Gefördert vom
bmb+f
Bundesministerium für
Bildung, Wissenschaft,
Forschung und Technologie

Verbmobil
Verbundvorhaben

Lexikonakquisitionstools für den Lexikonformalismus LeX4

Johannes Heinecke

Humboldt-Universität zu Berlin

Technisches Dokument 42
März 1996

März 1996

Johannes Heinecke

Humboldt-Universität zu Berlin
Philosophische Fakultät II
Institut für deutsche Sprache und Linguistik
Computerlinguistik
Jägerstr. 10/11
10099 Berlin

Tel.: (030) 20192 - 553
Fax: (030) 20192 - 560
e-mail: heinecke@compling.hu-berlin.de

Die vorliegende Arbeit wurde im Rahmen des Verbundvorhabens Verbmobil vom Bundesministerium für Bildung, Wissenschaft, Forschung und Technologie (BMBF) unter dem Förderkennzeichen 01 IV 101 G gefördert. Die Verantwortung für den Inhalt dieser Arbeit liegt bei dem Autor.

Inhaltsverzeichnis

1 Vorwort

Mit dem Lexikonformalismus LeX4 [2] steht ein ausdrucksmittelstarker Verarbeitungs- und Verwaltungsformalismus zur Verfügung. Lexikonstrukturen können mit ihm verwaltet und generiert, Daten können in nahezu beliebige Strukturen transformiert werden.

Bis jetzt war es jedoch ausschließlich möglich, LeX4-base-Terme „manuell“ (z.B. mit einem Texteditor wie emacs) einzugeben, und diese Daten anschließend vom Lexikonformalismus auf syntaktische Korrektheit zu überprüfen. Eine Reihe von Fehlern jedoch (z.B. ungewollte Unterspezifikation eines Merkmals) ließen sich letztendlich nur durch akribische Kontrolle der Ausgabe finden.

Um die Datenerfassung zu steuern und bereits bei der Akquisition eine Analyse des Materials vornehmen zu können, aber auch um Personen, die mit LeX4 oder einem in LeX4 spezifizierten linguistischen Modell nicht umgehen müssen, zur Datenerfassung einbeziehen zu können, wurde das im folgenden beschriebene Programmpaket BLAH ((*B*erliner *L*exikon*a*kquisitions*h*ilfe) entwickelt und in dem GUI-System *Tcl/Tk* [6] implementiert.

Mit der Bedienungsoberfläche BLAT (*B*erliner *L*exikon*a*kquisitions*t*ools) steht zudem ein Werkzeug zu Verfügung, welches auf graphisch-interaktive Art die Steuerung nahezu aller zum Lexikonaufbau notwendigen Systemkomponenten gestattet.

2 Installation

2.1 Erforderliche Software

Folgende Software ist erforderlich, um das Programmpaket zu installieren und zu nutzen:

- `Tcl` Version 7.4 Patch 1
- `Tk` Version 4.0 Patch 1
- `gcc` Gnu-C-Compiler
- Quintus Prolog 3.2
- Quintus Prolog Compiler (empfohlen)
- `X11` XWindow

2.2 Kompilieren

Ausgeliefert wird eine Datei `blah-`*n.m.o*`.tar.gz`, wobei *n.m.o* für die aktuelle Version steht. In dieser Datei sind alle hier beschriebenen Module enthalten, einschließlich der aktuellen Version des Lexikonformalismus $\mathcal{LEX}^4$.

Die Installation kann nun wie folgt vorgenommen werden (cf. `Blah/README`):

1. `gtar -zxf blah-`*n.m.o*`.tar.gz` (Entpacken)
2. `cd Blah/src` (ins Quellverzeichnis wechseln)
3. `Makefile.local` anpassen.

