

Computer mit Verstand:

CyC

**Eine Arbeit im Rahmen des Proseminars „Künstliche Intelligenz“
der Universität Ulm im Sommersemester 2004
von Ludwig Lausser**

Inhaltsverzeichnis:

	Index	2
I.	Kurzzusammenfassung	3
II.	Vorbereitung	3
	I. 1. Einleitung	3
	I. 2. Ontologie in der Philosophie	3
	I. 3. Ontologie in der Informatik	4
III.	Die Idee eines Intelligenten Computers	5
IV.	Wichtige Bestandteile von CYC	8
	III. 1. Die Cyc Knowledgebase	8
	III. 2. Die Cyc Inference Engine	9
	III. 3. Die Cyc Repräsentationssprache CycL	9
	III. 4. Das Natural Language Processing Subsystem	10
	III. 5. Der Cyc Semantic Integration Bus	12
	III. 6. Das Developer Toolset	13
	Zusammenfassung	13
	Literatur	14

Kurzzusammenfassung

Diese Arbeit beschäftigt sich mit der Idee und Entwicklung des Cyc – Projekts. Es werden die zugrunde liegenden Begriffe der philosophischen Ontologie und Ontologie in der Informatik näher erläutert. Danach werden die ursprüngliche Idee des Cyc–Projekts vorgestellt und die ersten Entwicklungsschritte und Schwierigkeiten aufgeführt. Als Letztes soll das Projekt in seinem aktuellen Entwicklungsstand (2004) näher untersucht werden.

I. Vorbereitung

I. 1. Einleitung:

Mit dem Cyc–Projekt ist 1984 ein Versuch unternommen worden, einem Programm ein Verständnis für menschliches Wissen zu vermitteln. Dabei wurde vor allem auf die Idee der Ontologien gesetzt. Die Probleme, die diese Technik aufwirft, hat sie vom ursprünglichen Sinn der Ontologie, der philosophischen Ontologie, geerbt. Diese Probleme wurden in den letzten 3000 Jahren nicht zufrieden stellend gelöst. Inwieweit der Ansatz der Informatik eine Lösung zu diesen Problemen darstellen kann, wird eine Beobachtung der verschiedenen Projekte auf diesem Gebiet zeigen. Das Cyc – Projekt ist im Augenblick das weltweit größte Projekt dieser Art. Der Aufsatz soll einen allgemeinen Überblick über dieses Projekt geben. Dazu werden die ursprüngliche Idee, die geschichtliche Entwicklung und der aktuelle Stand der Dinge beleuchtet. Zuerst soll jedoch der grundlegende Begriff der Ontologie in seiner philosophischen Grundbedeutung und seiner Verwendung in der Informatik erklärt werden.

I. 2. Ontologie in der Philosophie:

Die Fragestellungen der Ontologie sind in der Philosophie auf die griechischen Vorsokratiker zurückzuführen. Auch die Fragestellungen der „ersten Philosophie“ des Aristoteles, sofern sie nach dem Seienden, seinem Wesen und den ihm zukommenden Bestimmungen fragen, können der Ontologie zugeschrieben werden. Später wurde sie verallgemeinert und Metaphysik genannt. Für die Scholastik bildet insbesondere die spezielle Ontologie die höchste Form der Metaphysik. Die katholische Philosophie im Allgemeinen und die Neuscholastik im Speziellen lehren bis heute Inhalte der Metaphysik unter dem Begriff der natürlichen Theologie. Im 18. Jahrhundert wurde durch die Arbeiten Immanuel Kants (1724 - 1804) die Ontologie als unhaltbare Metaphysik entlarvt, da sie keine realistische Seinslehre darstellt. Auch in der darauf folgenden Epoche, dem deutschen Idealismus, galt diese Meinung weiter.

Erst im 20. Jahrhundert wurden wieder ontologische Forschungen aufgegriffen, die es sich allesamt zum Ziel gesetzt hatten, die angeklagten Mängel der alten Disziplinen, scholastischer, spekulativer und rationaler Ontologie zu beseitigen. Insbesondere sind hier die Arbeiten von E. Husserl, G. Jacoby, N. Hartmann, A. Meinong, J. Rehmke und M. Heidegger zu nennen.

Doch was ist die Ontologie eigentlich? Der Begriff Ontologie stammt aus dem Griechischen und bedeutet soviel wie die Lehre oder das Wissen (logos) vom Seienden (ontos). Ihr Ziel ist es, die Prinzipien des im allgemeinen Wirklichkeitsverständnis fraglos hingenommenen Bestandes des Gegebenen in Erfahrung zu bringen. Hierbei kann man die Ontologien in zwei Gruppen unterteilen, die formalen Ontologien und die materialen Ontologien. Die formalen Ontologien beschäftigen sich mit formalen Strukturen und Gesetzmäßigkeiten, die keine direkte Anschauung in unserer Wirklichkeit besitzen. Angewandt werden in ihr hauptsächlich die drei bestimmenden Grundsätze: Der Satz der Identität, der Satz des Widerspruchs und der Satz des ausgeschlossenen Dritten. Diese dienen zur Bestimmung und Festlegung von Kategorien und Transzendentalien [6], die die Festlegung eines einzelnen Seienden im Seinszusammenhang ermöglichen. Diese drei Sätze bilden auch die Grundlage vom Satz des zureichenden Grundes.

Die materiale Ontologie dagegen beschränkt sich auf die inhaltliche Gliederung des Seienden, also mit Gesetzmäßigkeiten unserer real existierenden Welt. Das Modell, das hierbei am häufigsten eingesetzt wird, ist das Schichtenmodell (Schichtenontologie). Man kann hierbei subjektive oder objektive Kriterien als Grundlage wählen. So liefert die Seinsordnung, die Platon vorgeschlagen hat, ein Beispiel für eine Basis von subjektiven Kriterien. Platon legt seiner Ordnung die drei menschlichen Seelenvermögen „Geist“, „Gemüt“ und „Leidenschaft“ zugrunde. Die Seinsordnung von Aristoteles dagegen basiert auf einer Gliederung in „Totes“, „Lebendiges“, „Geistiges“, und „Göttliches“. Damit ist sie ein Beispiel für eine Ordnung mit objektiven Kriterien als Basis. Innerhalb einer solchen Ordnung der Begriffe gibt es ein Vollkommenheitsstreben zur höchsten Seinsform. Dieses Vollkommenheitsstreben wird meist als eine hierarchische Struktur ihrer Begriffe

dargestellt. Den Begriff der höchsten Seinsform würde man heute wahrscheinlich mit dem Begriff der höchsten Abstraktheit gleichsetzen und meint soviel wie das Seiende, das keinem Wandel unterzogen ist, also das geistige, göttliche Sein. In diesem Punkt unterscheidet man zwischen der allgemeinen Ontologie und der speziellen Ontologie. Während die allgemeine Ontologie die höchste Seinsform im transzendentalen Bereich sucht, vermutet die spezielle Ontologie ein göttliches Wesen, das diese innehat. Die spezielle Ontologie erfreute sich von Anfang an in der christlich – religiösen Ausbildung besonderer Beliebtheit und wurde dort zu einem der Kernfächer der Scholastik. Alle drei vorgestellten Ausprägungen der Ontologie, also die platonische, die aristotelische und die scholastische Ontologie, sind durch ihr Dringen in das Transzendente untrennbar mit der Metaphysik verbunden. Durch den Fall der Metaphysik durch Immanuel Kants „Kritik der reinen Vernunft“ wurden auch unweigerlich die bestehenden Ansätze der Ontologie kritisch beäugt. Lange Zeit wurde sie deshalb gemieden, und die bestehenden Systeme nicht mehr wirklich weiter entwickelt. Die oben erwähnten neuen Arbeiten zu diesem Thema beschäftigen sich hauptsächlich mit der Diskretisierung der Ontologie und damit mit deren Entkoppelung von der Metaphysik [6].

