Robustes Parsen von Dialogen mit sematischen Grammatiken

Studienarbeit von
Christine Reck

Institut für Logik, Komplexität und Deduktionssysteme
Universtität Karlsruhe

9. April 1996

Inhaltsverzeichnis

1 Idee des semantischen Parsens zur Spracherkennung

1.1 Klassische Parser

Klassische Parser folgen im allgemeinen syntaktischen Prinzipien, d.h. sie versuchen, die Eingabe gemäß struktureller Regeln, die in in Form einer Grammatik vorgegeben sind, auszuwerten. Bei natürlicher Sprache ist dabei die Grammatik zunächst nichts anderes als die uns alle aus dem Sprachunterricht bekannte, mehr oder weniger logische Ansammlung von Regeln und Ausnahmen bzgl. Satzbau, Deklination, Konjugation etc. Diese vorgegebenen Regeln werden dann vereinfacht und in einer formalen Grammatik wiedergegeben. Basierend auf dieser formalen Grammatik findet das syntaktische Parsen statt.

1.2 Parsen gesprochener Sprache

Im besonderen Falle der Erkennung gesprochener Sprache bringt dieser Ansatz verschiedene Probleme mit sich. Die formale Grammatik wird nicht in der Lage sein, Sätze zu parsen, die grammatikalisch nicht korrekt sind, d.h. die nicht der im Duden vorgeschriebenen Form entsprechen. Gerade im Bereich der gesprochenen Sprache tauchen aber nicht korrekte oder unvollständige Sätze häufig auf. Erstaunlicherweise ist der Mensch in der Lage den Sinn unvollständiger oder nicht korrekter Sätze dennoch zu extrahieren. Der Mensch kann die Semantik eines Satzes auch dann erfassen, wenn Teile des Satzes weggelassen wurden oder grammatikalische Fehler auftauchen.

Für dieses Phänomen gibt es zwei Erklärungen: zum einen ist die natürliche Sprache an sich redundant, zum andern benutzt der Mensch zusätzliches Wissen über die Welt, den Kontext des Gesprächs, seinen Gesprächspartner usw. sowie zusätzliche Informationsquellen, d.h. er wertet nicht nur das aus, was er hört, sondern auch das, was er sieht (z.B. Gestik, Lippenbewegung, sichtliche Gefühlsregung seines Gesprächspartners) oder mit anderen Sinnesorganen wahrnimmt.

Es findet letztlich die Ausnutzung von zusätzlichem Wissen und eine, in Begriffen der Robotik gesprochen, Sensorfusion statt. Ersteres führte im Bereich der Spracherkennung zur Idee des semantischen Parsens, letzteres zur Entwichklung multimodaler Systeme.

1.3 Prinzipien des semantischen Parsens

Beim semantischen Parsen versucht man, Wissen über den Kontext des Gesprächs und die Welt in ihrer Gesamtheit auszunutzen. Probleme bereitet dabei das Wissen über die Welt. Es erscheint schwierig, all das Wissen, das der Mensch über einige Jahre oder Jahrzehnte angesammelt hat und ausnutzen kann, wiederzugeben und effizient zu speichern bzw. zu suchen, zumal die Mechanismen, denen der Mensch bei der Auswahl des entsprechenden Wissens folgt, noch weithin unbekannt sind. Offensichtlich muß man sich auf die Betrachtung einer 'Miniwelt' beschränken. Im hier betrachteten Beispiel handelt es sich dabei um die Miniwelt der Konferenzanmeldung. Man will Dialoge, die sich bei der telefonischen Anmeldung zu einer Konferenz ergeben, parsen, um das Ergebnis des Parses zur Übersetzung in eine andere Sprache (Englisch oder Japanisch) zu benutzen.

Im Gegensatz zum syntaktischen Parsen, wo Sätze, Teilsätze und Satzteile gemäß ihrer grammatikalischen Funktion klassifiziert werden (wie z.B. Hauptsatz, Nebensatz, Relativsatz, Subjekt, Objekt etc.), versucht man beim semantischen Parsen, semantische Konzepte zu entwickeln und Sätze oder Satzteile einem dieser Konzepte zuzuordnen. Die Auswahl der Konzepte und deren Ausformulierung ermöglicht es, Wissen über die Miniwelt einzubringen, und ist daher stark von ihr abhängig.

