Universität Zürich
Institut für Computerlinguistik

Sommer Semester 2002

Seminar Dr. Kai-Uwe Carstensen

Aspekte der Wissensrepräsentation in der Computerlinguistik

Cyc –

Repräsentation von Commonsense Wissen

Carole Egger
Halden 12
5000 Aarau
Tel. 062 824 30 86
carolee@gmx.ch

1. Einführung

Das Projekt Cyc (der Name ist hergeleitet von ''en*cyc*lopedia'') wurde 1984 von Douglas Lenat an der „Microelectronics and Computer Technology Corporation'' (MCC) in Austin, Texas gestartet, und ist bis heute unter Entwicklung. Cyc ist ein wissensbasiertes System, welches zum heutigen Zeitpunkt etwa 3 Millionen Faustregeln plus etwa 300'000 Terme oder Konzepte aus dem täglichen Leben umfasst. Im Gegensatz zu Expertensystemen, welche nur die Information enthalten, die sie zum Lösen eines Problems in einer beschränkten Domäne benötigen, wurde Cyc entwickelt, um eine grosse Spanne an generellem Commonsense Wissen abzudecken. Ein Experten System, welches Informationen über bakterielle Infektionen enthält, kann zum Beispiel sehr wohl eine solche Infektion genau diagnostizieren. Es kann aber wohl kaum Fragen beantworten, welche sich auf ein Gebiet ausserhalb seiner engen Domäne der bakteriellen Infektionen beziehen. Die Idee war nun, mit Cyc ein System zu konstruieren, das in der Lage sein sollte, nicht nur in speziellen Wissensgebieten zuverlässige Antworten liefern zu können, sondern vielmehr Expertensystemen die Grundlage des Commonsense Wissens zu geben. Expertensysteme sollten so (durch eine gemeinsame Sprache) auf generelles Wissen, welches die Grenzen der Wissensbasis des Expertensystems übersteigt, zugreifen können. Wie generell und unspezifisch das Commonsense Wissen ist, welches Cyc enthält, kommt zum Ausdruck in folgender Aussage von Douglas Lenat, der Cyc so beschreibt:

> You can think of Cyc as a big repository of knowledge - knowledge that is so obvious that it's confusing or insulting to ever say it to someone else. Knowledge like if you're holding a glass of water you should hold it with the open end up. Knowledge like most people sleep at night, and so if you call people in the middle of the night, they will probably be at home sleeping and you'll wake them up. So it's that kind of knowledge - knowledge which we usually don't even tell children because it's so obvious. [1]

Zu Beginn des Projektes war das erklärte Ziel, dass Cyc innerhalb von etwa 10 Jahren genügend Wissen enthalten sollte, um zu wissen, dass z.B. Leute, die gestorben sind, keine Dinge mehr kaufen, dass der Himmel blau ist und dass wenn George Bush in Washington ist, sein linker kleiner Zeh auch in Washington ist. Kurz gesagt sollte Cyc genügend Wissen enthalten, um eine einbändige Enzyklopädie und eine vollständige Zeitung lesen und verstehen zu können. Ab einem gewissen Zeitpunkt sollte Cyc dann fähig sein, von sich aus zu lernen – also neue Inferenzen zu bilden – und mitzuteilen, falls das System in einem gewissen Gebiet nicht genügend Informationen enthält, um ein Problem zu lösen. Wie schwierig sich dies gestaltet wird klar anhand des folgenden Beispiels von Douglas Lenat:

[1] http://auschron.com/issues/dispatch/1999-12-24/screens_feature.html

> We're already able to see isolated cases where Cyc is learning things on its own. Some of the things it learns reflect the incompleteness of its knowledge and are just funny. For example, Cyc at one point concluded that everyone born before 1900 was famous, because all the people that it knew about and who lived in earlier times were famous people. There are similar sorts of errors. But what we're seeing is not so much something that sits quietly on its own and makes discoveries but rather something that uses the knowledge it has to accelerate its own education. [2]

Trotzdem war Cyc 1994 bei einer Demo fähig, gewisse Commonsense Aufgaben relativ gut, wenn auch nicht fehlerfrei zu lösen. Die Idee war, für einen Kunden mit einer sehr grossen elektronischen Bildgalerie eine Anwendung zur Verwaltung seiner Bilder zu schaffen. Cyc wurde dazu übungsweise auf 20 Bilder trainiert, von denen jedes mit einem halben Dutzend Axiomen in der Repräsentationssprache von Cyc beschrieben worden war. Der Befehl „zeig mir jemanden, der relaxt'', brachte ein Bild hervor, auf dem 3 Männer in Badehosen abgebildet waren, die alle Surfbretter vor sich hielten. Cyc machte diese Inferenz anhand von einigen Axiomen, Eigenschaften die für das Photo eingegeben worden waren, und ein paar existierenden Assertionen über erholsame Aktivitäten. Ein anderes Bild zeigte eine junge Frau, die sich am Strand sonnte. Auf die Anfrage hin „Zeig mir jemandem mit dem Risiko für Hautkrebs'', brachte Cyc sowohl das Bild der drei Surfer, als auch das der jungen Frau hervor. Die Logik, die hier gebraucht wurde, war, dass sich am Strand ausruhen auch bräunen impliziert und bräunen steigert das Risiko für Hautkrebs.

Neben der Verwaltung von riesigen Bildgalerien, gibt es viele andere mögliche Anwendungsbeispiele. So wäre Cyc zum Beispiel geeignet, Datenbanken zu bereinigen oder miteinander abzugleichen. Bei einer Datenbank, die alle Daten von Patienten eines Spitals enthält, könnte Cyc zum Beispiel erkennen, dass wenn das Alter eines Mannes, der 11 Kinder hat, mit 19 angegeben ist, es sich dabei mit grösster Wahrscheinlichkeit um einen Tippfehler handelt. Entweder hat der Mann nur ein Kind, oder er ist vielleicht schon 29 oder 39.

Trotz allen Hindernissen wird die Cyc-Technologie seit 1994 von der eigens gegründeten Firma Cycorp kommerziell vermarktet, doch während das Projekt Cyc zu Beginn viel Aufmerksamkeit auf sich zog, ist die Aufregung um Cyc heute ein bisschen abgeflacht. Trotzdem wurden bis jetzt rund 600 Personen Jahre Arbeit in Cyc investiert. Viele genannte Ziele von Cyc sind jedoch immer noch nicht erreicht. So funktioniert das Natural-Language-Interface, das auch ungeübten Usern den Zugang zu Cyc öffnen würde, nur in sehr geringem Masse. Auf Cycorps Website wird im Moment nur gerade ein (abgeschlossenes) Produkt vorgestellt: CycSecure, ein Netzwerk Scanning Prozess, welcher

[2] http://auschron.com/issues/dispatch/1999-12-24/screens_feature.html

die Cyc Technologie nutzt, um die Schritte vorherzusehen, die ein Hacker, Angestellter oder Kunde möglicherweise benutzen könnte, um das Netzwerk einer Firma anzugreifen.

Diese Arbeit soll eine kurze Übersicht sowohl über die Cyc Technologie als auch über Cycs Top-Level Ontologie geben und basierend auf dieser Einführung aufzeigen, welche Bedeutung Cyc für die Computerlinguistik – und speziell für Natural Language Processing – hat. Es werden sodann Probleme und Nachteile von Cyc diskutiert und schlussfolgernd Gründe für die auch heute noch geringe Verbreitung von Cyc genannt.

2. Die Cyc Technologie

Um ein solch gigantisches, wissensbasiertes System zu entwickeln, wurde das Projekt Cyc in fünf hauptsächliche Bestandteile aufgebrochen: Die Cyc Wissensbasis, die CycL Repräsentations Sprache, das Cyc Inferenzen System, die Cyc Interface Tools und die Cyc Applikations Module. In den Abschnitten 2.1 bis 2.3 werden die wichtigsten dieser Bestandteile etwas näher behandelt, eine genaue Besprechung aller fünf Bestandteile ist allerdings im Umfang einer Seminararbeit nicht möglich.