   Hier müssen die lokalen Verzeichnisnamen des `Tcl/Tk`-Pakets angegeben werden. Dabei handelt es sich um die folgenden Variablen:

`TCLLIB=/usr/local/tcl7.4/lib`	Pfad nach `libtcl.a`. Es muß sich dabei um die Library der Tcl-Version 7.4 Patch 1 handeln.
`TKLIB=/usr/local/tk4.0/lib`	Pfad nach `libtk.a`. Es muß sich dabei um die Library der Tk-Version 4.0 Patch 1 handeln.
`TCLINC=/usr/local/tcl7.4/include`	Pfad nach `tcl.h`
`TKINC=/usr/local/tk4.0/include`	Pfad nach `tk.h`
`OWLIB=/usr/openwin/lib`	Pfad zu den Libraries des `X11` Systems
`OWINC=/usr/openwin/include`	Pfad zu den *include*-Dateien des `X11` Systems

4. Wenn der Quintus Prolog Compiler verfügbar ist kann das System mit

   ```
   make
   ```

   kompiliert werden, ansonsten lautet der Aufruf

   ```
   make noqpc
   ```

5. `cd ..`

Nach dem Auspacken der `.tar.gz`-Datei und dem Kompilieren ergibt sich die folgende Verzeichnisstruktur:

```
./Blah
```

- `bin` Binärdateien und Shell-Scripts
- `lex4` Sämtliche *Lex4*-Daten und nutzerspezifizierbare Dateien
- `qof` QuintusProlog Object Files
- `src` Quellprogramme der Tools
- `tcl` Tcl/Tk Programme

Die für die Benutzer relevanten Daten sind in dem Verzeichnis `Blah/lex4` zu finden. Hier stehen (eventuell auf weitere Unterverzeichnisse verteilt) sämtliche *Lex4*-Klassendefinitionen, Dateien mit neueingegebenen bzw. reeditierten Einträgen sowie die Zusatzdateien, die für die Generierung der Abfragestrategie und Transformierung der *Lex4*-Ausgabe benötigt werden.

Alle Programme werden grundsätzlich aus dem Verzeichnis `./Blah` aufgerufen.

3 Programmteilpakte und ihre Anwendung

Die nachfolgende Skizze verdeutlicht, wie die verschiedenen Programme des Gesamtpakets miteinander kommunizieren:

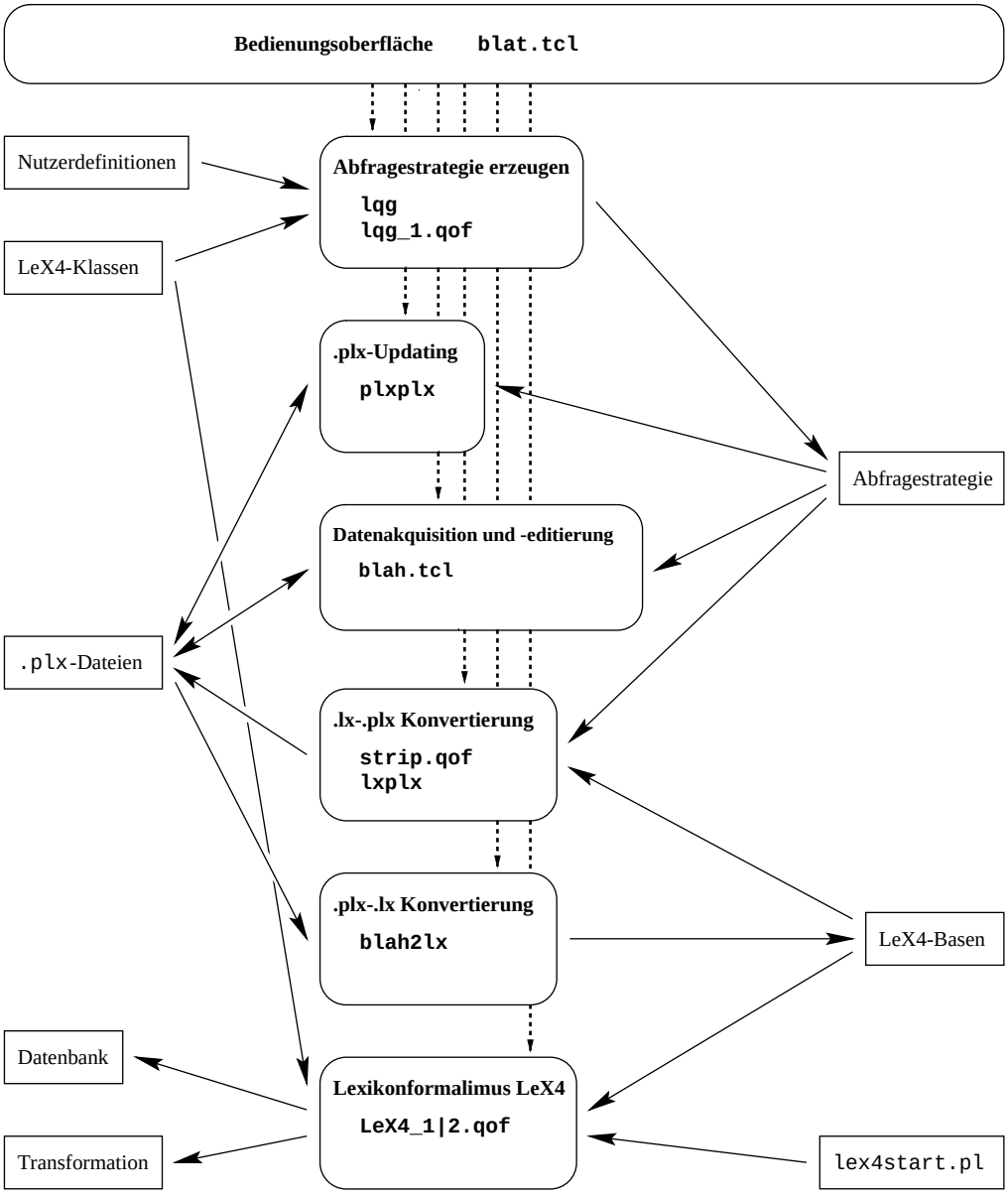

Abbildung 1. Datenfluß der Akquisitionstools

3.1 Bedienungsoberfläche BLAT

Diese Bedienungsoberfläche ist die Steuerzentrale, von der alle wichtigen Dateinamen verwaltet werden und die einzelnen Teilsysteme (mit korrekter Syntax) aufgerufen werden. Die Dateinamen sind in der Setup-Datei `Blah/blat.setup`[1] gespeichert.

Die Bedienungsoberfläche wird gestartet durch die Eingabe von:

```
tcl/blat.tcl
```

Optional kann auch ein Dateiname angegeben werden, wenn das Setup unter einem anderen Namen abgespeichert worden ist.

Nach dem Aufruf erhält man folgendes Fenster:

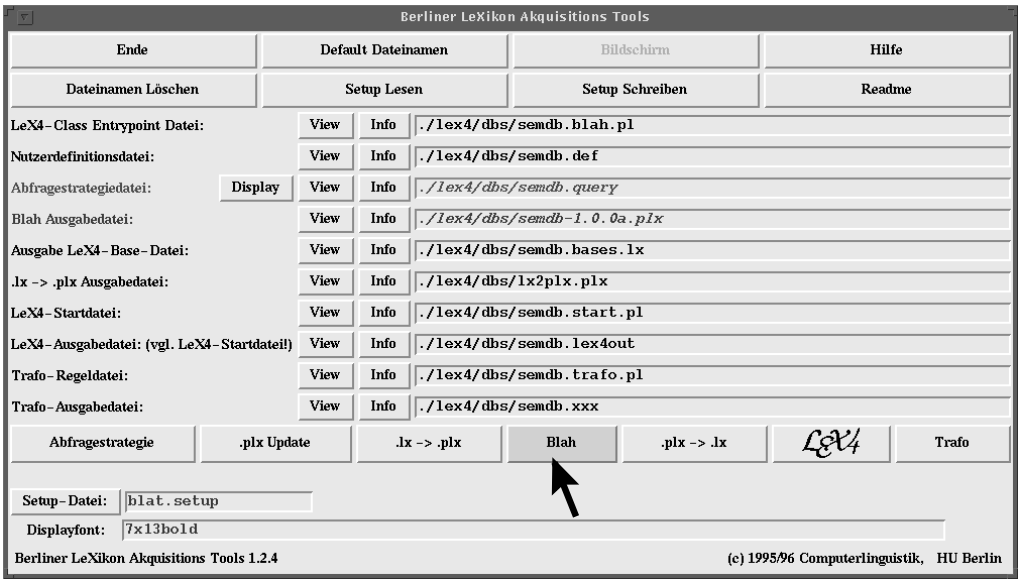

Abbildung 2. BLAT Oberfläche

Die Dateiennamen, die hier angegeben werden, werden von den Programmen, die durch Anklicken der Programmbuttons der unteren Programmzeile gestartet werden, gelesen bzw. geschrieben. Bewegt man die Maus über einen der Programmbuttons werden die Dateien, die von dem entsprechenden Programm gelesen werden, grün dargestellt, diejenigen, die geschrieben (überschrieben) werden, erscheinen in roter Farbe. Auf monochromen Bildschirmen werden zu lesende und zu schreibende Dateien in kursiver Schrift dargestellt.

[1] Für verschiedene Lexika kann ein anderer Name gewählt werden, der jederzeit von BLAT mit dem Button *Setup Lesen* eingelesen wird.

3.1.1 Beschreibung der von BLAT verwalteten Dateien

LEX4-Class Entrypoint Datei: In dieser Datei sind die LEX4-Klassen angegeben, zu denen Bases mittels BLAH zugeordnet werden können.

Nutzerdefinitionsdatei: Hier werden optionale Parameter angegeben, wie die Abfragestrategie aufgebaut wird (hierarchische Informationen, Ersetzungen von Feature-Namen in mnemotechnisch leichtere, natürlichsprachliche Ausdrücke).

Abfragestrategiedatei: Name der Abfragestrategiedatei, die BLAH liest, um die Abfragefenster zu erzeugen, in der die einzelnen Features abgefragt werden.

Blah-Ausgabedatei Alle **Neu**eingaben schreibt BLAH in diese Datei.

Ausgabe LEX4-Base-Datei: Die Einträge werden aus der BLAH-Ausgabedatei in diese Datei als reguläre LEX4-Bases geschrieben.

Achtung: Wenn diese Datei beim Lauf der LEX4-Maschine eingelesen werden soll, muß dies in der LEX4-Startdatei bzw. in einer der Dateien, die die LEX4-Startdatei liest, angegeben werden.

`lx`→`plx` *Ausgabedatei:* Bei der Rückkonvertierung von LEX4-Bases (`.lx`-Dateien) in das `.plx`-Format, (das Format, welches Blah lesen und schreiben kann) werden die Einträge in dieser Datei abgelegt.

LEX4-Startdatei: Beim Aufruf der LEX4-Maschine wird diese Datei geladen. Es muß spezifiziert werden, welche anderen Dateien von LEX4 geladen werden müssen (LEX4-Class-, LEX4-Base-Datei etc).

LEX4-Ausgabedatei: Die LEX4-Maschine schreibt ihre Ausgabe in diese Datei.

Achtung: Der Name dieser Datei wird in der LEX4-Startdatei (mit den Traforoutinen) angegeben. Eine Änderung in BLAT hat keine Wirkung auf die Ausgabe. Die Datei ist zur Vervollständigung hier aufgenommen worden.

Traforegeldatei: Die Ausgabe der LEX4-Maschine kann mit Regeln in dieser Datei in jede Form transformiert werden.

Trafoausgabedatei: Trafo schreibt seine Ausgabe in diese Datei.