I. 3.Ontologie in der Informatik

Der Begriff der Ontologie, wie ihn die Informatik gebraucht, kommt eher den Bedingungen der materialen Ontologie gleich, auch wenn diese Einteilung hier nicht ganz zutrifft, denn hier wird die Ontologie dazu eingesetzt Wissen zu repräsentieren. Die Repräsentation von Wissen gehört zu einer der Kernanwendungen eines Programms. In vielen Anwendungen ist es notwendig Erkanntes und Erdachtes zu repräsentieren. Das erworbene Wissen muss auch in Form von Fakten, Sachverhalten oder Regeln für andere (menschliche oder maschinelle) Teilnehmer an einem Geschäftsprozess, an einem juristischen Verfahren oder aber dem Benutzer einer technischen Anwendung oder einer Webseite verständlich gemacht werden.

Im Gegensatz zum maschinellen Anwender kann sich der Mensch, auf Grund seiner Wissensbasis über die Sache und deren Kontext Anleihen aus dem gespeicherten Wissen des jeweiligen Fachbereichs holen. Gespeichertes Wissen kann in diesem Zusammenhang sowohl eine Datenbank oder Webseite, aber auch ein gewöhnliches Lehrbuch, ein Regelwerk, ein Lexikon oder ein Schlagwortregister sein.

Ein Computer kann natürlich nur auf elektronische Wissensspeicher zurückgreifen. Dabei sind neben den gewünschten Daten auch immer Informationen über die Strukturierung der Daten enthalten. Ein Programm braucht, um solche Such- und Kommunikationsaufgaben zu erfüllen, eine Repräsentation der zugrunde liegenden Begriffe und deren Zusammenhang. Für solche Repräsentationen hat sich in verschiedenen Bereichen der Informatik (etwa Datenbanken und Künstliche Intelligenz) der Begriff Ontologie eingebürgert.

Eine Definition für den Ontologiebegriff in der Informatik liefert T. Gruber:

„An Ontology is a specification of a conceptualization.“
„Eine Ontologie ist eine Spezifikation der Konzeptionalisierung.“

Die Ontologie enthält also alle Beziehungen, Ableitungsregeln und genormten Begrifflichkeiten. Dieses zugrunde liegende Vokabular ist meist als Taxonomie gegeben. Enthalten in dieser sind Klassen, Relationen, Funktionen und Axiome. Im Gegensatz zur Philosophie ist in der Informatik auch der Plural „Ontologien“ geläufig, da es in einem Wissensgebiet eine eigene unabhängige oder mehrere gleichwertige Terminologien gibt.

Ontologien in der Informatik kann man primär nach ihrem Anwendungsbereich unterscheiden. Eine weitere Möglichkeit zur Differenzierung bietet der Umfang einer Ontologie. Ron Weber schlägt dafür folgende Differenzierung vor:

1. *„top level ontologies“; allgemeine bereichsübergreifende Ontologien*
2. *„domain ontologies“; anwendungsbereichbezogene Ontologien*
3. *bekannte konzeptionelle Daten- und Klassenmodelle, die lediglich durch den modischen Namen „Ontologie“ aufgewertet werden sollen.*

Die dritte Art ist im Kontext dieses Aufsatzes zu vernachlässigen. Ontologien sind allgemein gesprochen eine wichtige Grundlage für alle Bereiche der Informatik, die sich mit Wissen auseinandersetzen. Sie werden vor allem in den Bereichen der Kommunikation, des automatischen Schließens und der Wissensrepräsentation sowie der Wiederverwendung von Wissen eingesetzt.

Beim Erstellen einer Ontologie im Bereich der Informatik wird zwischen zwei Tätigkeiten unterschieden, dem Ontologie - Entwurf („ontology design“) und der Ontologie – Technik („ontology engineering“).

Ontological Engineering stellt dabei das Äquivalent zum *Software Engineering* dar. Hier werden alle Arbeiten geleistet, die den Umgang mit der Ontologie vereinfachen und sie robuster gegen falsche Eingaben oder schnelles Veralten machen.

Mit einem Ontologie – Entwurf legt der Entwickler fest, auf welche Weise das Wissen in seiner Ontologie repräsentiert und vermehrt werden soll. Bei einem induktiven Ansatz werden zuerst kleinere Ontologien gebildet und danach zu größeren verbunden. Bei einem deduktiven Ansatz wird zuerst ein allgemeines Gerüst aus allgemeinen Konzepten, Regeln und Begriffen in der Ontologie festgelegt und später durch eine nachträgliche Verfeinerung dieser Strukturen ausgebaut [5].

Obwohl Ontologien als Wissensrepräsentationen eines der ältesten Gebiete der KI – Forschung ist wurde das Interesse unserer heutigen Gesellschaft an diesen insbesondere durch zwei Entwicklungen geweckt, die mit der Netzwerktechnologie und insbesondere dem Internet zusammenhängen:

1. Ontologien sind nicht nur ein guter Ansatz um Wissen zu repräsentieren, sie sind auch dafür geeignet, aus großen Wissensmengen gesuchte Daten zu extrahieren. Das Internet als größter „Wissenspool“ stellt eine unüberschaubare Menge an Dokumenten dar. Webseiten können durch Ontologien so aufgewertet werden, dass sie für Suchmaschinen leicht zu finden sind (z.B. XML - Tags). So wird es dem Benutzer leicht gemacht, an gewünschte Informationen zu gelangen (*Semantic Web*).
2. Das zweite große Interesse liegt in der Erschaffung einer Ontologie über Software – Engineering. Eine solche Ontologie könnte z. B. als Übersetzer zwischen zwei unterschiedlich arbeitenden Maschinen eingesetzt werden. Dabei muss weder Maschine A etwas über Maschine B noch umgekehrt wissen. Maschine A sendet nur einen Arbeitsauftrag an Maschine B. Dieser wird durch die Ontologie geleitet und von dieser für die Maschine B übersetzt. Sollte nun Maschine B durch eine andere Maschine B' mit einem anderen Befehlssatz ersetzt werden, wird der allgemeine Arbeitsablauf nicht gestört.

Hier soll im Speziellen auf das Cyc – Projekt und seine Eigenheiten eingegangen werden. Es sei aber trotzdem erwähnt, dass es eine ganze Fülle anderer Ontologiebeschreibungssprachen gibt, die auch weit verbreitet eingesetzt werden, zum Beispiel:

XML, RDF, OWL (= OIL und DAML)

II. Die Idee eines Intelligenten Computers

„People have silly reasons, why computers don't really think. The answer is, we haven't programmed them right; they just don't have common sense. There's been only one large project to do something about that, that's the famous Cyc - project.”
~ Marvin Minsky, MIT, Mai 2001

Die Idee, einen Computer mit einer künstlichen Intelligenz im engeren Sinne von einem eigenen menschenähnlichen Verstand auszustatten, wird von den einigen Menschen als Science Fiction abgetan. Viele Gründe werden sowohl dafür als auch dagegen genannt. Doch durch die immer größere Leistung von Rechnern und Algorithmen tauchten in der letzten Zeit immer mehr einzelne Hinweise und Erfolge auf, die auf die Möglichkeiten, einen Computer zu intelligenten Handlungen zu bringen, deuten. So rückte die Diskussion wieder zurück in den Bereich der Wissenschaften. Die Disziplinen, die sich an ihr beteiligen, sind vor allem die Kognitionswissenschaften. Diese Gruppierung besteht aus Philosophie, Psychologie, Neurobiologie und Informatik.

