mehrfach wiederholte Suchen über die Konzepte ausgeführt werden. Dies kann recht zeitintensiv werden. Bei der freien Suche kann es aufgrund von einer Vielzahl an Suchanfragen bzw. dadurch definierten Konzepten zu zahlreichen möglichen unterschiedlichen Interpretationsmöglichkeiten kommen, dessen grobe Anfangsbewertung bzw. möglicher Verwurf des Konzeptes durch ein höher liegendes System und entsprechender neuer Suche eine große Anzahl an

Kombinationsmöglichkeiten darstellt. Dieses Verarbeitung bewältigt die *Hallucination Machine* nicht mehr in angemessener Zeit. Die Testdaten haben daher einen beschränkten Suchraum mit eingeschränkten Interpretationsmöglichkeiten. Die Verkleinerung des Suchraumes mittels Kontext-Information hat sich auch bei Teilen eines Konzeptes als effektive Unterstützung herausgestellt. Allerdings unterliegt die Architektur den Beschränkungen des Racer-Servers, der bei sehr großen Datenbeständen gerade bei komplexen Suchanfragen mit langen Antwortzeiten aufwartet.

Die in [19] erwähnte Instance Specialisation hat sich in dem vorgestellten Verfahren als überflüssig ergeben, da das Konvertieren eines Individuums in diesem Fall die Spezialisierung ablöst. Durch das benutze Bewertungssystem entsteht ein entsprechend guter Wahrscheinlichkeitswert für die Spezialisierung. So wird ein Objekt, welches mit einer hohen Wahrscheinlichkeit einen Teller repräsentiert, auch eine gute Bewertung für das Konzept Teller-aus-Meissner-Porzellan erhalten, da die wesentlichen Attribute wie z. B. Form, Farbe etc. sich nicht unterscheiden.

Insgesamt hat sich die Bewertungsfunktion, die Konzepte entsprechend ihrer Wahrscheinlichkeit und Gewichtung von unten nach oben beurteilt, als einfaches aber durchaus geeignetes Verfahren erwiesen. Mögliche andere Werte wie z. B. das Einfließen des Kontextes in die Berechnung haben sich als wenig sinnvoll erwiesen, da sie aufgrund geringer Information nur als binärer Wert bearbeitet werden könnten und durch das entsprechende Suchverfahren bereits berücksichtigt werden, wenn auch nicht in der Bewertung.

Dabei hat die Gewichtung der einzelnen Bestandteile den Vorteil, dass einzelne Konzepte als weniger wichtig erachtet werden könnten und so z. B. bei einem Essen-für-eine-Person eher der Teller, die Gabel etc. entscheidend sind als z. B. eine Kerze. Durch eine geringe Gewichtung der Kerze wird ihr fehlen keinen großen negativen Einfluss auf die Bewertung haben und somit trotzdem eine hohe Wahrscheinlichkeit für das Konzept Essen-für-eine-Person aufweisen.

Auch wenn ein oder mehrere Konzepte vernachlässigt werden können, dürfen die Rollen, die die Beziehungen untereinander beschreiben, erst beim Wegfallen der Konzepte nicht mehr berücksichtigt werden. Es hat sich gezeigt, dass ansonsten eine unterschiedliche Bearbeitungsreihenfolge zu verschiedenen Ergebnissen auf der gleichen Datenbasis führt. So wurde bereits das Beispiel angemerkt, dass je nach Ausführungsreihenfolge z. B. ein komplettes Gedeck und ein Gedeck ohne Tasse gefunden werden kann oder ein komplettes Gedeck, wobei die Tasse die Rolle in der Nähe nicht erfüllt, und ein Gedeck ohne Tasse. Diese beiden Resultate würden auch unterschiedliche Beurteilungen in der Bewertungsfunktion erhalten.

Es hat sich dabei auch ergeben, dass es nicht sinnvoll ist, Beziehungen wie z. B. in der Nähe von nur über ein Individuum zu definieren. Wenn z. B. die Tasse nicht gefunden wird, aber der Teller,

der Löffel etc. in der Nähe der Tasse liegen, bleibt beim Wegfall einer nicht gefundenen Tasse ein komplett in Bezug auf die Abhängigkeiten zueinander unbestimmtes Gedeck übrig. Damit würden dem Algorithmus sämtliche für den Menschen intuitive Beziehungen der Gegenstände untereinander vorenthalten. Deshalb ist eine vollständige Definition über alle Objekte eines Konzeptes in Bezug auf deren Zusammenhang unabdingbar.