2 Aufbau und Funktionsweise des betrachteten Parsers

2.1 Benutzte Software

Für diese Studienarbeit wurde ein unter der Leitung von Wayne Ward an der Carnegie Mellon University entwickeltes Softwarepaket benutzt. Dieses benötigt im wesentlichen eine kontextfreie Grammatik in einer speziellen Form als Eingabe. Die Grammatikregeln werden in eine interne Struktur (Recursive Transition Networks) umgewandelt. Jedes einzelne Netz entspricht einem finiten Automaten. Die Netze können sich gegenseitig aufrufen und realisieren daher insgesamt einen Kellerautomaten. Die Grammatikregeln und deren Kombination zu Funktionen (vergleiche Abschnitt 2.4) werden dann zum Parsen von Sätzen verwendet.

2.2 Grammatikregeln

Eine Grammatikregel besitzt eine linke und eine rechte Seite (bzw. mehrere Alternativen für die rechte Seite). Die linke Seite besteht aus genau einem Nichtterminalzeichen. Die rechte Seite kann aus einem oder mehreren Zeichen bestehen, wobei sowohl Terminal- als auch Nichtterminalzeichen erlaubt sind. Terminalzeichen sind in diesem Zusammenhang deutsche Wörter. Alle auf der rechten Seite einer Regel auftauchenden Nichtterminalzeichen müssen durch eine Grammatikregel erklärt werden.

Zusätzlich sind auf der rechten Seite einer Regel lokale Nichtterminalzeichen erlaubt. Der einzige Unterschied zwischen einem lokalen Nichterminalzeichen und einem (globalen) Nichtterminalzeichen besteht darin, daß die Definition eines lokalen Nichtterminalzeichens auf die Grammatikregel, in der es verwendet wird, beschränkt ist.

Um Teile der rechten Seite einer Grammatikregel optional zu machen, wird der Operator * angegeben. Wird * auf der rechten Seite einer Regel vor ein Terminalzeichen geschrieben, so bedeutet dies, daß dieses Terminalzeichen im untersuchten Satz auftreten kann, aber nicht auftreten muß. So kann man z.B. in einer Grammatikregel, deren rechte Seite folgendermaßen aussehen soll: 'schicken sie mir das *bitte zu' das Wort bitte optional machen. Das Angeben des *-Operators vor einem Nichtterminalzeichen macht denentsprechend den Aufruf der zugehörigen Grammatikregel optional.

Nichtterminalzeichen, welche, außer in der sie definierenden Regel, nur auf der rechten Seite von Regeln auftauchen, werden als 'Funktionsslots' bezeichnet, alle übrigen als 'Top-Level-Slots'. Der Einfachheit halber werden im folgenden Top-Level-Slots einfach mit Slots bezeichnet.

2.3 Auswahl der Grammatikregeln

Wie oben erwähnt betrachtet man eine Miniwelt und nutzt das Wissen über diese Miniwelt gezielt aus. Dies kommt v.a. bei der Auswahl der Grammatikregeln zum Tragen. Eine Grammatikregel soll nämlich nichts geringeres repräsentieren als ein semantisches Konzept. Man muß sich also zunächst überlegen, welche semantischen Konzepte in der betrachteten Miniwelt häufig auftreten. Dies wurde im vorliegenden Fall durch Studium eines 204 Sätze umfassenden Datensatzes versucht. Da im Bereich der Konferenzanmeldung per Telefon das Zuschicken von Anmeldeformularen relativ oft auftritt wur-

den daher die Konzepte [send_object], [send_subj] und [to_send] entwickelt. [send_object] soll dabei möglichst all die Dinge enthalten, die im Zusammenhang Konferenzanmeldung verschickt werden können, [send_subj] all die Personen, die als Verschicker in Frage kommen und [to_send] möglichst all die Verben, die man benutzen kann, um 'schicken' auszudrücken (also z.B. senden, zusenden, etc.).

2.4 Kombination der Grammatikregeln zu Funktionen

Der nächste Schritt besteht darin, einzelne Grammatikregeln zu Funktionen, die man auch als 'Frames' bezeichnet, zusammenzufassen. Im vorliegenden Beispiel wurde z.B. eine Funktion 'send' generiert, die die oben erwähnten Slots (und einige andere) enthält. Generell versucht man die zu einem semantischen Konzept gehörigen Slots in einer Funktion zusammenzufassen. Ein- und derselbe Slot kann durchaus in verschiedenen Funktionen verwendet werden.