2.1 Die Cyc Wissensbasis

Die Cyc Wissensbasis ist eine formalisierte Repräsentation einer riesigen Menge an fundamentalem menschlichen Commonsense Wissen. Dieses Wissen umfasst Fakten, Faustregeln, und Heuristiken aus dem täglichen Leben und wird durch die formale Sprache CycL dargestellt. Als das Cyc Projekt in 1984 gestartet wurde, wurde zur Repräsentation dieses Commonsense Wissens ein Frame System benutzt, welches prädikatenlogisch formulierte Constraints (einschränkende Bedingungen) für die Slots der Frames enthielt. Frames sind generische Objekte, die, falls keine spezifische Information über sie vorhanden ist, bestimmte Default Werte annehmen. Sie sind Strukturen zur Repräsentation eines Konzeptes und sind immer ein dreier Set bestehend aus Unit, Slot und Entry. Ihr Inferenzen System funktioniert vorwiegend durch Vererbung. Die Anwendung eines Frame basierenden Systems führte im Fall von Cyc, wie David Whitten erklärt, „to a set of increasingly baroque add-ons and work-arounds, such as encoding higher-arity predicates as entries which were tuples, having variant forms of predicates (in which the only difference was the order of the

arguments), and placing more and more stress on frame-oriented editing interfaces to navigate around in the knowledge base.“ [3]

Mit wachsender Grösse des Projektes erschien ein Frame System immer ungeeigneter für Wissensbasen der Dimension von Cyc. Seit 1991 ziehen die Entwickler von Cyc vor, die Wissensbasis als eine „sea of assertions“ zu verstehen, auch wenn dies aufgrund der geringeren Strukturierung der Wissensrepräsentation zu grossen Problemen führt, die in Abschnitt 4 erläutert werden. Das System der Frames wurde aufgegeben und die Wissensbasis besteht nun aus Termen und Assertionen. Das Cyc Team spricht von den Assertionen als „the fundamental unit of knowledge in the Cyc system“ [4]. Jede Assertion selbst besteht wieder aus fünf Teilen, welche zum Beispiel die CycL Formula, die den Content der Assertion beschreibt, einen zugeordneten Wahrheitswert und eine Mikrotheorie, von welcher die Assertion selbst ein Teil ist, beinhalten. Zum heutigen Zeitpunkt enthält die Cyc Wissensbasis rund eine Million – von Hand eingegebene – Assertionen. Die Terme (genauer Konstanten, eine Art von Term) sind das eigentliche Vokabular von CycL und die Assertionen stehen immer in Beziehung zu ihnen. Jede Assertion wird genauso über ihr erstes Argument wie auch ihr letztes gemacht. Sagt man zum Beispiel, dass Klara Andreas Schwester ist, so ist dies eine Aussage über Andrea genauso wie über Klara. Auch das Inferenzen System wurde erweitert und ist nun fähig, logische Ableitungen, Vererbung, automatische Klassifikation usw. durchzuführen. David Whitten erklärt, wie man sich die Cyc Wissensbasis als eine „sea of assertions“ vorstellen kann:

> So one way to visualize the Cyc KB is as a circle filled with assertions; a circular "assertion sea". Above this sea (or outside it, from a two-dimensional perspective) sit all the "constants". Attached to each constant is a bundle of thin wires or strings. The other ends are attached to all the assertions, in the sea, that mention that constant anywhere. Moreover, each of the assertions in the sea can itself be treated as a constant, if you want, and have its own wires reaching to other assertions which mention it [. . .] Inference rules in Cyc can now be thought of as ways of saying that if you have certain assertions in the sea (a set of them, that match a certain pattern) then you are justified in adding a particular new assertion. Each time an assertion is added, wires are automatically strung to all the constants that are mentioned anywhere inside the assertion, and "ripples" of its adding may cause yet other inferences to occur, yet other new assertions to get dumped into the sea, etc. Sometimes one of the new assertions is the answer someone was waiting for, for some problem; sometimes one of the inference procedures reaches a contradiction and has to cope with that.[5]

Jede Assertion wird als Teil einer Mikrotheorie gesehen. Eine Mikrotheorie bezeichnet den eigentlichen Kontext der Assertion. Mikrotheorien fokussieren jeweils auf ein bestimmtes Wissens-Gebiet, einen bestimmten Grad an Detail, ein bestimmtes Zeitintervall oder ähnliches. Die Aufteilung in Mikrotheorien erlaubt Cyc, gewisse Assertionen, die sich

[3] http://www.robotwisdom.com/ai/cycfaq.html
[4] http://www.cyc.com/glossary.html#assertion
[5] http://www.robotwisdom.com/ai/cycfaq.html

widersprechen, aufrecht zu erhalten. So wird es zum Beispiel im Kontext des Büroalltags als sozial nicht akzeptabel betrachtet, wenn jemand während des Arbeitens plötzlich aufspringt und schreit. Im Kontext eines Fussballspiels jedoch wird es als sozial nicht akzeptabel betrachtet, wenn jemand nach einem Tor der eigenen Mannschaft nicht schreiend aufspringt. Das Konzept des schreiend Aufspringens kann also sowohl eine negative wie auch eine positive Konnotation haben.

Im folgenden Abschnitt 2.2 werden wir ein bisschen genauer auf die Repräsentationssprache CycL eingehen.

2.2 Die Repräsentationssprache CycL

Der Repräsentationssprache CycL kommt als Medium der Repräsentation des fundamentalem Commonsense Wissens hohe Wichtigkeit zu. Da die Glanzzeiten des Projektes Cyc vorwiegend in seinen Anfängen lagen, möchte ich hier kurz auf CycL als frame-basierte Wissensrepräsentations Sprache wie vor 1991 eingehen. Neben den vier Arten von Frames in Cyc gibt es zusätzlich noch die CycL Constraint Sprache, die Prädikatenlogik erster Stufe anwendet zur Repräsentation von Disjunktion (z.B. der Satz „Bill ist entweder ein guter Lügner oder ein guter Mensch“), quantifizierenden Aussagen, Negation, oder Relationen zwischen Slot Values. Obwohl das Cyc Team also schon vor 1991 mit der CycL Constraint Sprache auf der Basis der Prädikatenlogik erster Stufe arbeitete, behielt das Team aus Effizienzgründen wo immer möglich die Frame Sprache bei, da sie Inferenzen einfacher und schneller machte. Die unten verwendeten Beispiele zur Illustration der vier Arten von Frames sind direkt aus dem Buch *Representation and Inference in the Cyc Project* von Douglas Lenat und R.V. Guha übernommen.