Außerdem hat sich ergeben, dass die Suche komplex wird, wenn man bei höheren Konzepten, die keine physikalischen Gegenstände repräsentieren, die Übernahme von Attributen aus den Bestandteilen erlaubt und darüber Rollenbedingungen (z. B. in der Nähe von) definiert. Wie bereits erwähnt, ist der Algorithmus dabei nicht optimal und ein Verzicht auf diese Art von Rollenbedingungen beschleunigt das Verfahren erheblich.

Bei einer Szenen Interpretation mittels einer Beschreibungslogik hat sich ebenfalls ergeben, dass Doppelreferenzierungen, wie z. B. zwei Kontext-Informationen, die ein Gedeck beinhalten, auf ein real auf dem Bild existierendes Gedeck vereinigt werden, eine Verfälschung der Halluzination darstellen. Dieses wurde in dieser Arbeit mit einer Bindung über die Rolle *are* an ein Individuum des Konzepts *used* verhindert. Diese Methodik bietet gegenüber einer internen Verwaltung den Vorteil, dass das nicht Vorhandensein dieser Bedingung mit in die Suchanfragen eingebunden werden kann und somit unnötige Ergebnisprüfung vermeidet und die Vorbewertung der freien Suche ermöglicht wird, da diese die Ergebnisse nicht im einzelnen überprüft.

Außerdem kann die Problematik eines Kreuzproduktes bei der Erstellung eines *Aggregates* mit einer anschließenden Überprüfung des Ergebnisses verhindert werden. Wie bereits beschrieben, kann z. B. ein *dinner-for-two* mit *cover-0* und *cover-1* gefunden werden und ein anderes *dinner-for-two* mit *cover-1* und *cover-0*. Das ist natürlich zweimal dasselbe *dinner-for-two*.

4.2 Ausblick

Im Laufe dieser Arbeit haben sich eine Reihe möglicher zukünftiger Schritte ergeben, die die Ausführung komfortabler und effizienter machen.

Zuerst wäre es eine sinnvolle Erweiterung der geometrischen Szenenbeschreibung anstatt der konkreten Objekte, wie z. B. Teller, Untertasse etc. geometrische Objekte, wie z. B. Quadrat, Kreis etc. aus dem Bild zu extrahieren. Eine Erweiterung der *Hallucination Machine* ist dafür jedoch nicht erforderlich, da es nicht von Bedeutung ist ob Teller oder Kreise gesucht werden. Gerade für eine Vielzahl von verschiedenen Gegenständen werden die GSB-Daten sehr groß, da für jedes Individuum die entsprechende Wahrscheinlichkeit für jedes andere Element mitprotokolliert wird (siehe z. B. bereits aus GSB extrahierte Wahrscheinlichkeiten in Abbildung 3.4). Lediglich die Suchanfragen und die TBox und natürlich die GSB-Daten müssen angepasst werden. Eventuell wäre auch die Erweiterung bzw. Anpassung des GSB-Parsers notwendig, da bei einer solchen Abänderung der GSB auch Farben, Muster etc. mitprotokolliert werden könnten.

Eine mögliche Weiterentwicklung der *Hallucination Machine* wäre eine Oberkonzeptsprache, die die Erstellung von Suchanfragen mit entsprechender TBox überflüssig macht. Es müsste sich dabei um eine Sprache handeln, die die in Kapitel 3.3 beschriebenen Einschränkungen nicht besitzt. Daraus würde sich dann automatisch eine TBox erstellen lassen und die entsprechenden Suchanfragen generieren, um die Beschränkungen der Sprache $\mathcal{SHIQ}(\mathcal{D}_n^-)$ zu umgehen. Eine komfortablere Bedienung der *Hallucination Machine* wäre das Ergebnis.

Weiterhin wünschenswert wäre die Möglichkeit der besseren und zielgerichteten Suche in der freien Suche, die statt einer groben Vorklassifikation eine speziellere erlaubt und somit das Auffinden von z. B. Tischgedecken an der Fassade des Hamburger Rathauses (vgl. Kapitel 2.3) vermeidet, und die Unwahrscheinlichkeit nicht erst durch eine schlechte Bewertung ermittelt.