2.5 Funktionsweise des Parsers

Bei Eingabe eines Satzes geschieht in etwa folgendes: Das erste Wort des Satzes wird betrachtet. Alle Grammatikregeln (man stelle sich diese als endliche Automaten vor) werden daraufhin untersucht, ob dieses Wort vom Startzustand in einen Folgezustand führt. Falls ja, merkt man sich die Grammatikregel und den erreichten Zustand. Dann wird das nächste Wort des Satzes betrachtet. Wiederum untersucht man, welche der Grammatikregeln dieses Wort benutzen können, um vom Startzustand in einen Folgezustand zu gelangen. Für die bereits im vorigen Schritt gefundenen Automaten, untersucht man, ob man vom aktuellen Zustand (dies ist der, den man sich im vorigen Schritt gemerkt hatte) in einen Folgezustand kommen kann. Auf diese Art und Weise wird der gesamte Satz Wort für Wort bearbeitet. Gelangt einer der Automaten während der Satzanalyse in einen Finalzustand, so merkt man sich diesen Automaten. Die Automaten, die sich nach Abarbeitung des gesamten Satzes in einem Finalzustand befinden, stellen die Liste der Grammatikregeln dar, deren rechten Seiten ein matching mit dem ganzen Satz oder einem Teil des Satzes erlauben. Man kann sich also vorstellen, daß jede der so gefundenen Grammatikregeln einen Teil des Satzes parst.

Als nächstes muß man nun diejenigen Slots (Grammatikregeln) auswählen, die man für den Parse verwenden will. Waynes Parser legt dazu verschiedene Kriterien an. Zunächst dürfen sich die ausgewählten Slots nicht überschneiden, d.h. es darf nicht sein, daß zwei Slots ausgewählt werden, die ein-und dasselbe Wort aus dem zu parsenden Satz verwenden. Außerdem versucht man gleichzeitig die Anzahl der verwendeten Slots zu minimieren und die Anzahl der geparsten Wörter zu maximieren. Was die Maximierung der Anzahl der geparsten Wörter anbetrifft, ist zu bemerken, daß unbekannte Wörter (d.h. Wörter, die nicht in dem automatisch aus den Grammatikregeln erzeugten Lexikon vorkommen) einfach ignoriert werden. Bekannte Wörter, die zwischen zwei aufeinanderfolgenden ausgewählten Slots auftauchen, also von keinem der beiden verwendet werden, werden ebenfalls nicht geparst.

Ferner müssen die ausgewählten Slots alle in ein und derselben Funktion zusammengefaßt sein. Hierbei ist gleichgültig, in welcher Reihenfolge sie in dieser Funktion auftreten.

Man untersucht also letztlich für jede Funktion, wieviele der Slots, die einen Teil des Satzes parsen können, in dieser Funktion enthalten sind und wieviele Wörter man somit im gegebenen Satz mit dieser Funktion parsen kann. Die Funktion, die unter Verwendung einer minimalen Anzahl von Slots eine maximale Anzahl von Wörtern parsen kann, gewinnt den Wettbewerb und wird ausgewählt. Hierbei hat die Maximierung der Anzahl der geparsten Wörter höhere Priorität als die Minimierung der Anzahl der ausgewählten Slots.

3 Vorteile dieses Parsers

Bei der Entwicklung dieses Parsers war ein hohes Maß an Robustheit eines der wichtigsten Ziele. Der Parser soll in der Lage sein, auch Sätze zu erkennnen, die grammatikalisch nicht korrekt oder unvollständig sind. Ebenfalls sollen Wortwiederholungen innerhalb eines Satzes toleriert werden können.

3.1 Wortwiederholungen

Zwischen den einzelnen Slots einer Funktion, die ausgeähnt wurde, können beliebige Wörter im Satz auftreten, die im Parse dann ignoriert werden, insbesondere also auch Wortwiederholungen. Folgendes Beispiel soll diese Ei-

genschaft des Parsers etwas verdeutlichen. Der eingegebene Satz lautet: 'ich moechte mich zur konferenz fuer kuenstliche intelligenz am acht am acht und zwanzigsten achten zwei und neunzig anmelden'. Ganze Zahlen müssen dabei mit Leerzeichen eingegegben werden (z.B. wird achtundzwanzig zu acht und zwanzig). Dies ist eine spezielle Eigenschaft der Grammatik, die notwendig ist, um die Aufzählung aller Zahlen zu umgehen. Die Ausgabe des Parsers sieht aus wie folgt:

> **registration**:ich moechte mich zur konferenz fuer kuenstliche intelligenz *AM *ACHT am acht und zwanzigsten achten zwei und neunzig anmelden
>
> Interpretation score 17
>
> [i_want] ICH MOECHTE MICH
> [conference] [what_for] ZUR KONFERENZ FUER KUENSTLICHE INTELLIGENZ
> [top_level_date] [date] AM [dates] ACHT UND ZWANZIGSTEN
> [top_level_date2] [date] [dates] ACHTEN [integer] ZWEI UND NEUNZIG
> [top_level_register] [register] ANMELDEN