Durch Anklicken des jeweils vorhanden Buttons *View* wird ein Fenster geöffnet, um die entsprechende Datei einzusehen.

3.1.2 Beschreibung der Funktions- und Programmbuttons in BLAT

3.1.2.1 Funktionsbuttons

Ende: Bricht das Programm sofort ab.

Default Dateinamen: Lädt die Default-Dateinamen.

Bildschirm: Einige Programme erzeugen Bildschirm-Loggings, die hier eingesehen werden können. Fehlermeldungen und Warnungen, die von den Programmen ausgegeben werden, welche durch die Programmbuttons aufgerufen wurden, lassen sich hier einsehen.

Hilfe: Online-Hilfe.

Dateinamen Löschen: Löscht die Dateinamen in den Fenstern.

Setup Lesen: Liest die Setup-Datei `blat.setup` erneut. Der Name der Setup-Datei kann jederzeit in dem dafür vorgesehenen Fenster unten links geändert werden. Ein Betätigen der Funktionsbuttons *Setup Lesen* bzw. *Setup Schreiben* verwendet den Dateinamen, der in diesem Fenster angezeigt ist.

Setup Schreiben: Die aktuellen Dateinamen werden in die Setup-Datei geschrieben.

Readme: Zeigt die mitgelieferte `README`-Datei an.

View: Öffnet ein Fenster, in dem die jeweilige Datei angezeigt wird. Der dabei verwendete X-Font kann in dem Fenster *Displayfont* unten rechts verändert werden. Der aktuelle Font wird bei Betätigung des *Setup Schreiben*-Buttons mit in die Setup-Datei geschrieben.

Info: Zeigt das Datum der letzten Dateiänderung sowie die Dateigröße an.

Display: Zeigt die Abfragestrategie graphisch an, mit der Möglichkeit, Informationen zu den $\mathcal{LILOG}$-Klassen abzufragen (cf. 3.1.2.4).

3.1.2.2 Progammbuttons

Der genaue Funktion der hier aufrufbaren Programme erfolgt in den Abschnitten 3.2 ff.

Abfragestrategie: Erzeugt die Abfragestrategie aus den $\mathcal{LILOG}$-Klassendefinitionen (cf. Abschnitt 3.2).

`plx`-*Update:* Paßt alte `.plx`-Dateien einer eventuell neueren Abfragestrategie an (cf. Abschnitt 3.4.3).

`lx→plx:` Konvertiert *LEX4*-Base-Dateien in (mit BLAH) editierbare `.plx`-Dateien (cf. Abschnitt 3.4.1).

Blah: Startet das eigentliche Akquisitionstool mit den angegeben Dateien (cf. Abschnitt 3.3).

`plx→lx:` Startet die `plx→lx` Konvertierung mit den angegeben Dateien (Umwandlung in *LEX4*-Bases) (cf. Abschnitt 3.4.2).

LEX4: Ruft den Lexikonformalismus auf (cf. Abschnitt 3.4.4).

Trafo: Ruft den Ausgabetransformator auf (cf. Abschnitt 3.4.5).

3.1.2.3 Tastatur-Shortcuts

Taste:	entspricht Button:
`ESC` oder `CTRL-C`	*Ende*
`F1` oder `CTRL-H`	*Hilfe*
`CTRL-X` `CTRL-F`	*Setup Lesen*
`CTRL-X` `CTRL-S`	*Setup Schreiben*

3.1.2.4 Graphische Ausgabe der Abfragestrategie

Die Abfragestrategie kann mit dem Button *Display* graphisch dargestellt werden. Durch Anklicken erscheint folgendes Fenster (vgl. Abbildung 3).

Diese Graphik kann mit den Scrollbars oder durch Drücken der mittleren Maustaste bei gleichzeitigem Bewegen der Maus verschoben werden.