4 Arten von Frames in CycL

- ***Normale Units***

  Normale Units sind die 90% aller Frames, die real-Welt Konzepte wie Fred, das Set aller Hunde, den Prozess des Spazierens, das Englische Wort *red*, den Staat Texas usw. beschreiben. Im folgenden Frame wird die Unit, die Texas repräsentiert, mit drei Slots dargestellt, jeder mit einem korrespondierenden Wert. Wie Abbildung 1 zeigt, ist der Wert eines Slots einer Unit immer eine Liste von einzelnen, ungeordneten Einträgen:

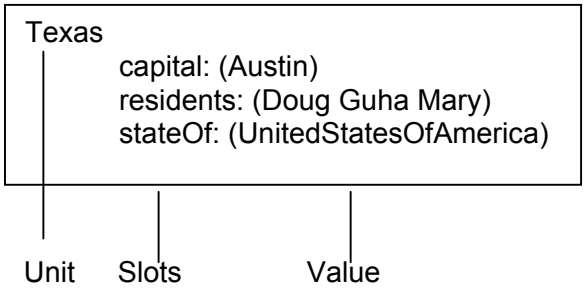

Abbildung 1

- ***SlotUnits***

  SlotUnits sind Frames, die Typen von Slots repräsentieren, wie z.B:

```
residents
    instanceOf: (Slot)
    inverse: (residentOf)
    makesSenseFor: (GeopoliticalRegion)
    entryIsA: (Person)
    specSlots: (lifelongResidents illegalAliens registeredVoters)
    slotConstraints: ((coTemporal u v))
```

  Der Slot *instanceOf* ist gleichbedeutend mit “member of” oder “element of”, dies bedeuted hier also, dass *residents* eine Art Slot ist. Der Slot *inverse* sagt aus, dass x im *resident* Slot von y ist, falls y im *residentOf* Slot von x ist; einfacher also zum Beispiel, dass wenn *Doug* im *resident* Slot der Unit *Texas* ist, dann muss umgekehrt *Texas* im *residentOf* Slot der SlotUnit *residents* sein. Weiter kann man aus dem Slot *makesSenseFor* lesen, dass nur geopolitische Regionen einen *resident* Slot haben sollten. Der Slot *specSlots* gibt an, dass falls x ein lebenslanger Einwohner von y (oder ein illegaler Fremder der in y lebt, oder eine Person registriert zum Stimmen in y) ist, dann kann x auch als Resident von y betrachtet werden. Zu guter letzt werden mit dem Slot *slotConstraints* nun noch Bedingungen für die Unit erklärt. So sollen, falls u ein Resident von v ist, u und v gleichzeitig existieren. Auf diese Weise würde Cyc zum Beispiel wissen, dass Julius Caesar nicht ein Resident von New York sein kann.

- ***SeeUnits***

  SeeUnits dienen als Fussnoten, welche metalevel Information über einen spezifischen Slot einer spezifischen Unit geben. Im folgenden Beispiel verweist der erste Diamant auf eine SeeUnit für einen Slot der Unit *Texas*, welchen sie modifiziert.

Texas
capital: (Austin)
◆residents: (Doug Guha Mary)
stateOf: (UnitedStatesOfAmerica)

SeeUnitFor-residents **.**Texas
instanceOf: (SeeUnit)
modifiesUnit: (Texas)
modifiesSlot: (residents)
◆rateOfChange: (◆)
cardinality: (10000000)

Die *SeeUnitFor-residents .Texas* wird dann selbst wieder modifiziert, indem auf einer nächsten Stufe wieder auf eine SeeUnit verwiesen wird:

SeeUnitFor-rateOfChange**.**SeeUnitFor-residents **.**Texas
instanceOf: (SeeUnit)
modifiesUnit: (SeeUnitFor-residents **.**Texas)
modifiesSlot: (rateOfChange)
qualitativeValue: (Low)

SeeUnits ermöglichen uns also zum Beispiel, in dem *resident* Slot der Unit *Texas* anzugeben, dass der Wert des Slots etwa 10 Millionen Einträge hat, obwohl wir ja nicht wissen, wie diese Einträge alle aussehen. Zudem können wir so auch ausdrücken, dass die Fluktuationsrate der Einwohner von Texas ziemlich gering ist, auch wenn wir nicht wissen, wieviele Einwohner Texas hat. Trotz dieser Vorteile wird anhand des oben genannten Beispieles einer SeeUnit aber auch klar, dass beliebige Verschachtelungen von SeeUnits ziemlich schnell unübersichtlich und undurchsichtig wird.

- ***SlotEntryDetails***

  SlotEntryDetails funktionieren ähnlich den SeeUnits, aber anstatt eine Aussage über einen ganzen Slot einer Unit zu machen, modifizieren sie einen einzigen Eintrag in einem Slot einer Unit. Wenn wir also etwas über Guha (Eintrag im *residents* Slot der Unit *Texas*) sagen möchten – zum Beispiel dass Guha zur seiner eigenen und Lenats Überraschung seit 1987 ein Resident von Texas ist – würde das etwa so aussehen:

SeeUnitFor-Guha residents **.**Texas
instanceOf: (SlotEntryDetailTypeOfSeeUnit)
modifiesUnit: (Texas)
modifiesSlot: (residents)
modifiesEntry: (Guha)
becameTrueIn: (1987)
surprisingTo: (Guha Lenat)
moreLikelyThan:
(SeeUnitFor-PickupTruck ownsA **.**Mary)

Texas
capital: (Austin)
residents: (Doug ◆ Guha Mary)
stateOf: (UnitedStatesOfAmerica)

Das Beispiel der SlotEntryDetails zeigt aber auch, dass es möglich ist, zwei völlig unterschiedliche Slots einander gegenüber zu stellen und zu vergleichen. Wohlbemerkt ist der Fakt, dass Guha seit 1987 ein Resident von Texas ist, zwar überraschend, aber dennoch in Lenats und Guhas Augen viel wahrscheinlicher, als dass Mary einen Pickup Truck besitzt.

Diese Beispiele haben anschaulich gemacht, wie schnell die Cyc Wissensbasis durch Verwendung von Frames kompliziert und unübersichtlich wird. Zudem gibt es, wie weiter oben bereits besprochen, offensichtliche Probleme, wovon die vielen unterschiedlichen Formen von Prädikaten, die sich nur durch die Stellung der Argumente unterscheiden, nur eines sind. Deshalb verwenden die Cyc Entwickler seit 1991 CycL als eine Form von Prädikaten Logik erster Stufe. Hier zum Beispiel die Darstellung der Regel: „Animals sleep at home“:

```
(ForAll ?X (ForAll ?S (ForAll ?PLACE)
        (implies (and
                (isa ?X Animal)
                (isa ?S SleepingEvent)
                (performer ?S ?X)
                (location ?S ?PLACE))
                (home ?X ?PLACE))))) [6]
```

Wie Prädikaten Logik erster Stufe erlaubt auch CycL nun die Anwendung von Allquantifikation, Existenzquantifikation, Konjunktion, Disjunktion und Negation. Neben der Prädikaten Logik erster Stufe enthält CycL auch einige Erweiterungen, welche die Ausdrucksfähigkeit der Sprache vergrössern, auf die ich aber hier nicht weiter eingehen werde. [7] Auf die Probleme der Bevorzugung des „sea of assertions“-Gedanken über dem des frame-basierten Systems und die offensichtlichen Nachteile, die sich aus dieser Änderung ergeben, werde ich in Abschnitt 4 dieser Arbeit eintreten.

[6] http://www.computerworld.com/news/2002/story/0,11280,69881,00.html
[7] Für eine detaillierte Beschreibung von CycL siehe: http://www.cyc.com/cycl.html

2.3 Das Inferenzen Modul in Cyc

Neben generellen logischen Ableitungen (wie Modus Ponens, Modus Tollens und All- und Existenzquantifikation) gibt es in Cyc Inferenz Schematas, die im Hinblick auf eine spezielle Inferenzart optimiert sind. Diese Schematas beinhalten Vererbung, automatische Klassifikation, Erhalt von inversen Links (*residents* vs *residentsOf*), Erhalt von Definitionen (grandparents = parents‘ parents), Erhalt von Abhängigkeiten, finden struktureller Analogien und mehr. In Cyc ist das Inferenzen Modul in eine Anzahl von ’’Features’’ aufgeteilt, welche jeweils für das Verarbeiten einer spezifischen syntaktischen Kategorie von Sätzen verantwortlich sind. Aus Effizienzgründen hat Cyc Dutzende von separaten Inferenz Features. Alles was von jedem spezialisierten Feature getan wird, kann leicht auch vom generellsten Feature getan werden, jedoch nur sehr langsam.