Der Parser gibt zunächst die ausgewählte Funktion an, hier also 'registration'. Dann wird der eingegebene Satz wiederholt. Dabei sind Wörter, die nicht zum Parse verwendet wurden in Großbuchstaben gedruckt. Im obigen Beispiel wurde demgemäß das erste Auftauchen von 'am acht' ignoriert. Der erste Teil des Satzes, 'ich möchte mich', konnte von einer Regel, deren linken Seite [i_want] ist, bearbeitet werden. Daher konnte der Slot [i_want] ausgewählt werden. Für den Satzteil 'zur konferenz fuer kuenstliche intelligenz' wurde entsprechend der Slot [conference] ausgewählt, der den Funktionsslot [what_for] benutzt. Das erste Auftreten von 'am acht' tritt zwischen den Slots [conference] und [top_level_date] auf und kann daher ignoriert werden. Für den letzten Teil des Satzes wird schließlich der Slot [top_level_register] ausgewählt, der wiederum den Funktionsslot [register] verwendet.

Die Ausgabe 'Interpretation score 17' bedeutet, daß 17 Wörter des Satzes zum Parse verwendet wurden.

Nach dem selben Prinzip können auch Wörter weggelassen werden, die keine Wiederholungen darstellen, sondern einfach bekannt, also im Lexikon

vorhanden sind, aber in dem gefundenen Parse nicht verwendet werden konnten.

Problematisch wird es allerdings, wenn bekannte Wörter nicht zwischen einzelnen Slots einer Funktion auftreten, sondern innerhalb eines Slots (vergleiche dazu Kapitel 4.1).

3.2 Unbekannte Wörter

Sollten in einem eingegegebenen Satz Wörter auftauchen, die nicht in dem automatisch aus den Grammatikregeln erzeugten Lexikon stehen, so können diese vom Parser ignoriert werden, unabhängig davon, an welcher Stelle im Satz sie auftreten, insbesondere also auch innerhalb eines Slots (nicht nur zwischen zwei Slots).Illustriert wird dies durch folgende Eingabe: 'ich moechte an der konferenz unbekanntes wort fuer kuenstliche intelligenz teilnehmen'. Die Ausgabe des Parsers sieht aus wie folgt:

> **registration**:ich moechte an der konferenz −UNBEKANNTES −WORT fuer kuenstliche intelligenz teilnehmen
>
> Interpretation score 9
>
> [i_want] ICH MOECHTE
> [wish_to_register] [what_for] AN DER KONFERENZ FUER KUENSTLICHE INTELLIGENZ [register] TEILNEHMEN

Die ausgewählte Funktion ist wiederum 'registration'. Die Wiederholung des eingegebenen Satzes in der Ausgabe zeigt, daß die beiden nicht im Lexikon stehenden Wörter 'unbekanntes' und 'wort' nicht im Parse verwendet wurden, da sie in Großbuchstaben abgedruckt sind. Das vorangestellte Minuszeichen bedeutet, daß die Wörter nicht im Lexikon sind.

Interessant ist, daß diese beiden Wörter ignoriert werden können, obwohl sie innerhalb des Slots [wish_to_register] auftreten.

3.3 Vertauschungen von Satzteilen

Sind die Satzteile eines Satzes vertauscht, so kann der Satz, falls die einzelnen Satzteile von Grammatikregeln bearbeitet werden können, trotzdem geparst werden. Ein Beispiel dafür liefert folgender Satz: 'an der konferenz fuer kuenstliche intelligenz teilnehmen ich moechte'. Dieser Satz liefert genau dieselbe Ausgabe wie der im vorigen Abschnitt angegebene.

3.4 Grammatikalische Fehler

Grammatikalische Fehler treten in gesprochener Sprache sehr häufig auf. Ein typisches Beispiel dafür ist der Satz: 'koennten sie mir ihre telefonnummer haben'. Offenbar wollte der Sprecher zunächst sagen: 'koennten sie mir ihre telefonnummer geben', hat dies dann aber nicht korrekt zu Ende gebracht, sondern mit dem Satz 'könnte ich ihre telefonnummer haben' vermischt. Der Mensch erkennt trotz des Fehlers sehr leicht, daß der Sprecher eine Telefonnummer haben möchte. Dazu benutzt der Mensch sowohl seine Erfahrung (er hat beide Satzkonstruktionen schon oft gehört) als auch sein Wissen über den Kontext des Gesprächs und die Miniwelt, in der es stattfindet.

