UBS Eine Unifikationsbasierte Sprache zur Implementation von HPSG

FRIEDER STOLZENBURG; MARTIN VOLK Universität Koblenz-Landau Institut für Computerlinguistik Rheinau 3-4 *5400* Koblenz 0261-9119-469 E-Mail volk@brian.uni-koblenz.de

1 Einleitung

In der Computerlinguistik gewinnen unifikationsbasierte Ansätze immer größere Bedeutung für die Formulierung von Grammatiktheorien. Denn die Benutzung der Unifikation erlaubt es erstens, eine Vielzahl von Phänomenen der Syntax und zum Teil auch der Semantik natürlicher Sprachen elegant zu beschreiben; zweitens läßt sich die Unifikation mit mathematischer Exaktheit definieren, so daß die linguistischen Theorien auch als Computerprogramme implementiert werden können. Die in diesem Artikel vorgestellte Arbeit beschäftigt sich mit dem Formalismus von HPSG ("Head-Driven Phrase Structure Grammar", Pollard & Sag 1987) und mit dem Problem, ihn vollständig zu formalisieren und zu implementieren. Die zu diesem Zweck einsetzbare Programmiersprache UBS ("Unifikationsbasierte Sprache") ist im Rahmen einer Studienarbeit (Stolzenburg 1991) entwickelt und in ARITY /PROLOG 5.1 (Arity Corporation 1988) implementiert worden.

Der Formalismus von HPSG ist vielseitig. Er umfaßt mehr Datenstrukturen und Operationen als bisherige Grammatikformalismen. Zu nennen sind insbesondere Negation, Mengen und allgemeine Funktionen, die zusätzlich zu den grundlegenden Strukturen, der Unifikation und den Merkmalstrukturen hinzukommen.

2 Der Ansatz in GULP

Heutzutage gibt es bereits eine ganze Reihe von Werkzeugen, die es ermöglichen, Grammatiken, die in einem unifikationsbasierten Formalismus beschrieben sind, mit dem Computer zu bearbeiten und an Sätzen der natürlichen Sprache zu erproben. Zu diesem Zweck sind formale Sprachen entwickelt worden, die als Computerprogramm in einer bestimmten Programmiersprache implementiert sind.

Die Verwendung einer solchen formalen Sprache zur Beschreibung von Grammatiken bringt jedoch einige Nachteile mit sich: Ein Anwender, der eine Grammatik entwickeln will, muß zunächst die Syntax und andere Eigenschaften der Sprache lernen. Außerdem ist er in seinen Ausdrucksmöglichkeiten auf die Elemente der formalen Sprache beschränkt. Solche Sprachen besitzen meist nur gerade die Funktionen und Datenstrukturen, die unbedingt zur Behandlung einer Grammatik nötig sind, aber sie besitzen nicht die Mächtigkeit einer allgemeinen Programmiersprache.

Der Ansatz von Covington (1989) versucht, diese Nachteile zu beseitigen: Covington hat die Syntax von PROLOG so erweitert, daß Merkmalstrukturen leicht formuliert werden können. Alle PROLOG-Möglichkeiten bleiben erhalten. Der PROLOG-Interpreter braucht nicht verändert zu werden. Covington nennt sein System GULP ("Graph Unification Logic Programming").

Beim Einlesen von Klauseln in PROLOG aus einer Datei werden Merkmalstrukturen, für die eine besondere Schreibweise eingeführt wird, in ein internes Format, sogenannte Wertlisten (englisch "value lists"), überführt. Dadurch wird die Unifikation von Merkmalstrukturen auf die in PROLOG schon vorhandene Unifikation von Termen zurückgeführt, ohne daß zusätzlicher Code in die Klauseln eingefügt werden muß.

3 Die Sprache UBS

Bei der Entwicklung der Sprache UBS sind die gleichen Absichten wie in GULP verfolgt worden. Es handelt sich bei UBS um eine Erweiterung von GULP, die dem Anwender alle Ausdrucksmöglichkeiten des Formalismus von HPSG zur Verfügung stellen soll. Zur Behandlung der in HPSG verwendeten Strukturen muß zum Teil aber zusätzlicher Code in die Klauseln von PROLOG-Programmen, die UBS verwenden, eingefügt werden. Die Details dieses Vorgangs sollen im folgenden erläutert werden.