Das Inferenzen Modul kann auch die Reihenfolge, in welcher verschiedene Regeln beachtet werden, bestimmen, indem es die Beziehung *overrides* in Betrachtung zieht. Wenn es zwei Regeln R1 und R2 gibt und Regel R2 Regel R1 überstimmt, dann kann Cyc gerade sogut die Regel R2 zuerst betrachten. Falls es so zu einem Ergebnis kommt, muss die Regel R1 gar nicht mehr betrachtet werden.

3. Cycs top-level Ontologie

Wie bei der kurzen Übersicht über die CycL Repräsentations Sprache werde ich mich auch bei der Einführung in Cyc’s top-level Ontologie an den Status vor 1991 halten. Cyc wurde entworfen als ein System, welches auf Frames basiert, und die nachfolgende top-level Ontologie korreliert eng mit diesem Frame System. Gemäss Cyc besteht die Welt aus einer Anzahl von Dingen, die miteinander in Beziehung stehen. Jede Unit (ein Cyc Frame) repräsentiert ein Ding und jedes Ding wird repräsentiert durch eine Unit. Alle Dinge sind in verschiedene Kategorien (auch Collections oder Sets genannt) aufgeteilt. *Genls* and *specs* sind Prädikate in Cyc, die Kategorien mit ihren jeweiligen Supersets respektive Subsets in Beziehung setzen. Jede Kategorie (ausser *Thing* – das universelle Set) muss also Einträge in seinem *genls* Slot haben. Das heisst, jedes benannte Set ausser *Thing* ist ein Subset eines anderen Sets. Jede Kategorie hat Elemente, *instances* genannt, wovon die umgekehrte Relation *instanceOf* ist. Wichtig zu wissen ist, dass in Cyc ein grosser Unterschied besteht zwischen den zwei Relationen *instances* – womit man die Elemente einer Kategorie betitelt – und *specs* (was die Subsets meint). So ist zum Beispiel die

Beziehung zwischen *Hund* und *Fido* eine *instance*-Beziehung während die Beziehung zwischen *Hund* und *Labrador* eine *specs* Beziehung ist. Am meisten Wichtigkeit wird allerdings der Generalisierungs- und Spezialisierungshierarchie in Cyc und deren jeweiligen *genls*- und *specs*-Beziehungen zugeschrieben. In der Cyc Ontologie gibt es zum heutigen Zeitpunkt weit über 10'000 verschiedene Kategorien. Diese Kategorien in einem Diagram darzustellen, ist praktisch unmöglich, vor allem, da die Kategorien untereinander mannigfach durch Beziehungen verlinkt sind. Die wichtigsten dieser Kategorien jedoch, und ihre Beziehungen untereinander, sind in dem folgenden Ontologie-Diagram dargestellt [8]:

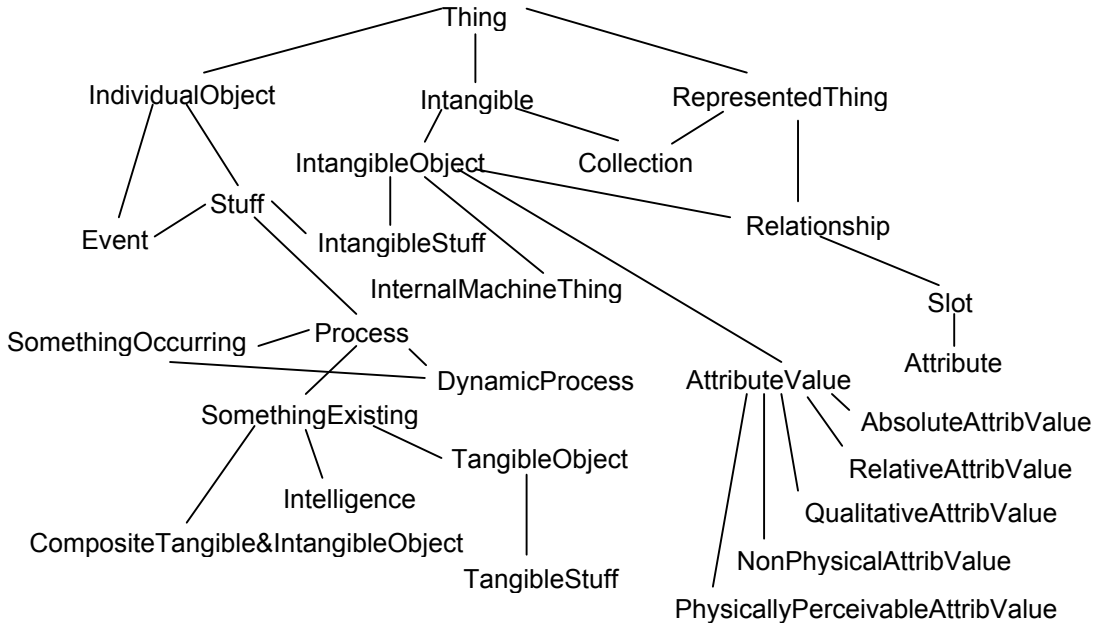

Abbildung 2

Ich werde in den folgenden Abschnitten auf die fundamentalsten dieser Kategorien eingehen, wobei Abbildung 2 als Orientierungshilfe in Cycs top-level Ontologie dienen kann.

3.1 *Thing* – das universelle Set

Thing ist das universelle Set. Sein definierendes Prädikat ist „true“. Wenn man Cyc also fragt, ob irgend etwas eine Instanz von *Thing* ist, so wird die Antwort immer „ja“ sein. Alles andere in der Welt ist in Cycs Ontologie ein Element des Sets *Thing*. *Thing* wird auf drei – anhand von Abbildung 1 nicht sehr leicht verständliche, da asymmetrisch angeordnete - Partitionen aufgeteilt:

[8] Lenat, Douglas B., R.V. Guha, *Building Large Knowledge-Based Systems: Representation and Inference in the Cyc Project* (Reading: Addison-Wesley Publishing Company, Inc., 1990) 172.

- *InternalMachineThing* versus *RepresentedThing*
- *IndividualObject* versus *Collection*
- *Intangible* versus *TangibleObject* versus *CompositeTangible&IntangibleObject*

Innerhalb dieser Partitionen muss jede Cyc Unit eine Instanz von genau einer – und nur einer – der beiden gegenübergestellten Kategorien sein.

3.1.1 *InternalMachineThing* versus *RepresentedThing*

InternalMachineThing beinhalted Daten Strukturen innerhalb der Maschine, auf der Cyc (in diesem Moment) läuft, das heisst Dinge, auf die Cyc direkt zeigen kann, also zum Beispiel Listen, Strukturen, Nummern usw. Gehen wir von einer Unit *TheNumber21* aus, so ist dies die Unit, die die Ganzzahl 21 darstellt. Die Unit *TheNumber21.instanceOf* gehört aber nicht zu der Kategorie *InternalMachineThing*, sondern vielmehr zu einer Kategorie *Represented-InternalMachineThing*, da alle Cyc Units höchstens Repräsentationen solcher Dinge sind. Es gibt also in Cyc gar keine Instanzen von *InternalMachineThing* und auch nur ganz wenige Units, die Instanzen von *Represented-InternalMachineThing* sind. Lenat und Guha erklären in einem Beispiel, weshalb auch *Represented-InternalMachineThing* Units so selten sind (zur Repräsentation einer Unit wird hier, anstatt kursive Schrift zu verwenden, #% vor die Unit gestellt):

> For example, if #%GeorgeBush were a Represented-InternalMachineThing, it would have an actualReferent slot whose entry would have to *be* (literally!) the real flesh-and-blood person George Bush! It's not that GeorgeBush has no actualReferent, it's just that there's no way for the machine to contain a „real pointer" to it. And that's the crux of the InternalMachineThing versus RepresentedThing partitioning. (Lenat/Guha, 177)

RepresentedThing steht sehr weit oben in der *genls*/*specs* Hierarchie, verglichen mit *InternalMachineThing* (siehe Abbildung 2). Das kommt daher, dass, im Gegensatz zu *InternalMachineThing*, *RepresentedThing* alles beinhaltet, was antastbar ist, alle Ereignisse, jedes theoretische Konstrukt (z.B. das Set aller Primzahlen) und viele andere Typen von Dingen. Wir können also sicher sein, dass weder ein *TangibleObject*, noch ein Ereignis in der realen Welt, noch eine Collection (Set aller Sets) ein *InternalMachineThing* sein können.
Das einzige Superset von *RepresentedThing* auf der anderen Seite ist das universelle Set *Thing*.