Eine zusätzliche Verbesserung ergibt sich aus der Übergabe der Rollenbedingung zwischen zwei *Aggregates* für den in Kapitel 3.4 beschriebenen Suchalgorithmus.

Wie bereits geschildert, ist diese Optimierung des Suchalgorithmus eine komplexe Angelegenheit, da z. B. bei der Übernahme der entsprechenden Attribute eines Oberkonzeptes, die von verschiedenen Individuen übernommen werden können, durchaus Ausdrücke wie Minimum oder Maximum etc. zugelassen sind. Dieses macht gerade die Bestimmung von mehreren möglichen Rollenverbindungen der zwei bzw. sogar mehrerer *Aggregates* kompliziert. Hierfür ergibt sich aber sicherlich noch Optimierungsspielraum.

Anhang

A. Testbeschreibung

Auf der CD werden neben dem Programm einige Testdaten mitgeliefert, die für die Entwicklung der *Hallucination Machine* äußerst dienlich waren. Ein Gedeck ist dabei definiert durch eine Gabel, die sich links von einem Teller befindet, einem Messer und einem Löffel, welche rechts vom Teller liegen und einer Tasse auf einer Untertasse rechts oberhalb des Tellers. Zur Untertasse gehört ebenfalls ein weiterer Löffel.
In der folgenden Tabelle wird jeweils ein Verzeichnis mit einer Beschreibung angegeben. Die in den Datensätzen enthaltenden Gedecke sind teilweise vollständig bzw. es fehlen einige Komponenten. In den Klammern wird entweder eine korrekte Klassifizierung durch die sensorische Verarbeitung gekennzeichnet oder der realistische Fall mit einer teilweise falsche Einschätzung der Wahrscheinlichkeiten für einige oder alle Gegenstände.
In jedem Verzeichnis sind neben der Datenbasis noch verschiedene Kontext-Informationen, die die unterschiedlichen Definitionen bzw. Suchanfragen ansprechen, und zwei unterschiedlich gewichtete Definitionen vorhanden. Entsprechend der ausgewählten Daten sollen die in der Beschreibung erwähnten Konzepte unter Berücksichtigung der Kontext-Information gefunden werden.

Verzeichnis	Beschreibung
1C-k-k	1 vollständiges Gedeck (korrekt klassifiziert)
2C-k-k	2 vollständige Gedecke (korrekt klassifiziert)
2C-k-k-eine-Fork-fehlt	1 vollständiges Gedeck (korrekt klassifiziert) + 1 Gedeck (korrekt klassifiziert) bei dem eine Gabel fehlt
2C-k-k-eine-Fork+ein-Spoon-fehlt	1 vollständiges Gedeck (korrekt klassifiziert) + 1 Gedeck (korrekt klassifiziert) bei dem eine Gabel und ein Löffel fehlen
2C-k-k-ein-C-nur-mit-Plate-Spoon-Knife	1 vollständiges Gedeck (korrekt klassifiziert) + 1 Gedeck (korrekt klassifiziert) bestehend aus Teller, Löffel und Messer
1C-t-k-k	1 vollständiges Gedeck (teilweise falsch klassifiziert)

Verzeichnis	Beschreibung
2C-t-k-k	2 vollständige Gedecke (teilweise falsch klassifiziert)
2C-t-f-k-ein-Cover-nur-mit-Plate-Spoon-Knife	1 vollständiges Gedeck (teilweise falsch klassifiziert) + 1 Gedeck (teilweise falsch klassifiziert) bestehend aus Teller, Löffel und Messer
2C-t-k-k-ein-C-nur-mit-Plate-Spoon-Knife-Saucer	1 vollständiges Gedeck (teilweise falsch klassifiziert) + 1 Gedeck (teilweise falsch klassifiziert) bestehend aus Teller, Löffel, Untertasse und Messer

Die nachfolgende Tabelle enthält zwei Tests, die die möglichen Rollenverbindung zwischen nicht physikalisch vorhanden Gegenständen überprüfen.