Wird ein Programm in UBS geladen, so wird es Term für Term gelesen, dann transformiert und der Wissensbasis hinzugefügt, nachdem die HPSG-spezifischen Konstrukte in reguläres PRO-

LOG übersetzt worden sind. Sie können nur auf den Argumentpositionen von Prädikaten vorkommen.

Zwei Beispiele sollen die Behandlung von erweiterten Merkmalstrukturen illustrieren. Sie sind bewußt einfach gehalten, um dem Leser das Verständnis der Arbeitsweise von UBS nicht unnötig zu erschweren. Selbstverständlich kann UBS auch mit komplizierten Merkmalstrukturen umgehen; diese werden einfach in ihre Bestandteile zerlegt, und die Transformation wird Stück für Stück durchgeführt.

BEISPIEL 1: Der Ausdruck `a#b` steht für alle Ausdrücke, die mit `a` oder mit `b` unifizierbar sind. Da es sich in beiden Fällen um Konstanten handelt, sind dies gerade die beiden Ausdrücke `a` und `b` selbst. Betrachtet werden soll nun ein Programm, das aus nur einer einzigen Klausel mit dem einstelligen Prädikat `test1` besteht. Das Prädikat `test1` soll dann wahr sein, wenn sein einziges Argument gleich `a` oder `b` ist. Das Quellprogramm lautet also:

```
test1(a # b).
```

Um das gewünschte Verhalten des Programms zu bewirken, wird es in ein reguläres PROLOG-Programm transformiert. Der Ablauf der Transformation ist ungefähr der folgende:

1. Der Ausdruck `a#b` in UBS wird in einen regulären PROLOG-Ausdruck übersetzt, und zwar in diesem Fall in eine ungebundene Variable:

   ```
   X
   ```

2. Es wird Code zum Berechnen des Ausdrucks `a#b` erzeugt, und zwar:

   ```
   (X = a ; X = b)
   ```

   (Dabei ist ; der ODER-Operator von PROLOG.)

3. Der Code wird in die ursprüngliche Klausel eingeführt, so daß das transformierte Programm nun wie folgt aussieht:

   ```
   test1(X) :- (X = a ; X = b).
   ```

4. Tatsächlich ist das entstehende Programm noch etwas komplizierter. Den ursprünglichen Prädikaten werden nämlich drei zusätzliche Argumente angefügt, die zur Behandlung der Negation, von Mengen und von Funktionswerten gebraucht werden, so daß das Zielprogramm letztendlich die folgende Gestalt hat:

   ```
   test1(X, NEG, SET, FUN) :-
           (X = a ; X = b).
   ```

BEISPIEL 2: Nun soll ein Programm mit Negation betrachtet werden, das den Parameter `NEG` gebraucht. Es soll die Ausdrücke erkennen, die nicht mit der Konstanten `c` unifizierbar sind. Es kann in UBS ausgedrückt werden als

```
test2(^c).
```

und wird transformiert in

```
test2(X,NEG,SET,FUN) :- neg_(NEG,X,c).
```

Die Negation ist in UBS mit dem Prädikat `neg_/3` als konstruktive Negation (Walinsky 1987) implementiert, die auf einer dreiwertigen Logik basiert. Das Prädikat `neg_/3` überprüft, ob seine beiden letzten Argumente miteinander unifizierbar sind. Bei der Auswertung der Negation kann es jedoch vorkommen, daß es nicht entschieden werden kann, ob zwei Ausdrücke miteinander unifizierbar sind ("true"- oder "success"-Fall) oder nicht ("false"- oder "fail"-Fall), nämlich im Zusammenhang mit ungebundenen Variablen ("undef"-Fall). Im letzten Fall wird das Paar `(X,c)` in einer offenen Liste aufbewahrt, die an die Variable `NEG` gebunden ist, um später noch einmal untersucht zu werden.