Eines der wichtigsten Argumente gegen die mögliche Erschaffung eines denkenden Computers ist, dass der Maschine einfach geeignete Begriffe fehlen, über die sie nachdenken kann. Geeignete Begriffe bedeutet in diesem Fall, dass der Computer zu einem Begriff genügende und anwendbare Hintergrundinformationen hat, um diesem Begriff ein Objekt zuzuordnen zu können. Da dem Computer solche Begriffe aber fehlen, kann er auch nicht über deren Semantik Rückschlüsse oder logische Verknüpfungen ziehen, die denen des menschlichen Denkens gleichkommen. Kurz gesagt, der Mensch ist dem Computer durch seinen Menschenverstand oder seine Vernunft (*common sense*) überlegen. Diese Aussage kann man noch etwas konkretisieren, indem man den Begriff „Vernunft“ durch den der empirisch (meist unterschwellig) erworbenen Regeln ersetzt, die ein Mensch, auch bevor er in dem hier gemeinten Sinne zu denken anfängt, im Laufe seines Lebens erwirbt.

Dieses Argument zielt eigentlich darauf ab, dass Wissen nur auf der Grundlage von schon vorhandenem Wissen entstehen kann, und trifft damit eine Aussage, die nicht nur für die Informatik von großem Interesse ist.

Glücklicherweise bewirkte dieses Argument, das ja eigentlich gegen den „denkenden Computer“ gerichtet war, eine ganz andere Wirkung als erwartet. Anstatt das Gesagte als gegeben hinzunehmen, entstanden nun verschiedene Bestrebungen, den Computer mit Menschenverstand, einem „common sense“, auszustatten. Eines der heute erfolgreichsten Projekte dieser Art wurde von Douglas B. Lenat ins Leben gerufen und unter dem Namen Cyc bekannt [4].

Das Cyc – Projekt hat sich im Gegensatz zu anderen Projekten mit einer allgemeinen Breite an Wissen auseinandergesetzt und war deshalb im Laufe seiner Entwicklung immer wieder anderen Herausforderungen ausgesetzt. Dies hatte zur Folge, dass sich die ursprünglichen Konzepte immer wieder beugen und verändern mussten, um allen Bereichen gerecht zu werden. In Folgenden soll auf die ursprünglichen Ideen der Entwickler eingegangen werden. Die jetzige Lage (Mai 2004) schildert der Abschnitt „**III. Wichtige Bestandteile von Cyc**“. Der Name des Cyc – Projekts leitet sich übrigens vom englischen Wort *cycle* ab und symbolisiert den Aufbau der inneren Struktur der Projekts. Bevor aber näher darauf eingegangen wird, soll die innere Struktur näher beschrieben werden.

Sucht man nach Wissen, das Mensch und Computer gemeinsam haben, stößt man sehr schnell an Grenzen. Selbst unwichtigste Banalitäten sind dem Rechner fremd, da er sie für seine Anwendungen nicht benötigt. So hören sich die folgenden Wissensgrundlagen für den normalen Leser nicht eben nach wissenschaftlichen Erkenntnissen an.

- *Man muss wach sein um zu Essen.*
- *Normalerweise kann man die Nase eines Menschen sehen, nicht aber sein Herz.*
- *Angenommen, wir betrachten zwei Berufe. Dann ist entweder der eine eine Spezialisierung des anderen oder sie sind beide unabhängig.*
- *Man kann sich nicht an Dinge erinnern, die noch nicht passiert sind.*
- *Wenn man ein Stück Erdnussbutter in zwei Hälften teilt, ist jede Hälfte ein Stück Erdnussbutter; aber wenn man einen Tisch in zwei Hälften teilt, ist keine Hälfte ein Tisch.*

Solche Erkenntnisse sind dem Leser wahrscheinlich in irgend einer Form bewusst, auch wenn er sie nie explizit aus einem Schulbuch, Lexikon oder Wörterbuch gelernt hat. Doch genau auf dieser Ebene begann 1984 ein kleines Team um Douglas B. Lenat und seiner Kollegin Mary Shepherd mit der Kodifizierung unseres alltäglichen Wissens. Es wurde damit begonnen, eine großangelegte Ontologie zu errichten, in der die Gegenstände und die dazugehörigen Aussagen abgelegt werden sollten. Genannt wird diese *Ontologie Cyc – Knowledgebase*, was übersetzt ungefähr Cyc – Wissensbasis bedeutet. Auf die *Knowledgebase* geht der Punkt „**III. 1. Wichtige Bestandteile von Cyc – Die Cyc Knowledgebase**“ noch genauer ein.

Zunächst blieb es bei Aussagen über Gegenstände und Situationen unseres alltäglichen Lebens. Die einzige sinnvolle Möglichkeit, dem Computer solche Regeln und Erklärungen mitzuteilen, die zu diesem anfänglichen Zeitpunkt gesehen wurde, war die, die Regeln per Hand in die Wissensbasis des Rechners zu schreiben. Im Team sollten sich im Laufe der nächsten Jahre einige Schlüsselfiguren hervortun, die besonders dafür verantwortlich waren, den Überblick über die immer größer werdende Masse an Regeln und Erklärungen zu behalten: Karen Pittman, R. Guha, Nick Siegel, Kathy Burns, Keith Goolsbey, Ken Murray, David Gadbois und weitere [4].

Andere Möglichkeiten, dem Computer Wissen zu vermitteln, wurden vom Cyc–Team allerdings anfangs gemieden bzw. bis jetzt noch nicht umgesetzt.

Ungefähr zur selben Zeit wie das Konzept der Ontologien und im speziellen die Anfänge des Cyc – Projekts ist auch die Idee des Natural Language Understandings (NLU) entstanden [7]. Idee dieser Technologie ist es den Inhalt eines geschriebenen Textes in syntaktische Gerüste einzufügen, die dem Computer verständlich sind. Unter Einsatz eines solchen Programms könnte man sich unter anderem auch das Kodifizieren von Wissen durch einen menschlichen Programmierer sparen. Einer NLU – tauglichen Knowledgebase könnte man zum Beispiel geeignete Texte vorlegen oder das Wissen in einer gesprochenen Sprache (z. B. Englisch) eingeben. Der Computer wäre nun von sich aus in der Lage das Vorgelegte in seine intern verwendete Repräsentationsform umzulegen und es in seiner Wissensbasis einzuordnen. Diese Technologie steckte jedoch zu Beginn des Cyc – Projekts noch in den Kinderschuhen, und so wurde sie, um nicht zusätzliche Fehlerquellen und technische Probleme auf sich zu ziehen, außen vor gelassen. An einem kurzen Beispiel sei gezeigt, wie sich Ungenauigkeiten eines solchen Programms auf die Wissensrepräsentation auswirken können:

Die Polizei verhaftete die Demonstranten, weil sie Gewalt fürchteten.