Daß eine solche Konstruktion möglich ist, kann man dem Parser beibringen, indem man anstelle von 'geben' in manchen Sätzen auch 'haben' zuläßt.

Ist obiger Satz gegeben, so produziert der Parser erwartungsgemäß folgende Ausgabe:

> **name_and_address**: koennten sie mir ihre telefonnummer haben.
>
> Interpretation score 6
>
> [top_level_request] [request] KOENNTEN SIE MIR
> [phone_fax_number] IHRE TELEFONNUMMER
> [give] HABEN

Das Wissen, das der Parser benutzt, steckt also in den Grammatikregeln und wird vom Designer der Grammatik, der seine Erfahrung über die Welt und sein Wissen über die Miniwelt ausnutzt, in diese hineincodiert.

Ein anderer typischer Fehler wird mit Hilfe des Satzes: 'kann ich mich anmelde' illustriert. Bei diesem Satz kann es sowohl sein, daß der Sprecher den letzten Buchstaben verschluckt hat, als auch, daß der Erkenner einen Fehler gemacht hat. Die Fehlerquelle ist für den Parser jedoch unwichtig. Die Ausgabe, die der Parser liefert, ist wie erwartet:

> **registration**: kann ich mich anmelde
>
> Interpretation score 4
>
> [top_level_request] [request] KANN ICH MICH
> [top_level_register] [register] ANMELDE

3.5 Ambiguitäten

Mitunter ist es nicht möglich die Bedeutung eines Satzes eindeutig festzulegen durch die ausschließliche Betrachtung seines grammatikalischen Aufbaus (wie beim syntaktischen Parsen). Insbesondere läßt sich häufig nicht klären, worauf sich angehängte Präpositionalphrasen beziehen. Der Satz 'ich moechte mich zur konferenz am dritten januar anmelden' hat, theoretisch betrachtet, zwei mögliche Bedeutungen: einmal bezieht sich 'am dritten januar' auf den Zeitpunkt des Anmeldens, das andere Mal auf den Zeitpunkt, zu dem die Konferenz stattfindet. Ein syntaktischer Parser wird hier also eine Ambiguität finden.

In der Miniwelt der telefonischen Konferenzanmeldung ist allerdings klar, daß ein Anrufer sich entweder sofort anmeldet oder gegebenenfalls später noch einmal anruft, um sich anzumelden, keinesfalls aber den Zeitpunkt, zu dem er sich anmelden wird, bekannt gibt. Man kann davon ausgehen, daß man in diesem Fall in den allermeisten Fällen richtig liegt, wenn man die Präpositionalphrase auf die Konferenz bezieht.

Die Ausgabe des Parsers ergibt:

> **registration**: ich moechte mich zur konferenz am dritten januar anmelden.
>
> Interpretation score 9
>
> [i_want] ICH MOECHTE MICH
> [wish_to_register] [what_for] ZUR KONFERENZ[top_level_date] [date] AM [dates] DRITTEN[month] JANUAR[register] ANMELDEN

Dabei ist zu beachten, daß [top_level_date] auf der rechten Seite der Regel für [what_for] auftaucht, d.h. von dieser aktiviert wird. [date], [dates] und [month] werden entsprechend von [top_level_date] aus aktiviert.Damit wird die Präpositionalphrase eindeutig als zur Konferenz gehörig betrachtet.

4 Probleme mit diesem Ansatz

4.1 Einfügungen

Ein großes Problem für den Parser stellen Einfügungen dar. Wie oben bereits erwähnt, ist das Einfügen von Wörtern, die nicht im Lexikon stehen, völlig

unkritisch. Problematisch wird es nur, wenn man an sich bekannte Wörter einfügt, die in der Regel, die eigentlich angewendet werden sollte, nicht bedacht wurden. Klar wird dieser Sachverhalt an folgendem Beispiel:'wieviel kostet ein einzelzimmer' liefert:

> **cost**: wieviel kostet ein einzelzimmer
>
> Interpretation score 4
>
> [how_much] WIEVIEL [cost_verb] KOSTET [cost_object] EIN EINZELZIMMER

Andererseits liefert der Satz: 'wieviel kostet aber ein einzelzimmer':

> **stay_over_night**: *WIEVIEL *KOSTET *ABER ein einzelzimmer
>
> Interpretation score 2
>
> [rooms] EIN EINZELZIMMER

Der erste Satz wird problemlos geparst. Es wird auf oberster Ebene nur der Slot [how_much] aktiviert, der seinerseits die Funktionsslotslots [cost_verb] und [cost_object] aktiviert. Der zweite unterscheidet sich vom ersten nur dadurch, daß das Wort 'aber' eingefügt wurde. Durch diese minimale Veränderung des Satzes ist der im ersten Satz verwendete Parse nicht mehr möglich. Dies rührt daher, daß 'aber' ein bekanntes Wort ist, in der Grammatikregel [how_much] jedoch an dieser Stelle nicht vorgesehen wurde. Daher kann die Regel für [how_much] überhaupt nicht angewandt werden. Anstelle von [how_much] wird der Slot [rooms] ausgewählt, der in der Funktion stay_over_night auftaucht.