3.1.2 *Collection* versus *IndividualObject*

Die Kategorie (Set) *Collection* steht für das Set aller Sets, dessen Instanzen all die verschiedenen Kategorien sind, die explizit in Cyc erwähnt werden. *Collections* sind nicht fassbare (*Intangible*), nicht sichtbare Dinge, also mathematische Konstrukte.
Lenat und Guha nennen folgende Beispiele für Instanzen von *Collection* (179):

- Person (the set of all people)
- MetsOutfielder (a smaller set of people; this is a spec of Person)
- DiningAtARestaurant (each instance of this collection is one event)
- Jelly (each instance is a particular piece or portion of this stuff)

Ein *IndividualObject* auf der anderen Seite ist alles, was keine Kategorie ist. Nur ein *IndividualObject* kann Teile (parts) haben und nur eine *Collection* (Kategorie) kann Instanzen (instances), genls und specs haben. So ist zum Beispiel der Kopf Teil des Körpers und eine Instanz der Kategorie *HumanHead* (dem Set aller menschlichen Köpfe). Meistens ist immer nur eines der beiden Sets *Collection* und *IndividualObject* von Wichtigkeit. Lenat und Guha unterscheiden die beiden Units *TheSetOfPartsOfFredsCar* (eine *Collection*) und *TheStructuredIndividualThatIsFredsCar* (ein *IndividualObject*). Hier ein Beispiel mit Slots und Values für die zwei Units (180):

TheStructuredIndividualThatIsFredsCar
    instanceOf: Camaro
    allInstanceOf: Camaro, Chevy, AmericanCar, Automobile, Vehicle, Device, individualObject, Thing
    owner: Fred
    yearOfManufacture: 1988
    originalCostInUS$: 15000
    parts: Fred'sCar'sSteeringWheel, Fred'sCar'sLeftFrontTire...
    setOfParts: TheSetOfPartsOfFredsCar

TheSetOfPartsOfFredsCar
    instanceOf: Collection
    allInstanceOf: Collection, Thing
    genls: CamaroPart
    allGenls: TheSetOfPartsOfFredsCar, CamaroPart, CarPart, DevicePart, ManufacturedProduct, PhysicalObject, IndividualObject, Thing
    setOfPartsOf: TheStructuredIndividualThatIsFredsCar

Während die *Collection* also „members" hat, hat das *IndividualObject* „parts". Wann man nun aber etwas mehr als eine *Collection* oder mehr als ein *IndividualObject* auffasst, erklären Lenat und Guha so (181):

> If many key relationships hold among the parts, we usually focus on the thing as an IndividualObject. If the parts don't really interrelate much, we usually focus on the thing as a Collection. Occasionally, we might want to flip back and forth among the two views, in which case we maintain both units, but that's a rarity!

Zwei separate Units, die in der Cyc Wissensbasis tatsächlich existieren, sind *AmericanPublic* und *AmericanPerson*. *AmericanPublic* ist ein *IndividualObject*, also eigentlich ein riesiges Individuum mit hunderten von Millionen komplexen, miteinander in Beziehung stehenden Teilen. *AmericanPerson* ist eine *Collection*, ein „Set of people", also eine Kategorie.

3.1.3 *Intangible* versus *TangibleObject* versus *CompositeTangible&IntangibleObject*

Wie bei den Partitionen oben ist auch bei dieser Partition jede Cyc Unit eine Instanz von genau einem (und nur einem) der drei Sets: *Intangible*, *TangibleObject* oder *CompositeTangible&IntangibleObject*. *Intangible* ist alles, was keine Masse (oder präziser keine Mass-Energie) hat. Beispiele für *Intangible* sind (nach Lenat und Guha, 182):

- Person (the set of all people, eine Kategorie kann niemals gewogen werden)
- TheEarthMoonSunTriangle (wir können diesen zwar messen, aber niemals wägen)
- TheCycProgram (ein Programm hat keine Mass-Energie)
- Plus (mathematische Funktionen und Prozeduren haben keine Masse)
- DiningInARestaurant (dies ist der generelle Prozess, es ist also eine Kategorie und kann keine Masse haben)

Als *TangibleObject* gilt alles, was Mass-Energie hat. Es gibt keine Kategorie, die Masse hat, also muss ein *TangibleObject* immer auch ein *IndividualObject* sein. Ein *TangibleObject* ist zum Beispiel ein Stein, ein Baum, ein Auto oder eine Person. Während die ersten drei Beispiele keine Aspekte der *IntangibleObjects* enthalten, können einer Person durchaus auch solche Aspekte zugeschrieben werden. So ist zum Beispiel alles, was den Geist und die Handlungskraft einer Person betrifft, vielmehr den *IntangibleObjects* als den *TangibleObjects* zuzuordnen. Lenat und Guha geben weiter das Beispiel eines Landes wie Frankreich, welches zum einen zwar Masse, Einwohner, Bodenschätze usw. hat, zum anderen aber auch Proklamationen ausrufen oder Dinge verkaufen und kaufen kann (183). Für solche Fälle brauchen wir also eine Kategorie, die sowohl fassbare (*tangible*) und nicht fassbare (*intangible*) Aspekte hat. Lenat und Guha schlagen vor, eine Kategorie *CompositeTangible&IntangibleObject* zu benutzen. Diese Kategorie hat die zwei wichtigen Slots *physicalExtent* und *intangibleExtent*. Sie besteht also aus einem *Intangible* und einem *TangibleObject*. Eine Person würde zum Beispiel in diese Kategorie fallen, die sowohl das physische Ausmass wie auch den nicht fassbaren Teil darstellen kann. Für die Unit GeorgeBush geben Lenat und Guha folgende Slots und Values an (183):

GeorgeBush
instanceOf: Person...
allInstanceOf: CompositeTangible&IntangibleObject...
physicalExtent: GeorgeBushsBody
intangibleExtent: GeorgeBushsMind
interestingness: High

GeorgeBushsBody
instanceOf: HumanMaleBody
allInstanceOf: TangibleObject...
physicalExtentOf: GeorgeBush
weightInLbs: 200
speed: Low
interestingness: low

GeorgeBushsMind
instanceOf: HumanMind
allInstanceOf: Intangible...
intangibleExtentOf: GeorgeBush
speed: Moderate
interestingness: Moderate

Weiter nennen Lenat und Guha den folgenden Grund für die Existenz einer Kategorie der Art von *CompositeTangible&IntangibleObject* (184):

> [. . .] we might have a slot (like interestingness) that has *different* values for his tangible part (his body), his intangible part (his mind), and the composite that is „the whole person". This may seem like a luxury, but it is a distinction humans do easily make and therefore must represent; it is a distinction that we occasionally *need* to make.

Weitere Beispiele für *CompositeTangible&IntangibleObjects* sind Firmen, Länder usw. Eine Firma hat einen physischen Aspekt (zum Beispiel das Gebäude, oder die Körper der Angestellten), aber auch einen nicht fassbaren Aspekt (wie ihre Beziehungen mit anderen Firmen). Ihre nicht fassbare Seite macht eine Firma also zu einem „intelligent agent: it's capable of reacting to novel situations, learning from experience, communicating with other intelligent agents, and so on" (Lenat/Guha, 185). Dies kann auf alle Organisationen und eben auch Länder ausgedehnt werden. Die Kategorie *CompositeTangible&IntangibleObject* ist aber noch viel ausgedehnter. Auch Bücher, Filme, Hörspiele usw. gehören der Kategorie *CompositeTangible&IntangibleObject* an, da sie immer auf den physischen Teil (z.B. Buch, Videokassette, Audiokassette) und den nicht fassbaren Teil (z.B. Bedeutung, Inhalt, Thema usw.) aufgeteilt werden können.