Einem normalen Menschen ist dieser Satz von Anfang an völlig klar. Das Problem, das sich hier für ein NLU – System stellt, ist die Zuordnung des kausalen Nebensatzes. Das Pronomen „sie“ könnte sich, rein grammatikalisch, sowohl auf „die Polizei“ als auch auf „die Demonstranten“ beziehen. Ein reines NLU – System könnte in diesem Fall nicht unterscheiden. Das Cyc – Team beschäftigte sich dann ab 1996 zum ersten Mal näher mit der Implementierung eines solchen Systems. Genauer zur technischen Umsetzung und der entstandenen Problemen auf diesem Gebiet im Abschnitt „**III. 4. Wichtige Bestandteile von Cyc - Das Natural Language Processing Subsystem**“.

Auch die Methode des *Machine Learning* [7], ebenfalls um dieselbe Zeit entstanden wie das Cyc – Projekt, war zu Beginn der Arbeiten nicht sinnvoll anwendbar. Diese Technik arbeitet dabei der Idee nach auf einer bestehenden Wissensbasis. Hier werden die bestehenden Regeln untersucht und weitergegeben oder enger gefasst. Beispiele hierfür wären in etwa:

Regel 1: Jeder Tag ist entweder Sonntag, Montag, Dienstag, Mittwoch, Donnerstag, Freitag, Samstag.
Regel 2: Von Sonntag, Montag, Dienstag, Mittwoch, Donnerstag, Freitag, Samstag hat jeder 24 Stunden.
→ *Regel 3 :Ein Tag hat 24 Stunden.*

Regel 1: Schwäne sind weiß
Regel 2: In Australien gibt es schwarze Schwäne
→ *Regel 1: Es gibt weiße und schwarze Schwäne*

Das erste Beispiel zeigt den Fall einer sog. Erweiterung, das zweite den einer Verengung. Das Problem dieser Methode liegt auf der Hand: Die Technik basiert auf einer schon vorhandenen Wissensbasis. Die Schlüsse, die das *Machine Learning* zieht, können sich nur auf die Wissensregeln beziehen, die dem Rechner schon bekannt sind. Würde so zum Beispiel die 2. Regel im Schwanenbeispiel fehlen, würde eine Anfrage, ob ein schwarzer schwanähnlicher Vogel ein Schwan sein könnte, sicherlich mit nein beantwortet werden. Wird dem Machine Learning die Wissensbasis ganz entzogen, wird es komplett nutzlos. Im Prinzip ist es nämlich dasselbe Problem, das Fehlen einer „menschlichen“ Vernunft, das den Computer am Lernen wie auch an anderem intelligentem Handeln hindert. Da das Cyc – Projekt gerade dieses intelligente Handeln ermöglichen soll, ist es nur logisch, dass es in seiner Entstehung nicht darauf zurückgreifen kann. Erst nach dem Erreichen der „kritischen Masse“ an Wissensregeln (etwa Mitte der 1990er) war es dem Cyc – Team möglich, auf diese Technologie zurückzugreifen und an einer Implementierung zu arbeiten.

Des Weiteren wurde größtenteils auf numerische und statistische Werte bei der Formulierung verzichtet. Der Grund hierfür liegt in der Fülle der vorhandenen Terme und ihrer zugehörigen Regeln. Speziell die Ontologie des Cyc – Projekts hat sich ja zum Ziel gesetzt, dem Rechner eine „menschliche“ Vernunft, also ein Wissen über die alltäglichen Bezugsgegenstände und Umgebungen des Menschen zu vermitteln. Wie wir oben gesehen haben, ist der Wissensinhalt einer solchen Regel für den Menschen banal oder deren Bedeutung obskur. Genau aus diesem Grunde sind zu den meisten dieser Aussagen nie statistische Erhebungen gemacht worden. Wie groß ist zum Beispiel die Wahrscheinlichkeit, dass ich die Nase einer anderen Person sehe? Man müsste zu jeder Beziehung, in der eine Regel steht, einen numerisch gesicherten Wert nachliefern. Ein solches Unterfangen würde selbst einen so groß angelegten Rahmen, wie den des Cyc–Projekts sprengen. In der Wissensrepräsentation an sich werden bis heute nur dort solche Zahlen eingesetzt, wo großangelegte statistische Erhebungen bekannt sind. Im Vergleich zur gesamten *Knowledgebase* ist dies ein verschwindend geringer Anteil.

Dennoch wird nicht komplett auf die Möglichkeit verzichtet, Wahrscheinlichkeiten ausdrücken zu können. Allein ihre Basis wirkt auf den ersten Blick für eine maschinelle Implementierung befremdlich. Ob nun im Kopf des Programmierers oder auf maschineller Ebene werden hierfür Pro- und Contra- Argumente gesammelt. Diese Sammlung ist im Allgemeinen frei von strenger Logik (dem Propagieren von absolutem Wahr oder Falsch) und Arithmetik (das Verrechnen und Propagieren von numerisch sicheren Werten). Das Ergebnis dieser Überlegungen wird dann in einer Regel über schon bestehende Regeln festgeschrieben. Diese neue Regel heißt dann Metaregel und ist eines der speziellen Konstrukte der Repräsentationssprache des Cyc – Projekts. Ein Beispiel würde in etwa so lauten:

Regel 1: Menschen wohnen in Häusern.
Regel 2: Menschen leben in Zelten.
Metaregel: Regel 1 ist wahrscheinlicher als Regel 2

Eine solche Metaregel trifft nur für einen bestimmten Kontext zu. Der Begriff Kontext wird später in diesem Abschnitt erklärt.

Näher auf die Anforderung an die Repräsentationssprache wird im Abschnitt „**III. 3. Wichtige Bestandteile von Cyc - Die Cyc Repräsentationssprache CycL**“ vorgestellt.

Ein anderes Problem, das sich den Programmierern von Anfang an stellte, ist die Tatsache, dass das „menschliche“ Wissen nicht universell gültig ist. Es gilt sogar meist nur in dem Teilbereich unseres Lebens, in dem es gewonnen worden ist. So gilt zum Beispiel die Regel „Normalerweise kann man die Nase eines Menschen sehen, nicht aber sein Herz“ nicht während einer Herzoperation. Diese Eigenschaft des Wissens erfordert einige spezielle strukturelle Voraussetzungen von einer Wissensbasis. In der Cyc – *Knowledgebase* wurde dieses Problem durch zwei miteinander verwandte Konzepte gelöst. Das erste war die Einführung von

Kontexten. Ein Kontext, oder auch Mikrotheorie genannt, beschreibt, wie im üblichen Sprachgebrauch auch, die Umstände, unter welchen ein Gegenstand betrachtet wird. In unserem oberen Beispiel wäre ein solcher Kontext die Herzoperation. In einem solchen Kontext wird festgelegt, welche Regeln für ihn zutreffen und welche nicht. Zudem werden zu den bestehenden Regeln neue hinzugefügt, die nur speziell auf diesen Kontext zutreffen. Doch ein Kontext in der Cyc – Wissensbasis kann auch abstraktere Bedeutungen haben, wie zum Beispiel der Kontext „temporär“. Dieser Kontext enthält zum Beispiel alle Regeln, die sich auf zeitlich begrenzte Ereignisse beziehen. Eine typische Regel für solch einen Kontext wäre:

„Heute ist Donnerstag“

Das diese Regel nicht immer zutreffen kann, ist offensichtlich.