Offenbar gibt der zweite Parse den Inhalt des Satzes sehr viel schlechter wieder als der erste. Eine mögliche Lösung für dieses Problem besteht darin, das Wort 'aber' (und am besten auch alle vergleichbaren Wörter, die an dieser Stelle des Satzes auftauchen könnten) in die Grammatikregel für [how_much] aufzunehmen. Zum einen wird man es jedoch nie schaffen alle Möglichkeiten zu bedenken, und zum anderen ist dieses Wort ('aber') semantisch gesehen völlig unwichtig. D.h. man möchte den Satz trotz Einfügungen, die nicht bedacht wurden, richtig parsen können.

4.2 Zusammengesetzte Verben

Ein weiteres Problem sind zusammengesetzte Verben, die z.B. in der Imperativform aufgespaltet werden ('einwerfen' wird zu 'werfen sie ein', 'wegwerfen' wird zu 'werfen sie weg'). In Fällen, wo der zweite Teil des Verbs nicht unbedingt gebraucht wird, um den Sinn des Satzes zu erfassen, ist dies völlig unkritisch. Man wird im Parse einfach den zweiten Teil des Verbs ignorieren. Ein Beispiel hierfür ist der Satz: 'senden sie mir bitte die zusammenfassung zu'. Der Parser liefert als Ausgabe:

> **registration**: senden sie mir bitte die zusammenfassung *ZU
>
> Interpretation score 6
>
> [imperative] [imperative_verb] SENDEN [imperative_subject] SIE [to_whom] MIR BITTE [imperative_object] DIE ZUSAMMENFASSUNG

Der zweite Teil des Verbs wird ignoriert und der Sinn des Satzes trotzdem korrekt bestimmt.

Bei den Verben wegwerfen bzw. einwerfen ist der zweite Teil des Verbs eben sehr wohl entscheidend für die Bedeutung. Obige Lösung ist also in diesem Fall nicht angebracht.

Will man den zweiten Teil des Verbs parsen können, so muß man eine Regel aufstellen, die (z.B. für den Imperativ) in etwa folgendermaßen aussieht:

```
[imperative]
    ( [imperative_verb] [imperative_subject] *[to_whom]
     *[imperative_object] [imperative_verb_part2] )
```

Die linke Seite der Regel ist [imperative]. Die rechte Seite besteht aus dem Aufruf weiterer Regeln. Der Satz füllen sie bitte ein Anmeldeformular aus, wird damit folgendermaßen geparst:

> **registration**:
>
> fuellen sie bitte ein anmeldeformular aus
>
> Interpretation score 6
>
> [imperative] [imperative_verb] FUELLEN [imperative_subject] SIE BITTE [imperative_object] [form] EIN ANMELDEFORMULAR [imperative_verb_part2] AUS

Das Parsen dieses Satzes funktioniert offenbar nur deshalb, weil 'füllen' als imperative_verb zugelassen ist, (d.h. in der Regel für [imperative_verb] auf der rechten Seite als eine Möglichkeit zugelassen ist) 'sie' als imperative_subject zugelassen ist 'ein anmeldeformular' als imperative_object zugelassen ist und schließlich 'aus' als imperative_verb_part2 zugelassen ist. Die Schwierigkeit bei der Entwicklung einer solchen Regel ist, alle möglichen Imperativkonstruktionen, d.h. insbesondere alle Verben in Imperativform, alle Subjekte, alle Objekte (dies ist ganz besonders schwierig), die in einem Imperativsatz auftauchen könnten in die Regeln mitaufzunehmen.

Es erscheint nahezu unmöglich eine solche 'Klammerregel', also eine Regel, deren erster und letzter Teil (hier: erster Teil des Verbs und zweiter Teil des Verbs) zusammen eine Art Klammer bilden, so zu formulieren, daß sie tatsächlich für viele Sätze und nicht nur für die Testdaten funktioniert.