3.2 *Stuff* versus *IndividualObject*

Am anschaulichsten erklärbar ist die Kategorie *Stuff* folgendermassen: Wenn man etwas der Kategorie *Stuff* zerschneidet, erhält man lauter kleine Stücke von *Stuff*. Ein Beispiel ist Wasser oder Butter, aber auch etwas wie Holz, der Prozess des Spazierens oder Energie. Spazieren ist auch ein Typ von *Stuff*, denn wenn man eine Stunde Spazieren in zwei teilen könnte, dann könnte jede dieser Hälften als ein ganzer Spazier-Event betrachtet werden.
Es ist wichtig, *Stuff* und *IndividualObject* zu unterscheiden. Wenn wir eine Axt nehmen und damit *Table255* (ein *IndividualObject*) zerschmettern, so ist keines der Teilchen mehr ein Tisch, noch sind die Teilchen kleine Tische. Wenn *Table255* aber aus Holz gemacht ist und als Instanz der Kategorie *Wood* notiert ist, dann sind nach dem Zerhacken alle Teilchen immer noch Instanzen des Sets *Wood*. Ein Komplikation ergibt sich allerdings, wenn man bedenkt, dass jedes Stückchen von *Stuff* (zum Beispiel ein bestimmtes Stückchen Butter auf dem Tisch) auch wieder ein *IndividualObject* ist. Auf der anderen Seite besteht auch jedes Indivduum aus irgend einer Substanz (*Stuff*). Es besteht also eine enge gegenseitige Beziehung zwischen *Stuff* und *IndividualObject*. Lenat und Guha meinen dazu:

> [. . .] Rather surprisingly, the two properties are extensionally equivalent. [. . .] We still choose to distinguish between them since they have different intensional descriptions. [. . .] There are certain properties that are *intrinsic* in that if an individual has them, „slices" of that individual also have them (at least as a default), while there are other properties that are extrinsic (i.e., the parts don't inherit that property from the whole.) The notion of intrinsicness is closely related to that of substances in the following way. Consider a particular table made entirely of wood - *Table103*. It inherits various default properties from *Wood*, which is the kind of substance it's an instance of (properites such as *density, flash point,* etc.) and it inherits other properties from *Table,* which is the kind of individual object it's an instance of (properties such as *number of legs, cost, size,* etc.) The former properties are intrinsic, the latter are extrinsic. This is no coincidence! We have noticed that an object (typically) inherits its intrinsic properties from whichever instances of *SubstanceType* it's an instance of, and it inherits extrinsic properties from whichever instances of *ObjectType* it's an instance of. [. . .] This also illustrates the importance of having collections of collections - we could actually dispense with the concepts *Substance* and *IndividualObject* (since they're co-extensional), but we can't do without *SubstanceType* and *ObjectType.* [9]

3.3 *Event* versus *Process*

Die Kategorie *Event* steht für das Set aller Dinge, die ein temporales Ausmass haben (*startingTime*, *endingTime*, *duration*), also auch für jene Dinge, die durch zeitliche Relationen miteinander in Beziehung stehen (wie zum Beispiel „vorher", „nachher"; „während" usw.). Die

[9] http://www.cyc.com/tech-reports/act-cyc-134-90/sectionstar3_5.html

Kategorie *Process* verhält sich zu der Kategorie *Event* wie die Kategorie *Stuff* zu *IndividualObject*. Wenn wir uns jemanden vorstellen, der während 20 Minuten in einem Musical singt, und uns dann fragen, was diese Person in der fünften Minute getan hat, so ist die Antwort „sie hat gesungen". Jedes Zeit Stück aus einem *MusicalSinging* Event ist ein kleinerer *MusicalSinging* Event. *MusicalSinging* ist also ein Prozess, da man irgend ein zeitliches Stück aus dem gesamten Prozess herausschneiden kann und das herausgeschnittene Stück wieder ein Prozess ist. Stellt man sich aber nun, wie bei Lenat und Guha als Beispiel erwähnt (187), eine Unit *FredAndBobPlayingTwoGamesOfTennis-RightAfterLunch* vor, so ist dies zwar ein *EventType* (die *Collection* aller Typen von Events), aber sicher nicht ein *ProcessType* (die *Collection* aller Typen von Prozessen). Schaut man sich nämlich, wie zuvor beim Beispiel mit der Musical Sängerin, die fünfte Minute dieses Tennis Spiels an, so ist dies zwar immer noch eine Instanz von *PlayingTennis* aber nicht mehr eine Instanz von *PlayingTwoGamesOfTennis.*

4. Kritik

Nach diesem kurzen Einblick in die Idee, die Technologie und die Ontologie von Cyc möchte ich mit einem Zitat David H. Freedmans den Abschnitt der Kritik beginnen:

> [Douglas Lenat insists] that Cyc will change the way we live and work. Schools will use Cyc to provide one-on-one tutoring to students, he says.... Cyc will make scientific discoveries, apply justice, and even counsel unhappy couples. As Cyc continues to grow, he predicts, pieces of it will be stored in computers around the world, its contents made available through phone lines and radio waves. Cyc's intelligence, he claims, will flow like electricity through a gigantic, ubiquitous knowledge grid.[10]

Das Zitat stammt aus dem Jahre 1990, also nur rund sechs Jahre nachdem das Projekt Cyc gestartet wurde. Heute, fast zwanzig Jahre nach Start des Projektes, ist die Erreichung dieser Ziele, die Lenat für Cyc gesetzt hatte, immer noch in weiter Ferne. Gründe dafür gibt es viele. Eine Rolle spielt sicherlich das „General Knowledge Problem" oder „Commonsense Knowledge Problem". So wurde erstens das Wissen, das implizit in den Menschen vorhanden ist, massiv unterschätzt, und zweitens stellt sich immer wieder die Frage, wie denn nun dieses Wissen in künstlichen Systemen repräsentiert werden soll. Christopher Locke greift in einem polemischen Essay ein weiteres, generelles Problem auf. Was nämlich genau ist Commonsense Wissen, und wer entscheidet, welche Art von Wissen den Namen

[10] David H. Freedman, *Common Sense and the Computer* (Discover, August, 1990); http://www.panix.com/~clocke/ieee.html

Commonsense Wissen wirklich verdient? In seinem Essay meint Locke:

> One way out of this dilemma is to assume that the knowledge in a system is "common knowledge" precisely because it has been put into the system by "knowledgeable experts." [...] The politically charged correlate of this assumption is that experts are those who have the necessary power to define or affect the contents of such systems. If the representation of this knowledge depends on a formal language which most of the community of practitioners cannot interpret (e.g., Lisp, Prolog, C), then the larger community will have been effectively disenfranchised. [...] Also, the knowledge on which many expert systems are based is more the opinion of a single, and sometimes questionable, "expert" than widely accepted fact. That is to say, it is not "common" knowledge at all. This problem has long been acknowledged by AI system designers and theoreticians. Knowledge is not static -- "it depends..." as we say. Statistical information can be considered fact because there is a truly common foundation of well articulated axioms on which it is based. No such universal conventions underpin the kind of knowledge which is conceived and transmitted via language. Though the Cyc project at MCC is attempting to create just such an axiom base, it may be deceptively generous to say that the challenge is enormous. [11]

Was Locke hier anspricht ist der berühmte „knowledge acquisition bottleneck“. Das Problem ist, dass das Cyc Interface für die Dateneingabe so kompliziert ist, dass eigentlich nur ausgewählte Philosophen, Anthropologen, Logiker und Informatiker fähig sind, dem System Wissen „einzuflössen“. Dies ist also ganz klar Wissen von Leuten aus einer beschränkten Domäne und kann niemals das „common sense“ Wissen in der Welt verkörpern. Leuten ausserhalb dieser Domäne, die relevantes und wertvolles Wissen zu der Cyc Wissensbasis beisteuern könnten, steht der Zugang zu Cyc nicht offen, einerseits wegen dem anwender-unfreundlichen Interface, andererseits, weil Cyc auf kommerzieller Basis funktioniert. Das Cyc Team um Douglas Lenat hat dieses Problem erkannt und arbeitet seit längerer Zeit an einem Natural Language Interface, welches mit einem System genannt OpenCyc ermöglichen würde, Cyc öffentlich zur Wissenseingabe frei zu geben. Doch gerade für Natural Language Processing ergeben sich grosse Anforderungen an ein wissensbasiertes System wie Cyc. Besondere Wichtigkeit kommt dabei der Auswahl einer geeigneten Wissensrepräsenation zu.