Das zweite Konzept basiert eben auf diesem Begriff des Kontextes. Wie oben schon gezeigt, können diese Kontexte auf verschiedenen Abstraktionsebenen liegen. Das zweite Konzept stellt nun eine hierarchische Ordnung dieser Konzepte dar. Umso abstrakter ein Kontext ist, desto höher steht er in der Hierarchie. Zur genauen Strukturierung dieser Hierarchie siehe den Abschnitt „**III.1. Wichtige Bestandteile von Cyc - Die Cyc Knowledge Base**“ eingegangen.

Diese beiden Konzepte zusammen brachten dem Cyc – Team entscheidende Vorteile ein. Zum einen konnte diese Hierarchie gleichzeitig auch für Vererbung genutzt werden, so dass bestehende Regeln einfacher von einem Kontext einer hohen Abstraktionsebene auf verwandte Kontexte einer niedrigeren Abstraktionsebenen übertragen werden konnten. Für unser Beispiel „Herzoperation“ gelten zum Beispiel die Regeln des Kontextes „temporär“.

Zum anderen konnten Regeln, die auf den ersten Blick widersprüchlich waren, aber beide ihren Belang und ihre Richtigkeit hatten, parallel verwaltet werden, ohne sich gegenseitig zu behindern. Dies ist gerade unter dem Anspruch, das gesamte „menschliche“ Grundwissen abzudecken, von äußerster Wichtigkeit.

So wäre eine Verwaltung folgender Regeln ohne eine Kontext Bildung nur schwer möglich:

Ohne Kontext:

Regel 1: Es gibt keine Vampire.

Regel 2: Es gibt Vampire.

Mit Kontext:

Regel 1: Kontext „Realität“ : Es gibt keine Vampire.

Regel 2: Kontext „Märchen“ : Es gibt Vampire.

Ein weiterer großer Vorteil waren diese Strukturen in Bezug auf die Wiederauffindung von Wissensregeln. Durch diese hierarchische Strukturierung war es möglich, sich anhand der Kontexte zu dem Wissensgebiet vorzuarbeiten, das benötigt wurde. Die Hierarchie bietet hierbei eine Art Baumstruktur und schließt damit von Anfang an eine gewaltige Menge von Wissen, das für die momentane Anfrage nicht gebraucht wird, aus. Eine unstrukturierte Suche wäre bei einer Wissensansammlung von Cycs Größenordnung nicht denkbar.

Wie oben bereits erwähnt waren diese Kontexte es, die dem Cyc – Projekt seinen Namen gaben. Cyc ist eine Abkürzung für das englische Wort cycle, was soviel bedeutet wie Kreis oder Ring. Douglas B. Lenat sah jeden dieser Kontexte als einen einzelnen Ring in einem Kettenhemd an. Jeder einzelne an sich ist nicht besonders stark und biegsam. Verflechtet man sie jedoch zu einem Kettengewebe, so erhält man eine Rüstung, die den Träger vor Angriffen schützt.

III. Wichtige Bestandteile von CYC

Nachdem die grundsätzlichen Ideen des Cyc – Projekts vorgestellt worden sind, soll nun darauf eingegangen werden, wie ein derzeitiger Cyc –Server der Firma Cycorp arbeitet (Stand Mai 2004). Die Informationen hierzu entstammen der Homepage der Firma Cycorp www.cyc.com [1]. Besonders sollen hier die wichtigsten Bestandteile eines Cyc – Servers näher betrachtet werden. Diese Bestandteile sind *Cyc – Knowledgebase*, *Inference Engine*, *Cyc Repräsentationssprache CycL*, *Natural Language Processing Subsystem*, *Semantic Integration Bus*, *Developer Toolset*.

III. 1. Die Cyc Knowledgebase

Die *Cyc Knowledgebase* ist die formalisierte Wissensrepräsentation, in der das „menschliche“ Wissen des Programms gespeichert ist. Hierbei ist zu unterscheiden zwischen den Termen und den Regeln, die in dieser zu finden sind. Die Terme repräsentieren hierbei die Objekte, über die Aussagen getroffen werden sollen. Die Regeln sind das Wissen, das über die Terme kodifiziert wurde. Die Repräsentationssprache, die intern in der

Knowledgebase verwendet wird, heißt CycL. Im Jahre 2004 ist der Fundus an Termen auf ca. 200.000 Stück angewachsen. Im Durchschnitt sind zu jedem Term mehrere Dutzend Regeln vorhanden. Inzwischen ist der Großteil dieser Regeln automatisch vom Computer generiert und ergänzt worden. Dies beruht allerdings hauptsächlich auf der Systematisierung der Regeln und nicht auf einer vollständigen Neugenerierung. Doch auch die menschlichen Programmierer bringen immer wieder neue Terme und Regeln in die *Knowledgebase*.

Die *Knowledgebase* ist nicht mehr auf einem Rechner konzentriert. Vielmehr ist das Wissen auf mehrere Rechner, Cyc – Agenten, verteilt. Jeder Cyc – Agent verfügt hierbei über eine gewisse Menge an Grundwissen, die er mit allen anderen Cyc – Agenten gemein hat. Der eigentliche Bestand seines Wissens besteht allerdings aus Regeln und Termen zu einem ganz speziellen Fachbereich. Dieser Fachbereich ist den anderen Agenten bekannt, und so können Anfragen gezielt von einem Rechner zum anderen gesendet werden. Es gibt dabei keine hierarchische Struktur unter den Agenten, und jeder Cyc – Agent kann Anfragen an einen beliebigen anderen Cyc – Agenten stellen. Zwei Cyc – Agenten sind beide der CycL mächtig, was später den Unterschied zu den anderen Komponenten des Cyc Semantic Integration Bus darstellen wird (virtuelle Agenten).

Erste Versuche auf diesem Gebiet wurden von Cycorp in Zusammenarbeit mit dem Computer Science Department of Maryland County (UMBC) gemacht. Bei den ersten Experimenten wurden drei Agenten miteinander verschaltet. Ihre drei Fachbereiche waren Geographie, Politik und Volkswirtschaftslehre. Eine der Testfragen an den Geographie – Agenten lautete:

„*elected heads of government of countries north of the equator:*“

in CycL:

```
(#$and
  ($headOfGovernmentOf ?x ?y)
  ($hasAttributes ?x #$Elected)
  ($northOf ?y #$Equator))
```

Der Geographie – Agent war selbstständig in der Lage, die Länder nördlich des Äquators zu finden. Das Wissen über Wahlen und Regierungsformen musste er sich allerdings vom Politik – Agenten besorgen. Wegen dieser Spezialisierung der einzelnen Agenten spricht man in solch einem Zusammenhang oft auch von Expertensystemen [1].