Die drei Fälle sollen nun noch anhand von Beispielen erläutert werden. Gleichzeitig wird mit ihnen gezeigt, wie Programme in UBS aufgerufen werden können.

ANFRAGE:	ANTWORT:	FALL:
test2(c,NEG,_,_).	no	"true"
test2(d,NEG,_,_).	yes NEG = _	"false"
test2(X,NEG,_,_).	yes NEG = [(X,c)—_]	"undef"

(_ ist eine sogenannte anonyme Variable in PROLOG.)

Die Negation von komplexen Ausdrücken, z.B. Merkmalstrukturen, ist nicht allein mit dem Prädikat `neg_/3` lösbar. Um sie zu ermöglichen, werden komplexe Ausdrücke mit Negation umgeformt in äquivalente Ausdrücke, die nur noch negierte Konstanten enthalten.

4 Die Implementation

Bei der Implementation von UBS sind die wichtigsten Konstrukte von HPSG integriert worden. Dabei traten aber auch einige Probleme auf. Hier ein kurzer Überblick über Datenstrukturen und Operationen in UBS:

1. Datenstrukturen

▷ Die grundlegende Datenstruktur im Formalismus von HPSG ist die MERKMALSTRUKTUR. Jede Art von linguistischer Information wird mit ihrer Hilfe beschrieben. Merkmalstrukturen können in UBS (analog zu GULP) als Merkmal-Wert-Paare in beliebiger Reihenfolge formuliert werden.

> Eine LISTE ist eine möglicherweise leere Folge von Beschreibungen meist ähnlicher Objekte, in HPSG z.B. die Valenzpartner eines Verbs in den Angaben zur Subkategorisierung des Verbs. In UBS können PROLOG-Listen als Werte von Merkmalen auftreten.

> In einer MENGE werden wie bei Listen mehrere Beschreibungen zusammengefaßt. Im Gegensatz zu Listen spielt bei Mengen die Reihenfolge der Elemente keine Rolle. Außerdem kann es sein, daß mehrere gleiche Beschreibungen dasselbe Objekt bezeichnen, also identisch sind. Die Formalisierung der Unifikation von Mengen erfolgt in UBS mit Hilfe von surjektiven Funktionen. Die Unifikation von Mengen stellt eine Erweiterung des Begriffs der Unifikation von Merkmalstrukturen dar. Pollard und Sag (1987) liefern keine formale Definition; Hinweise liefern Büttner (1986) und Kapur und Narendran (1986). Die Unifikation von Mengen ist in UBS noch nicht vollständig implementiert worden.

> Der Formalismus von HPSG läßt die Einführung von (Zeichen-) TYPEN zu. Sie sind hierarchisch angeordnet: Zeichen allgemein sind vom Typ "sign". Sie werden eingeteilt in lexikalische Zeichen und phrasale Zeichen. Ein typorientierter Ansatz zur Implementation von HPSG wird von Franz (1990) vertreten. Er betrachtet HPSG als eine Theorie, die aus Zeichentypen und damit verbundenen Bedingungen (englisch "constraints") besteht. In UBS ist ein komplementärer Ansatz verfolgt worden: Er ist mehr unifikationsbasiert. Typen sind nicht explizit verfügbar, denn PROLOG kennt auch keine Typen. UBS stellt jedoch MAKROS zur Verfügung, mit denen man die Notation von Merkmalstrukturen abkürzen kann. Mit Hilfe dieser MAKROS lassen sich auch Typen simulieren.

2. Operationen

0 Die grundlegende Operation im Formalismus von HPSG ist die UNIFIKATION. Die bekannte Definition (Shieber 1986) wird in UBS erweitert auf Listen und Mengen. Die Unifikation kann in UBS beliebig innerhalb von Merkmalstrukturen durchgeführt werden.

0 Zusätzlich eingeführt wird die DISJUNKTION von Werten. Sie wird verwendet, um auszudrücken, daß ein Attribut mehrere mögliche Werte haben kann, der Wert also nicht eindeutig festgelegt ist (vgl. Beispiel 1).