So kam das Cyc Team auf der Suche nach einer Wissensrepräsentation, die sich für eine Wissensbank der Grösse von Cyc eignet, von der eines frame-basierenden Systems in den Anfängen des Projektes zu einer Wissensrepräsentation basierend auf der Prädikatenlogik erster Stufe, in der die Wissensbasis als eine „sea of assertions“ gesehen wird. Welche Probleme diese Repräsentation besonders für die Sprachverarbeitung mit sich bringt, evaluieren Mahesh, Nirenburg, Cowie und Farwell (MNCF) in Ihrem Essay „An Assessment of Cyc for Natural Language Processing“ (1996), der vorwiegend auf die Anwendung von Cyc für die „word sense disambiguation“ und „coreference resolution“ eingeht. Eine Anwendung wie Cyc ist für Natural Language

[11] http://www.panix.com/~clocke/ieee.html

Processing (NLP) sehr interessant, da wissensbasierte Methoden potentiell eine effizientere Lösung zur „word sense disambiguation“ anbieten als als andere Methoden wie zum Beispiel korpus-basierende statistische Methoden (MNCF 4). NLP, als besonderes Problem innerhalb des Information Processings, benötigt jedoch riesige Mengen an breitem Weltwissen. In welchem Masse nun Cyc geeignet ist, dieses Weltwissen für NLP zur Verfügung zu stellen, wird durch ausgewählte Anfragen an Cyc getestet. Mahesh, Nirenburg, Cowie und Farwell konzentrieren sich vorwiegend auf die zwei Grundprobleme „word sense disambiguation“ und „coreference resolution“. Sie legen zuerst die Anforderungen fest, die eine Wissensbasis erfüllen muss, um für NLP überhaupt brauchbar zu sein. So führen sie die Begriffe der Tiefe („depth“) und Breite („breadth“) als Massstab der semantischen Abdeckung zusätzlich zu der Grösse („size“) ein. Die Anzahl der Informationen (Tiefe) und die Typen von Informationen (Breite) die in jedem Element einer Wissensbasis enthalten sind, sind genauso wichtig wie die totale Anzahl der Elemente (Grösse der Wissensbasis) (MNCF 10). Vom Cyc Team wird Cyc’s Grösse mit einer Anzahl von rund 3 Millionen Faustregeln plus etwa 300'000 Termen oder Konzepten aus dem täglichen Leben angegeben. Dabei unterscheidet Cyc allerdings nicht zwischen Konzepten und Instanzen von Konzepten. So werden zum Beispiel in Cyc eigentliche Instanzen von Konzepten wie AlabamaState, CityOfMadrasIndia, CorazonCAquino, oder SoutheasternUniversityOfTheHealth-SciencesCollegeOfOsteopathicMedicine als Konzepte gezählt. Auch die meisten Prädikate in Cyc sind auf der Ebene der Instanzen – und nicht auf der der Konzepte - definiert und untersuchen so Instanzen von Konzepten und die Beziehungen der Instanzen untereinander, was ein generelles Problem ist in Cyc. Zudem gibt es einige eigenständige Konzepte in Cyc, die eigentlich leicht aus anderen Konzepten zusammengesetzt werden könnten; AvailableForDating, StandingWithLegsCrossed und SwimmingTheLengthOfOlympicPool sind nur einige dieser Fälle. Mahesh, Nirenburg, Cowie und Farwell begreiffen Instanzen von Konzepten nicht als eigenständige Konzepte und geben so die Anzahl der Konzepte in Cyc mit etwa 10‘000 bis 12'000 an.

Obwohl diese Konzepte wirklich eine grosse Breite an Domänen abdecken, gibt es unausweichliche Lücken in der konzeptuellen Abdeckung. So konnten zum Beispiel keine Konzepte für „playpen“, „sacrifice“, „devotion“, „adopt“, „beg“; „tea bag“ etc. gefunden werden (MNCF 15). Doch noch problematischer als eine Anzahl von fehlenden Konzepten in der Repräsentation ist, dass unter den Konzepten, die repräsentiert werden, eine nicht-uniforme Auswahl deren Instanzen besteht. Mahesh, Nirenburg, Cowie und Farwell meinen dazu:

> Although there are many instances of concepts in Cyc, the gaps in coverage seem to be much larger in instances. In any category, such as languages, cities, religions, companies, etc., many missing instances can be listed. For example, AmericanAirlines is the only airline listed and AlamoAutoRentalCompany is the only car rental company listed. Although it was perhaps never the intention of Cyc to cover a large set of instances within the Cyc knowledge base, the coverage is rather non-uniform among the instance that are included in Cyc. (15)

Dies führt uns wieder zurück zu den Begriffen der Tiefe und der Breite zusätzlich zu der Grösse als Massstab der semantischen Abdeckung. So ist zwar die Grösse der Wissensbasis, also die totale Anzahl der Elemente, von Bedeutung, gleichzeitig aber ist die Anzahl der Informationen (Tiefe) und die Typen von Informationen (Breite), die in jedem Element enthalten sind von grosser Wichtigkeit. Die Anzahl der Elemente in Cycs Wissensbasis lässt also noch lange nicht auf eine genügende semantische Abdeckung schliessen. So enthält Cyc erwiesenermassen Konzepte, deren Instanzen weder in genügender Tiefe noch in genügender Breite definiert sind, woraus sich eine unvollständige semantische Abdeckung ergibt. Als notwendige Kriterien zur Messung der semantischen Abdeckung – also sowohl der Tiefe, Breite als auch Grösse einer Wissensbasis - nennen Mahesh, Nirenburg, Cowie und Farwell folgende (11):

- Die Anzahl der Eigenschaften oder Links, die für ein individuelles Konzept definiert worden sind,
- die Anzahl der Typen von nicht-taxonomischen Beziehungen zwischen Konzepten,
- die durchschnittliche Anzahl der Links pro Konzept,
- die Typen von Wissen, die vorhanden sind (z.B. „defaults“, „selectional constraints“, „complex events“ usw.),
- die Anzahl der Einträge in einem Lexikon verglichen mit der Anzahl der Konzepte in der Ontologie,
- und schliesslich die totale Anzahl der Konzepte in der Ontologie.