III. 2. Die Cyc Inference Engine

Diese Einheit dient dazu, aus bereits bestehenden Regeln über Deduktion neue Regeln abzuleiten. Hierbei werden sowohl die Möglichkeiten des *Modus ponens*, des *Modus tolens* und der *universellen* und *existentiellen Quantoren* ausgenutzt. Dazu kommen Verfahren aus der KI, wie zum Beispiel automatische Klassifikation, zum Einsatz. Um den Suchaufwand in der enormen Wissensbasis von Cyc gering zu halten, werden vor allem zwei Strategien verfolgt. Die eine davon ist die Verwendung von Mikrotheorien, wie sie weiter oben beschrieben wurde. Die andere ist eine interne Sortierung der einzelnen Wissensregeln nach ihrer Wahrscheinlichkeit. Die für das Entscheiden notwendigen Objekte bildet die *Inference Engine* aus der als normalem Text gespeicherten CycL. Erst der *Inference Engine* ist es möglich den Zusammenhang des gespeicherten Wissens zu sehen und dieses Wissen weiterzuverarbeiten.

Für einige spezielle Kontexte gibt es spezielle Inferenz – Module. Diese behandeln dann spezielle Begriffsgruppen, wie Zusammengehörigkeit, Unabhängigkeit, Gleichheit, Zeit und Mathematik.

Über die *Inference Engine* können nicht nur logische Formeln hergeleitet werden, es können auch Aussagen auf ihre Richtigkeit überprüft werden, was im Bereich der Cyc – Anwendungen von größter Wichtigkeit ist [1].

III. 3. Die Cyc Repräsentationssprache CycL

Die CycL ist zusammen mit ihren immer komplexer werdenden Aufgaben gewachsen und hat dabei immer neue Formulierungs- und Aussagemöglichkeiten bekommen. Heute wird CycL als sehr flexibel und ausdrucksstark angesehen. Im Prinzip ist CycL eine Erweiterung der Prädikatenlogik erster Ordnung, *First-order predicate Calculus (FOPC)*, .

Ein FOPC ist eine formale Sprache, die Prädikatensymbole, Funktionssymbole, Konstantensymbole, logische Operatoren und Quantoren miteinander vereinigt, welche dazu benutzt werden können, Fakten über eine Welt auszudrücken. Im Gegensatz zur propositionalen Logik ist es in FOPC möglich, allgemeine Aussagen zu machen oder Aussagen über Existenzen zu treffen, indem quantifizierte Variablen benutzt werden. „First-order“ meint hierbei, dass Aussagen, welche quantifizierend über Prädikate und Funktionen sind, nur eine Stufe erlaubt sind.

Die verwendeten Erweiterungen erlauben es dem FOPC der CycL unter anderem, Aussagen über Gleichheiten und Default- Entscheidungen zu treffen [1].

Hier noch ein paar grundlegende Beispiele der CycL:

Allgemein beachtet die CycL die Groß – und Kleinschreibung.

Konstanten:

Unter Konstanten werden die Terme, Prädikatsymbole und Funktionssymbole zusammengefasst. Prädikate drücken hierbei Relationen zwischen Konstanten aus und Funktionen beziehen bilden mit Termen zusammen neue Terme.

Konstanten beginnen immer mit „#\$“. In ihnen dürfen alle Buchstaben, Zahlen und die Sonderzeichen „-“, „_“ und „?“ verwendet werden. Konstantennamen sind systemweit eindeutig zu wählen. Alle Konstanten außer den Prädikaten beginnen mit Großbuchstaben oder numerisch. Setzt sich ein Name aus mehreren Wörtern zusammen wird jeder Wortanfang großgeschrieben.

Bsp:

Terme:

#\$AnimalWalkingProzess

#\$Typewriter

#\$Walking00036

Prädikat:

#\$isa

#\$likesAsFriend

Funktionen:

#\$GovernmentFn

#\$GovernmentOf#\$Canada

Variablen:

Variablen werden in der CycL mit einem ? am Anfang gekennzeichnet.

Formulare:

Formulare legen fest, wie eine Aussage repräsentiert werden muss. Die Formulare in CycL sind an Lisp angelehnt. Ein Formular klammert mehrere Argumente ARG0 – ARGN in der Form „(ARG0 ARG1..ARGN)“ zusammen. Das erste Argument ist hierbei ein Prädikat, eine logische Verknüpfung oder ein Quantor. Die anderen Argumente können Konstanten, Formulare, Variablen, Zahlen oder Strings (mit Anführungszeichen) sein. Das Argument ARG0 bestimmt, wie viele weitere Argumente ARG1 – ARGN vorhanden sein müssen, damit das Formular korrekt ist. Manche Prädikate erlauben auch verschiedene Anzahlen von Argumenten.

Bsp:

(#\$likesAsFriend #\$DougLenat #\$KeithGoolsbey)

(#\$likesAsFriend #\$DougLenat #\$KeithGoolsbey #\$Fido)

(#\$colorOfObject #\$CAR #\$COLOR)

III. 4. Das Natural Language Processing Subsystem

Unter *Natural Language Understanding* versteht man die Idee, Sätze, wie sie ein Mensch in seiner Rede formulieren würde, in eine Repräsentation zu übersetzen, die ein Programm verarbeiten kann. Das *Cyc Natural Language Subsystem* ist Cycorps Versuch eines solchen Programmes. Hierbei geht es in erster Linie darum, Anfragen an und Regeln für die Cyc – Knowledgebase in Umgangssprache formulieren zu können. Die wichtigsten Komponenten des Cyc Natural Language Subsystems, das Lexikon, der syntaktische Parser und der semantische Interpreter werden nun kurz vorgestellt [1].

Das Lexikon

Eigentlich besitzt das *Cyc Natural Language Subsystem* mehrere Lexika. Dabei ist jedes für eine eigene Sprache verantwortlich. Das einzige aber, das wirklich für die gesamte *Cyc - Knowledgebase* ausgebaut ist, ist das für die englische Sprache, weshalb man bei Cycorp meistens den Singular verwendet.

Das Lexikon ist die Grundlage für das Cyc Natural Language Subsystem. Es wird als eigene Sammlung von Mikrotheorien in der *Knowledgebase* repräsentiert und beinhaltet eine Sammlung von (englischen) Wörtern. Diese Wörter werden wiederum als Terme dargestellt. So wird zum Beispiel das englische Wort „light“ in der Knowledgebase als „#\$Light – TheWord“ dargestellt. Die Regeln zu einem Term beinhalten unter anderem, ob das Wort als Nomen, Verb, Adjektiv oder Adverb eingesetzt werden kann. Andere Regeln beinhalten die syntaktischen Konstrukte, die dieses Wort eingehen darf. Bei einem Verb ist hier zum Beispiel festgehalten, ob es transitiv oder intransitiv gebraucht werden kann. In unserem Zusammenhang sind jedoch die Regeln, die die

verschiedenen Wort – Terme den Termen anderer Mikrotheorien zuordnen. So wird unser Beispiel „light“ den Termen „#\$LightEnergy“ und „#\$LightningDevice“ zugeordnet. Auch Eigennamen können auf diese Weise zugeordnet werden.

Das Lexikon ist die erste Instanz, die einen eingegebenen Satz analysiert. Im folgenden Beispiel aus dem Fundus der Cyc – Homepage sieht der Satz nach diesem Schritt so aus:

„John	saw	the	light	with	the	telescope.“
Proper Noun	Verb Noun	Determiner	Noun Verb Adjective	Preposition	Determiner	Noun Verb

Dieses Beispiel soll uns durch die weiteren Abschnitte begleiten. Der eingegebene Satz wurde nun in eine Kette von Wort – Termen verwandelt und enthält einige strukturelle Zusatzinformationen.