0 Auch die NEGATION von Werten ist möglich. Ein negierter Wert bedeutet, daß an dieser Stelle stehen darf, was nicht mit diesem Wert unifiziert werden kann. Die Interpretation der Negation ist problematisch: Sie kann nicht als "Negation by Failure" realisiert werden, denn das würde im Zusammenhang mit ungebundenen Variablen zu unerwünschten Ergebnissen führen (vgl. Beispiel 2). Die Negation ist deshalb in UBS als konstruktive Negation implementiert worden.

0 Ferner kann auch die IMPLIKATION ausgedrückt werden. Sie wird in UBS wie in der klassischen Logik auf Negation und Disjunktion zurückgeführt.

0 Schließlich können Merkmalwerte durch FUNKTIONEN berechnet werden. Z.B. können zwei Listen durch die zweistellige Funktion "append" aneinandergefügt werden. Das Ergebnis dieser Funktion ist also die Konkatenation der beiden Listen und kann als Wert einem Merkmal zugewiesen werden. Der Gebrauch von allgemeinen Funktionen erlaubt es, praktisch jede beliebige Manipulation an Merkmalwerten vornehmen zu können. In UBS ist die Möglichkeit geschaffen worden, beliebige PROLOG-Prädikate einzubinden. Sie berechnen die Funktionen, die auch allgemeine Relationen sein können, und werden verzögert ausgeführt.

5 Ausblick

UBS ist bereits im Rahmen einer Diplomarbeit an der Universität Koblenz erfolgreich eingesetzt worden, um ein Grammatikfragment des Deutschen in HPSG zu formalisieren. Dennoch soll UBS in nächster Zukunft erweitert werden: So ist daran gedacht, Typen explizit zur Verfügung zu stellen. Das würde die Effizienz bezüglich Zeit und Speicherplatzverbrauch steigern.

Wir haben versucht, die vielschichtigen Überlegungen bei Entwurf und Implementation einer formalen Sprache zur Behandlung von HPSG - einer linguistischen Theorie - zu skizzieren. Weitere Untersuchungen sind notwendig, um die praktische Verwertbarkeit von UBS auszutesten.

Literatur

[1] The Arity /Prolog Language Reference Manual. Concord, Massachusetts: Arity Corporation, 1988.

[2] Büttner, Wolfram: Unification in the Data Structure Sets. In: Goos, G.; Hartmanis, J. (Hrsg.): Proceedings of the 8th International Conference on Automated Deduction, Oxford, 1986, 489-495. (LNCS 230) Berlin; Heidelberg: Springer, 1986.

[3] Covington, Michael A.: GULP 2.0: An Ex tension of Prolog for Unification-Based Gram

mar. (Research Report AI-1989-01) Athens, Georgia: The University of Georgia. 1989.

[4] Franz, Alex: A Parser for HPSG. (CMULCL-90-3) Pittsburgh: Carnegie Mellon University, Laboratory for Computational Linguistics. Juli 1990.

[5] Kapur, Depak; Narendran, Paliath: NP - Completeness of the Set Unification and Matching Problems. In: Goos, G.; Hartmanis, J. (Hrsg.): Proceedings of the 8th International Conference on Automated Deduction, Oxford, 1986, 470-488. (LNCS 230) Berlin; Heidelberg: Springer, 1986.

[6] Pollard, Carl; Sag, Ivan A.: Information Based Syntax and Semantics. Volume 1: Fundamentals. (CSLI Lecture Notes 13) Leland Stanford Junior University: CSLI. 1987.

[7] Shieber, Stuart M.: An Introduction to Unification Based Approaches to grammar. (CSLI Lecture Notes 4) Leland Stanford Junior University: CSLI. 1986.

[8] Stolzenburg, Frieder: UBS - Eine unifikationsbasierte Sprache zur Implementation von HPSG. Studienarbeit. Koblenz: Universität Koblenz-Landau. 1991.

[9] Walinsky, Clifford: Constructive Negation in Logic Programs. Dissertation. Oregon Graduate Center. 1987.