Verzeichnis	Beschreibung
2C-k-k-mit-Rollenverbindung-bei-Dinner-for-two-korrekt	2 vollständige Gedecke (korrekt klassifiziert) und eine Rolle pseudonear zwischen diesen wird gefunden
2C-k-k-mit-Rollenverbindung-bei-Dinner-for-two-falsch	2 vollständige Gedecke (korrekt klassifiziert) und eine Rolle pseudonear zwischen diesen wird nicht gefunden

Literaturverzeichnis

[1] Internetsuchmaschine, http://www.google.com

[2] Microsoft, http://www.microsoft.com

[3] Desktop-Suchprogramme, http://desktop.google.com/ bzw. http://desktop.msn.com/

[4] Earth Observing System (EOS) der NASA, http://eospso.gsfc.nasa.gov/

[5] CoreMedia CMS 2005 - Strategisches Content Management, http://www.coremedia.com/coremedia.aspx/products/cms-2005-news/

[6] H. Müller: Suchen ohne Worte – Wie inhaltsbasierte Suche funktioniert, c't, Bd. 15, 2001, S. 162–167.

[7] Query By Image Content, http://www.qbic.almaden.ibm.com/

[8] Racer Pro, http://www.racer-systems.com

[9] Moving Picture Experts Group (MPEG), http://www.chiariglione.org/mpeg/

[10] NAOS - Ein System zur Beschreibung zeitveränderlicher Szenen, http://kogs-www.informatik.uni-hamburg.de/~neumann/Denkmaschinen-SS-2005/Vortrag-10.5.05.pdf

[11] Dr. rer.-nat. habil. Volker Haarslev, Description Logics: A Logical Foundation of the Semantic Web and its Applications, http://www.cs.concordia.ca/~haarslev/publications/dlsemweb.pdf

[12] Horrocks, I., Sattler, U., and Tobies, S., Reasoning with individuals for the description logic SHIQ., 2000

[13] Dr. rer.-nat. habil. Volker Haarslev, Prof. Dr. rer. nat. Ralf Möller, RACER Pro User's Guide, http://www.racer-systems , April 2005

[14] Frequently Asked Questions about RDF, http://www.w3.org/RDF/FAQ , 2005

[15] DAML-OIL Reference, http://www.w3.org/TR/daml+oil-reference

[16] OWL Web Ontology Language, http://www.w3.org/TR/owl-guide/

[17] Bechhofer, The DIG Description Logic Interface:DIG/1.0, 2000

[18] P.F. Patel-Schneider, B. Swartout, Description LogicKnowledge Representation System Specification from the KRSS Group of the ARPA Knowledge Sharing Effort, November 1993

[19] Prof. Dr. Bernd Neumann, Prof. Dr. rer. nat. Ralf Möller, On Scence Interpretation with Description Logic, http://kogs-www.informatik.uni-hamburg.de/~neumann/publications/scene_interpretation.pdf, 2004

[20] Prof. Dr. Bernd Neumann, T. Weiss, Navigating through logic-based scene models for high-level scene interpretations, In Proc. 3rd Int. Conf. on Computer Vision Systems (ICVS-2003), Springer, 2003 , 212-222

[21] Arbeitsbereiches Kognitive Systeme der Universität Hamburg, http://kogs.informatik.uni-hamburg.de/

[22] Qualitative Analyse zeitveränderlicher Objektmerkmale, Alexander Scharaf, 29.Juli 2004

[23] Java Programmiersprache, http://java.sun.com

[24] A schema for integrating concrete domains into concept languages, F. Baader, P. Hanschke, In Proc. 12th Int. Joint Conf. On Artificial Intelligence (IJCAI-91), 1991, S. 452-457

[25] Dr. rer.-nat. habil. Volker Haarslev, Prof. Dr. rer. nat. Ralf Möller, RACER Pro User's Guide, http://www.racer-systems.com , April 2005, Kapitel Concrete Domains, S. 37-40

[26] Extensions of concept languages for a mechanical engineering application, F. Baader, P. Hanschke, In Proc. 16th German Workshop on Artifical Intelligence (GWAI-92), LNCS. 671, Springer, 1992, S. 132-143

[27] The New Racer Query Language – nRQL, Dr. rer.-nat. habil. Volker Haarslev, Prof. Dr. rer. nat. Ralf Möller, Dipl.Inform. Michael Wessel, February 14, 2005

[28] Lothar Hotz, Prof. Dr. Bernd Neumann, Scene Interpretation as a Configuration Task, März 2005

Erklärung

Ich versichere, die vorliegende Arbeit selbstständig und nur unter Benutzung der angegebenen Hilfsmittel angefertigt zu haben. Die Arbeit ist in gleicher oder ähnlicher Form noch nicht als Prüfungsarbeit eingereicht worden.

Hamburg-Harburg im September 2005