Der syntaktische Parser

Dem syntaktische Parser steht eine Menge von kontextfreien Grammatikregeln zur Verfügung. Wird das Ergebnis aus der ersten Analyse durch das Lexikon dem syntaktischen Parser übergeben, sucht dieser grammatikalische Konstrukte, die in diesem möglicher Weise verwendet wurden. Als Ergebnis liefert er eine Menge von baumartigen Strukturen, die alle möglichen zugrunde liegenden syntaktischen Zusammenhänge des eingegebenen Satzes widerspiegeln. Im obigen Beispiel wären, rein grammatikalisch, zwei Lösungen denkbar.

1.

```
{:SENTENCE
  {:NP
    {:DETP {#$Determiner [the]}}
    {:N-BAR {#$SimpleNoun [man]}}
  }
  {:VP
    {#$Verb [saw]}
    {:NP {:DETP {#$Determiner [the]}}
      {:N-BAR {#$SimpleNoun [light]}}
    }
    {:PP {#$Preposition [with]}
      {:NP {:DETP {#$Determiner [the]}}
        {:N-BAR {#$SimpleNoun [telescope]}}}}}}}}}
```
2.

```
{:SENTENCE
  {:NP
    {:DETP {#$Determiner [the]}}
    {:N-BAR {#$SimpleNoun [man]}}
  }
  {:VP
    {#$Verb [saw]}
    {:NP {:DETP {#$Determiner [the]}}
      {:N-BAR
        {:N-BAR {#$SimpleNoun [light]}}
        {:PP {#$Preposition [with]}
          {:NP {:DETP {#$Determiner [the]}}
            {:N-BAR {#$SimpleNoun [telescope]}}}}}}}}}
```

Die 1. Variante würde übersetzt ungefähr bedeuten „John benutzte ein Teleskop um das Licht zu sehen“. Die 2. Variante dagegen wäre gleichwertig mit dem Satz „John sah ein Licht, das ein Teleskop besitzt“. Es sei noch einmal darauf hingewiesen, dass diese beiden Varianten syntaktisch äquivalent sind.

Der semantische Interpreter

Hier werden die Ergebnisse des syntaktischen Parsers in CycL übersetzt. Die Aussagen in CycL können über verschiedene Verfahren (z. B. *Inferencing*) auf die semantischen Bedeutungen der grammatikalischen Konstrukte überprüft werden. Hierbei greift der semantische Interpreter auf die Regeln zurück, die die Wort – Terme mit anderen Termen der *Knowledgebase* verbinden. Bei unserem Beispiel würden wir auf Regeln stoßen, die uns sagen, dass Teleskope normalerweise zum Sehen von Dingen benutzt werden und dass Lichter normalerweise keine Teleskope besitzen. Aus unserer Fülle von möglichen syntaktischen Konstrukten bleiben nur noch einige sinnvolle Sätze übrig (meist sogar nur einer), aus denen dann der richtige ausgewählt werden kann. Der semantische Interpreter ist der Bestandteil des Cyc Natural Language Subsystems, der es von NLU –

Systemen ohne ontologische Unterstützung unterscheidet (Vergleiche Abschnitt „**Die Idee des intelligenten Computers**“).

Das Cyc Natural Language Subsystem brachte dem Cyc –System an sich wesentliche Vorteile. Es können geschriebene Texte zu einem Thema analysiert und in CycL übersetzt werden. Nach dem selben Prinzip können auch direkte Tastatureingaben in Prosa ausgewertet werden. Die entstehenden CycL Konstrukte können dann in die Cyc – Knowledgebase eingefügt werden. Dieses Vorgehen erleichtert die Arbeit der Wissenskodierer sogar in zwei Hinsichten. Zum Einen kann nun enormes Wissen in schriftlicher Form automatisch in CycL übersetzt werden, was das langwierige Kodieren durch simples Bestätigen ersetzt. Zum Anderen können nun auch Menschen, die keine Ahnung von CycL haben, der Cyc – Knowledgebase Wissen nahe bringen. Solche Menschen werden in der Regel Experten auf einem Fachbereich sein und damit genauer über diesen Bescheid wissen als ein normaler Programmierer. Über dieselben Mechanismen kann ein normales Textdokument auch als virtueller Cyc – Agent genutzt werden, was im nächsten Abschnitt genauer erklärt wird.

III. 5. Der Cyc Semantic Integration Bus

Nicht nur das Übersetzen allgemeinen Wissens in die Repräsentationssprache CycL war von Anfang an ein sehr mühsames Geschäft. Auch das Finden und Sammeln von Termen und Regeln war mit einem größeren Aufwand verbunden. Eines dieser Probleme stellte vor allem die Vollständigkeit von Listen und Faktenreihen dar. Diese war anderen Repräsentationsformen gegeben. Die Frage war nur, wie man diese am besten nutzt. Neben dem ständigen Erweitern der Cyc – Knowledgebase wurde hier 2002 ein zweiter schnellerer Weg gewählt, der *Semantic Integration Bus*. Dieser Bestandteil eines Cyc – Servers ist extra dazu ausgelegt, das Cyc – System mit fremden Datenquellen zu verbinden und diese nutzbar zu machen (Structured Knowledge Source Integration, SKSI). Angeforderte Daten werden aus ihnen extrahiert und dem Cyc – System zur Verfügung gestellt. Hierzu muss der Semantic Integration Bus die Informationen in CycL übersetzen (Vergleiche Cyc - Agent). Die Informationen werden dem System dann als virtuelle, temporäre Agenten repräsentiert und können, wie alle anderen Regeln auch, bei der Entscheidungsbildung mitwirken.

Bei den Repräsentationsformen wird im Allgemeinen zwischen drei Arten unterschieden: Strukturierte Daten (z. B. Datenbanken), semistrukturierte Daten (z. B. Webseiten, Tabellen) und unstrukturierte Daten (z. B. Textdokumente).

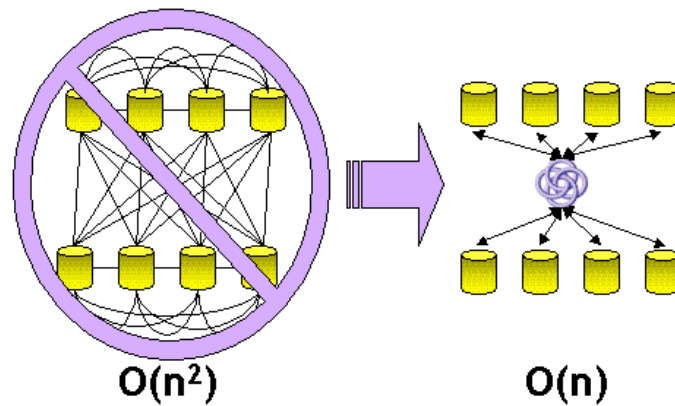
Wegen ihres enormen Umfangs und ihrer Ordnung sind Datenbanken und Webseiten von besonderer Bedeutung. Das Einbinden von Textdokumenten geschieht in erster Linie mit Hilfe des Natural Language Processing Subsystems und soll deshalb kein zweites Mal behandelt werden.