Was Mahesh, Nirenburg, Cowie und Farwell unter anderem hier erwähnen, und worauf wir noch nicht eingegangen sind, ist das Lexikon, das vorhanden sein muss, bevor Cyc überhaupt zur Lösung von semantischen Problemen benutzt werden kann. Ein solches Lexikon muss Wörter in der gewünschten Sprache (in Cyc: Englisch) mit Konzepten in Cyc mappen. An Cycs Lexikon für Englisch wird immer noch gearbeitet, es ist also noch lange nicht vollständig. Mahesh, Nirenburg, Cowie und Farwell zählen folgende Anforderungen auf, die erfüllt werden müssen, um ein Lexikon zu erarbeiten, indem Wörter mit Cyc gemappt werden (11):

- es muss eine genügende Anzahl Konzepte vorhanden sein,
- die Konzepte müssen einfach zu finden sein,
- es muss genügend Wissen über ein Konzept und über seine Beziehungen mit anderen Konzepten vorhanden sein,
- Wissen über ein Konzept muss einfach zugänglich sein; es muss einfach sein, alles, was über ein bestimmtes Konzept bekannt ist, zu sehen,
- die Wissensrepräsentation muss genügend aussagekräftig sein, um Wortbedeutungen zu ermöglichen, die Bedeutung von anderen Wörtern zu erweitern oder einzuschränken.

Genau hier liegt ein Haupt-Problem von Cyc. Damit das Wissen in Cyc brauchbar wird, muss die richtige Information, um ein bestimmtes Problem in einem bestimmten Kontext effizient zu lösen, einfach zugänglich sein. In diesem Sinn führte der Wechsel von einem frame-basierenden System zu dem eines logik-basierenden Systems zu einer grossen Verschlechterung. In der logik-basierenden Version von Cyc ist die Repräsentation nämlich nicht „object-oriented" (MNCF 31), was heisst, dass das Wissen, das über ein bestimmtes Konzept verfügbar ist, nicht an einem Ort repräsentiert wird und deshalb nicht in einer vernünftigen Weise zugänglich ist. Mahesh, Nirenburg, Cowie und Farwell drücken sich klar aus (20):

> A basic problem of usability in Cyc (the current logic-based versions, not the frame-based versions of the eighties) is that **every piece of knowledge present in Cyc about a given concept is not accessible** at all. In fact, we wonder how knowledge acquisition in Cyc could be accomplished without such access. How could a new concept be added to the taxonomy or a relation added between a pair of concepts without seeing what knowledge is currently there (both explicitly and implicitly through inheritance and inference)?

Da man nicht wirklich weiss, welches Wissen eigentlich in Cyc enthalten ist, ist auch nicht nachvollziehbar welches Wissen nicht in Cyc enthalten ist. So könnte man „easily spend a day exploring Cyc and trying to find answers to sub-questions recursively and still not be able to answer the relevant questions one way or the other" (MNCF 22). Die Nicht-Uniformität von Cycs Wissen lässt sich auch am folgenden Beispiel erkennen: Cyc kennt zwar TheYear1984 und TheYear1986, es lässt sich aber merkwürdigerweise kein Eintrag für TheYear1985 finden. Während das frame-basierende System noch Beziehungen zwischen einem gegebenen Konzept und anderen Konzepten in der Wissensbasis finden konnte, scheint es Cyc in seiner logik-basierenden Form nicht mehr möglich zu sein, alle Prädikate zu finden,
die ein Konzept mit anderen Konzepten in Beziehung setzen. Der Grund dafür liegt in der fehlenden Struktur der „sea of assertions" Repräsentation. So ist das Wissen, das über ein Konzept vorhanden ist, in individuellen Assertionen überall in der riesigen „sea of assertions"

verteilt, ohne klare Strukturierung, die alles an einem Ort (eben z.B. einem Frame oder Objekt) zusammenbringt. Wissen über ein beliebiges Konzept jedoch, das nur lokal repräsentiert wird, „is not sufficient either for using that concept in an application or for acquiring further knowledge (about the same concept or other concepts in the ontology); we must be able to see and quickly digest all of the *inherited or otherwise inferred knowledge* about a concept“ (MNCF 33). Stellen wir zum Beispiel Anfragen an Cyc wie „Are Concept1 and Concept2“ related (through some unknown relation)? If so, how closely?“ (MNCF 13), so sollte Cyc die Beziehungen finden, die in den kürzesten Pfaden zwischen Konzepten resultieren. Handelt es sich bei diesen Beziehungen aber nicht um direkte Beziehungen zwischen Paaren von Konzepten, die in einem Text erscheinen – wovon man in dem grössten Teil der Fälle ausgehen kann – so ist es praktisch unmöglich in der „sea of assertions“ auf die richtige Information zugreifen zu können. Die für das NLP so notwendigen „selectional constraints on relationships between concepts“ (MNCF 32) sind in Cyc – in der heutigen Form - nicht vorhanden. Zwar enthält Cyc zur Einschränkung der möglichen Pfade die sogenannten Mikrotheorien. Es ist allerdings sehr unwahrscheinlich, dass wir natürlichsprachliche real-Welt Texte einer Mikrotheorie zuteilen und so alle anderen Mikrotheorien ausschliessen können

Es lässt sich also zusammenfassend sagen, dass Cyc zwar eine grosse Menge an potentiell sehr brauchbarem Wissen enthält, dieses Wissen aber leider weder in einem ganz Cyc-basierenden System noch in einem System, indem Cyc zur Leistungssteigerung anderer Methoden ergänzend gebraucht wird, einfach zur Problemlösung angewandt werden kann.
Mahesh, Nirenburg, Cowie und Farwell schlagen abschliessend in ihrem Essay vor, dass, anstatt eine Ontologie wie bei Cyc neu und eigenständig zu entwickeln, auf bereits bestehende „general-purpose“ (MNCF 35) Ontologien zugegriffen werden sollte. Solche Ontologien sind oft erstaunlich ähnlich und könnten so effizient zusammengefasst werden. Weiter verneinen Mahesh, Nirenburg, Cowie und Farwell die Möglichkeit einer einzigen, korrekten Ontologie und schlagen vor, einer Wissensbasis mehrere top-level Klassifikationen überzuordnen (MNCF 35). Top-level Ontologien unterscheiden sich zwar oft sehr von einander, es scheint jedoch, dass diese top-level Klassifikationen nicht so kritisch zur Lösung von bestimmten Problemen sind wie die lower-level Unterscheidungen. Wenn eine „high-performance KB“ (MNCF 35) wie Cyc zudem das Importieren von domain-spezifischen Ontologien auf den untersten Ebenen erlauben würde, könnte so von bestehenden Klassifikationen Gebrauch gemacht werden. Dies würde viele der von Cyc (durch eine ungeeignete Ontologie) eigens erzeugten Problemen umgehen und könnte Cyc so auch brauchbar für NLP machen.

5. Referenzen

Collins, Harry M.: <http://www.stanford.edu/group/SHR/4-2/text/collins.html>

Computerworld: <http://www.computerworld.com/news/2002/story/0,11280,69881,00.html>

Cycorp. <http://www.cyc.com>

Dreyfus, Hubert: <http://www.psych.utoronto.ca/~reingold/courses/ai/cyc.html>

Freedman, David H., *Common Sense and the Computer* (Discover, August, 1990); auf Locke, Christopher: <http://www.panix.com/~clocke/ieee.html>

Lenat, Douglas B., R.V. Guha. *Building Large Knowledge-Based Systems: Representation and Inference in the Cyc Project*. Reading: Addison-Wesley Publishing Company, Inc., 1990.

Lenat, Douglas B.: <http://auschron.com/issues/dispatch/1999-12-24/screens_feature.html>

Lenat, Douglas B.: <http://www.cyc.com/glossary.html#assertion>

Lenat, Douglas B. R.V. Guha: <http://www.cyc.com/tech-reports/act-cyc-134-90/sectionstar3_5.html>

Mahesh, Kavi, Sergei Nirenburg, Jim Cowie und David Farwell. *An Assessment of Cyc for Natural Language Processing: MCCS-96-302*. Las Cruces: Computing Research Laboratory, New Mexico State University.

Pratt, Vaughan: <http://www.cs.umbc.edu/~narayan/proj/cyc-critic.html>, 19/04/1994.

Whitten, David: <http://www.robotwisdom.com/ai/cycfaq.html>