Verstärkt wird dieses Problem dadurch, daß, wie oben erwähnt, Einfügungen problematisch sind. Kommt nur ein einziges bekanntes Wort in obigem Satz vor, das beim Entwurf der Grammatikregel für [imperative] nicht bedacht wurde, so kann die Regel nicht angewendet werden für den Parse.

Robustheit bezüglich Einfügungen besteht also nur auf oberster Ebene, nämlich auf Funktionsebene. Zwischen den ausgewählten Slots einer Funktion können beliebige Wörter auftauchen, die im Parse ignoriert werden können, aber innerhalb eines Slots führt dies zu Schwierigkeiten. D.h. gleichzeitig, daß Gruppierung von Grundbausteinen zu komplexeren Strukturen entweder nur auf Funktionsebene durchgeführt wird, und damit eine gute Strukturierung nahezu unmöglich wird, oder Regeln wie die für [imperative] eingeführt werden, die keine große Robustheit aufweisen, sondern meist sehr genau auf den betrachteten Datensatz zugeschnitten sind. Beides ist letztenendes nicht befriedigend.

4.3 Deklination und Konjugation

Ein weiteres Problem, das v.a. in der deutschen Sprache sehr ausgeprägt ist, kommt durch die im Deutschen relativ zum Englischen gesehen komplizierte Konjugation und Deklination zustande. Während sich im Englischen im Präsenz nur die Verbform für die dritte Person Singular von den anderen unterscheidet, unterscheiden sich im Deutschen meist außer der Verbform für erste und dritte Person Plural alle Verbformen voneinander. Dies bedeutet, daß all diese Verbformen in die Regeln und somit ins Lexikon mitaufgenom-

men werden müssen. Bei Substantiven unterscheiden sich die Endungen je nach Fall. Der Designer der Grammatik muß daher bei Entwurf der Regeln versuchen, alle möglicherweise auftretenden Endungen mit in die entsprechende Grammatikregel aufzunehmen. Dabei ist zu beachten, daß man z.B. an einer Stelle im Satz, an der eigentlich ein Akkusativ stehen müßte auch in der Lage sein will einen Dativ zu akzeptieren, um den Parser robust und unempfindlich gegenüber grammmatikalischen Fehlern zu machen.

4.4 Nicht lösbare Ambiguitäten

Immer wieder treten nicht lösbare Ambiguitäten auf. Ein Beispiel hierfür sind die beiden Sätze: 'gibt es einen rabatt fuer mitglieder' und 'einen rabatt fuer mitglieder gibt es'. Der erste satz ist eindeutig eine Frage. Der zweite Satz könnte sowohl eine Frage sein als auch eine Aussage. Es erscheint jedoch wahrscheinlicher, daß es sich um eine Aussage handelt. Für beide Sätze erhält man aber denselben Parse:

> **discount**: gibt es einen rabatt fuer mitglieder (bzw. einen rabatt fuer mitglieder gibt es)
>
> Interpretation score 6
>
> [is_there_question] GIBT ES
> [discount] EINEN RABATT FUER
> [member] MITGLIEDER

Man kann an dieser Stelle nicht unterscheiden, daß es sich einmal um eine Aussage, das andere Mal um eine Frage handelt. Dies rührt daher, daß man nicht bestimmen kann, an welcher Position im Satz ein Slot auftauchen muß. Man kann nicht sagen, ein Satz, in welchem 'gibt es' vorkommt ist eine Frage, falls dies am Anfang des Satzes steht und ansonsten eine Aussage.

Läßt man 'gibt es' auch als Aussage zu und generiert dafür einen Slot, so löst man damit das Problem nicht, sondern überläßt es dem Zufall bzw. der Implementierung, welcher Slot vorzugsweise ausgewählt wird.

Dies ist selbstverständlich nicht die einzige auftretende Ambiguität, die mit den vorhandenen Möglichkeiten nicht zu lösen ist, sondern sollte nur zur Illustrierung einer solchen dienen.

4.5 Auswahl einer falschen Funktion

Die Grammatik wurde zunächst basierend auf 204 Testsätzen erstellt. Anschließend wurden Lexikonerweiterungen vorgenommen. Danach wurden dem Parser etwa 900 neue Sätze präsentiert. Auffällig war dabei, daß der Parser sehr oft nicht die gewünschte Funktion auswählte. Dies war einerseits dann der Fall, wenn insgesamt nur sehr wenige Wörter geparst wurden und ist in diesem Fall wohl auch nicht zu verhindern. Aber auch wenn relativ viele Wörter erkannt wurden, trat das Phänomen auf. Dies läßt sich dadurch erklären, daß dem Parser als Eingabe sehr oft nicht ein einzelner Satz, sondern zwei oder mehr Sätze präsentiert wurden. Diese werden dann aber behandelt, als seien sie ein Satz. Daraus resultieren Funktionen, die zu allgemein sind, da sie Slots beinhalten, die eigentlich gar nicht in ihren Bereich fallen.