Das Einbinden von Datenbanken geschieht in erster Linie durch zwei Schritte. Zum Einen muss der Knowledgebase die Datenbank als virtueller Experte eines Fachbereichs gemeldet werden, damit diese mit in die Entscheidung einwirken kann. Zum anderen muss das Cyc – System wissen, wie das Wissen in der fremden Quelle zu erschließen ist. Hierzu müssen Anfragen in der entsprechenden Datenbanksprache (z. B. SQL) gestellt werden können. Dazu muss der Cyc – Server die Konstruktionsmöglichkeiten der Datenbanksprache gespeichert haben und die Tabellen – Namen deuten können. Die Konstruktionsmöglichkeiten einer Datenbanksprache sind in einer eigens dafür vorgesehenen Mikrotheorie der Cyc – Knowledgebase gespeichert. Mit ihrer Hilfe können die fremden Daten übersetzt und gedeutet werden. Die Bedeutungen der Tabellen - Namen werden entweder explizit im jeweiligen Kontext gespeichert oder durch das NLU – Subsystem erschlossen.

Das Erschließen von Webpages stellte ähnliche Anforderungen an das System, wie die Datenbanken. Hier konzentrierte man sich vor allem auf das HTML – Format. In diesem können unter anderem über Tags Zusatzinformationen gegeben werden, die Cyc bei der Deutung der Inhalte helfen können. Ein weiterer Vorteil des HTML – Formates ist seine relativ einfache Strukturierung. Diese erlaubt es, mit sehr einfachen Algorithmen den Quellcode zu parsen.

Eine weitere sehr interessante Eigenschaft, die der Semantic Integration Bus hat, sind seine Fähigkeiten als Mediator. Unter einem Mediator versteht man ein Programm, das es erlaubt Abfrage zu verarbeiten, bei denen verschiedene unabhängige Wissensquellen (vor allem Datenbanken) gebraucht werden. Das heißt, es ist Cyc möglich das Wissen verschiedener virtueller Agenten zu verbinden und auszuwerten, ohne die eigene Wissensbasis einzubringen. Die benötigten Komponenten sind hierbei größtenteils Cycs Kenntnisse der verwendeten Sprachen und Terme der einzelnen Wissenssammlungen. Die Wissensbasis wird hierbei dazu eingesetzt, die Anfrage in CycL zu übersetzen und dann so weiterzugeben, dass sie für den virtuellen Agenten verständlich ist. Verbindung von Wissensquellen über einen Mediator würde die Komplexität der üblichen Verbindungen von $O(n^2)$ auf $O(n)$ drücken [1].

Diese Eigenschaft ist eines der Hauptargumente für Wirtschaft und Militär, die Forschungen von Cycorp zu fördern [2].



III. 5. Das Cyc Developer Toolset

Hierunter werden alle entstandenen Tools zusammengefasst, die es dem User erlauben, einen Bestandteil des Cyc – Projekts einzusehen, zu verwenden oder zu manipulieren [1]. Die meisten dieser Programme präsentieren ihre Ergebnisse als HTML. So können zum Beispiel die Zusammenhänge der einzelnen Terme in der Cyc – *Knowledgebase* über eingebettete Links verfolgt werden. Auch mit der Inference Engine kann über HTML kontakt aufgenommen werden.

Andere wichtige Tools erlauben Folgendes:

- Ein Hierarchybrowser zeigt die gewünschten (Vererbungs-) Zusammenhänge eines Unterbaumes der Wissenshierarchie.
- Ein Lexikoneditor erlaubt es, das Cyc – Lexikon (NLU) auf benutzerfreundliche Weise zu editieren oder zu erweitern.
- Ein Englisch zu CycL - Parser, welcher es dem User erlaubt, das NLU – System zu testen.
- Ein Datenbanktool, welches eine Oberfläche zum Semantic Integration Bus bietet.
- Ein WordNet – Browser, welcher die begrifflichen Zusammenhänge in der Cyc – Ontologie darstellt.
- Ein Englischgenerator, der Regeln in CycL ins Englische übersetzt.

Zusammenfassung

Vom heutigen Standpunkt aus sind die meisten Ideen, die durch Cyc's Hilfe verwirklicht werden können, eher noch in einem spekulativen oder experimentellen Stadium anzusiedeln. Die Palette der Möglichkeiten ist weit und die Chancen, die semantisches Wissen einem Programm bringen kann sind enorm. Eine Idee, die sehr weit fortgeschritten ist, setzt direkt am NLU – Subsystem an. Cycorp entwickelt mehrere Tools, die es erlauben einen Text semantischen Prüfungen zu unterziehen. Die direkte Anwendung ist hierbei nicht das Gewinnen von Wissen aus dem Text sondern die Überprüfung des Inhaltes des Textes. Mit diesen Anwendungen soll der Benutzer auf falsche Aussagen, fehlende Ausführungen und mangelnde Literaturangaben und Querverweise hingewiesen werden. Diese Anwendungen könnten in Zukunft, ähnlich wie heute die Rechtschreib- und Grammatikprüfung, in Textverarbeitungsprogrammen eingearbeitet sein. Diese Tools verschaffen dem Benutzer gerade bei größeren Texten einen besseren Überblick. Ob Cyc wirklich das gewünschte leisten kann wird gerade getestet. Im Test ist es momentan nur möglich kleinere Texte oder Abschnitte zu überprüfen.

Es wurden die ersten Anfänge des Cyc – Projekts betrachtet. Es wurde gezeigt, dass das selbstständige Lernen (sammeln von Wissen) des Computers nicht ohne eine Grundlage aus von Hand kodiertem Wissen auskommt. Wir haben in Grundzügen gesehen, welche Möglichkeiten Techniken bieten, die auf eine ontologische Grundlage angewiesen sind. Doch ist weder die Entwicklung des Cyc – Projekts abgeschlossen noch die Tragweite der Anwendungen komplett sichtbar. Die momentane Entwicklung in der Cyc – Technologie zielt darauf ab, die Wissensbasis zu verbreitern. Zu diesem Zweck stellt Cycorp eine Opensource – Variante der Cyc – Technologie zur Verfügung. Diese soll es Interessierten ermöglichen, sich an der Erschaffung der Ontologie zu beteiligen. Was diese Entwicklung allerdings für Vorteile bringt, wird aber erst die Zeit zeigen. Zu finden ist ein kostenloser Download auf jeden Fall unter www.cyc.com oder www.opencyc.org.

Literatur:

- [1] Cycorp Homepage
<http://www.cyc.com>, 5.2004
- [2] Masters, James (2002)
'Structured Knowledge Source Integration and its applications
to information fusion'
ISIF, 2002. Masters, J.: Structured Knowledge Source Integration and its applications to
information fusion
- [3] O'Hara, Tom, Nancy Salay, Michael Witbrock, et al. (2003)
'Inducing criteria for mass noun lexical mappings
using the Cyc KB, and its extension to WordNet',
to appear in Proc. 5th Intl. Workshop on Computational Semantics,
Tilburg, 2003.
- [4] Lenat, Douglas B.: CYC: A Large – Scale Investment in Knowledge Infrastructure, in:
Communication of the ACM, Band 38, No.11 (November 1995), S. 33 – 38.
- [5] Homepage der Gesellschaft für Informatik
Ontologie(n)
<http://www.gi-ev.de/informatik/lexikon/inf-lex-ontologien.shtml>, 5.2004
- [6] Brockhaus Enzyklopädie in 20 Bänden, 17. Auflage, Band 13, Wiesbaden 1971. S. 742f.
- [7] Stuart Russell, Peter Norvig: Artificial Intelligence: A Modern Approach. Prentice Hall, 2003.