Dieses Problem ist daher nicht Parser-inhärent, sondern müßte durch geschickte Vorverarbeitung lösbar sein.

5 Verbesserungen

5.1 Änderungen an der Parsersoftware

Einige der im letzten Kapitel vorgestellten Probleme könnte man sicher verbessern durch Änderungen an der Parsersoftware.

Das Problem mit Einfügungen könnte erheblich gemildert werden, wenn innerhalb eines Slots nicht nur eingefügte unbekannte Wörter, sondern auch eingefügte bekannte Wörter erlaubt wären. Dies würde gleichzeitig das Schreiben von Klammerregeln sehr erleichtern.

Das Problem der relativ komplizierten Konjugation und Deklination hat zunächst mit dem benutzten Parseansatz wenig zu tun. Es wäre allerdings einfacher, wenn man nicht alle Verbformen einzeln in die Regeln hineinschreiben müßte, sondern eine Morphologiekomponente in die Parsersoftware integrieren könnte. Eine ähnliche Lösung wäre auch für Deklinationen denkbar.

Die vorgestellte nicht lösbare Ambiguität bei is_there_question wäre lösbar, wenn man innerhalb einer Funktion die Position eines Slots im geparsten Satz verbindlich angeben könnte. Man könnte im vorliegenden Fall fordern, daß ein Satz mit 'gibt es' nur dann eine is_there_question ist, wenn 'gibt es' am Satzanfang auftaucht. In den übrigen Fällen würde man einen solchen Satz

als Aussage interpretieren. Eine Restfehlerquote bleibt selbstverständlich erhalten, da man in manchen Fällen, (z.B. einen rabatt fuer mitglieder gibt es) ohne die Intonation des Sprechers zu kennen, nicht zweifelsfrei entscheiden kann, ob es sich um eine Frage oder um eine Aussage handelt.

Das sicherlich wichtigste Problem ist die Auswahl falscher Funktionen. Dies könnte zum einen dadurch gemildert werden, daß immer nur genau ein Satz präsentiert wird, zum anderen durch obligatorische Slots. Es müßte möglich sein zu spezifizieren, welche Slots einer Funktion im Parse vorhanden sein müssen, damit diese ausgewählt werden kann.

5.2 Nachverarbeitung

Zusätzlich sollte an die Ausgabe des Parsers unbedingt eine Nachverarbeitung angeschlossen werden, die entscheidet, ob in einem Satz überhaupt so viel geparst werden konnte, daß mit einer sinnvollen Ausgabe des Parsers zu rechnen ist. Dies könnte dadurch geschehen, daß man festlegt, wie viele Wörter prozentual im Parse sein müssen. Eine andere, in meinen Augen bessere, Möglichkeit bestünde darin, eine Liste von semantisch unverzichtbaren Wörtern zu erstellen und dann den gefundenen Parse daraufhin zu untersuchen, ob ein oder mehr nicht geparste Wörter in dieser Liste auftauchen. Falls ja, sollte der gefundene Parse abgelehnt werden. Weitere, vom Parser schlechter bewertete Analysen können dann daraufhin untersucht werden, ob sie dem Vergleich mit der Liste standhalten. Beispiel für ein Wort, welches auf alle Fälle auf einer solchen Liste auftauchen müßte, ist das Wort 'nicht'.

6 Ausblick

Rückblickend betrachtet ist zu erkennen, daß die Erstellung einer robusten Grammatik mit sehr hohem Aufwand verbunden ist. Die Grammatikregeln tendieren dazu zu speziell auf die Testdaten abgestimmt zu sein. Zusätzlich wächst die Menge der benutzten Regeln sehr schnell an, so daß die Grammtik leicht unübersichtlich wird.

Es wäre mit Sicherheit besser, Software zu entwickeln, die, basierend auf statistischen Methoden und unter Ausnutzung eines sehr großen Datensatzes, die Generierung der semantischen Konzepte automatisch vornimmt. Dies

wäre auch im Hinblick auf die Übertragbarkeit der Methode auf andere Miniwelten ein erstrebenswertes Ziel.

Ob dieser Ansatz letztenendes weiterentwickelt werden sollte, um noch eine grobe Analyse zu liefern, falls andere Parsemethoden fehlgeschlagen sind, hängt sicherlich sehr stark davon ab, ob es möglich sein wird, die Software zu verbessern.