Die *LEX4*-Klassen können angeklickt werden, um eine Liste aller Features dieser Klasse mit den Wertebereichen zu erhalten. Durch erneutes Anklicken der *LEX4*-Klasse kann dieses Fenster wieder ausgeschaltet werden. Der sichtbare Fensterausschnitt kann mit *PostScript-Datei erzeugen* in eine druckbare PostScript Datei geschrieben werden. Eine eventuelle Verkleinerung kann im Feld *Skalierung* angegeben werden.

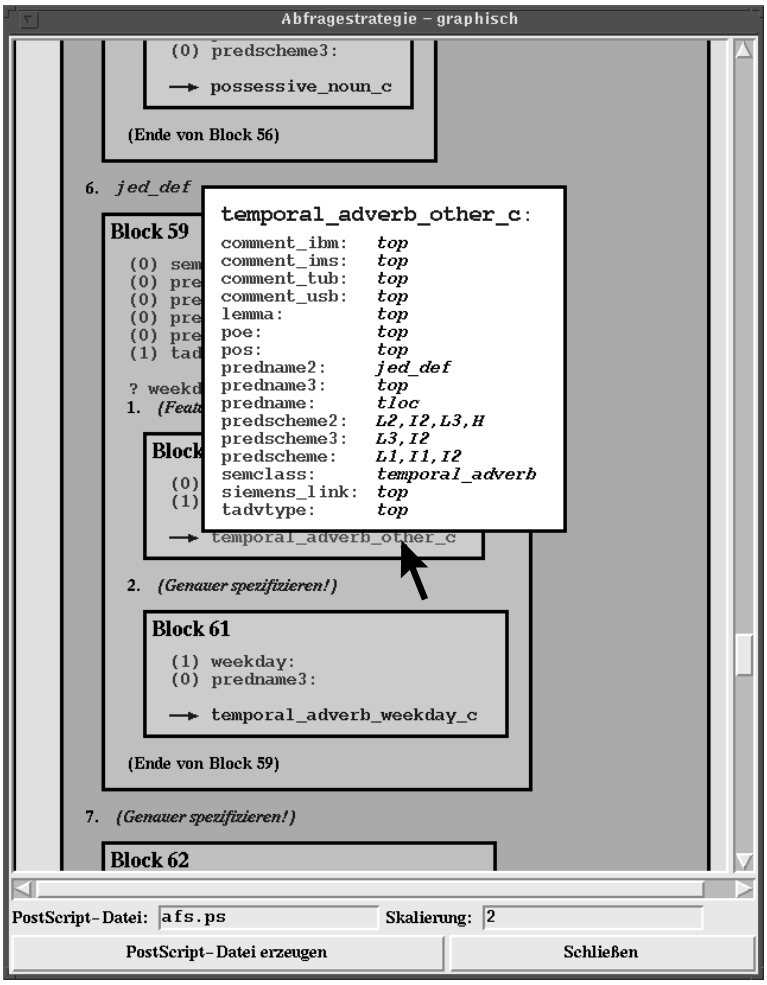

Abbildung 3. Abfragestrategie graphisch dargestellt

3.2 Abfragestrategiegenerator

Das im Abschnitt 3.3 beschriebene Akquisitionstool BLAH führt den Benutzer über mehrere Seiten, auf denen einzelne Features der einzugebenden Einträge spezifiziert werden müssen.

Dieses Programm erstellt die Abfragestrategie aus den Definitionen der LEV4-Classes und einer Liste aller LEV4-Classes, die direkt an Bases vererben können, sowie einer Datei, in dem der Nutzer Abfrageprioritäten angeben kann, [1].

Diese (Prolog)-Datei besteht zum einen aus `consult/1`-Aufrufen, um die LEV4-Class-Definitionen zu laden, zum anderen aus dem Prolog-Term `entry_points/1`,

dessen Argument eine Liste mit allen $\mathcal{LEX}$-Classes ist, denen Bases zugeordnet werden können:

```
% Hier werden die LEX-Class-Definitionen geladen:
:-['dbsem.class.lx'].
:-['dbsem.nom-class.lx'].
:-['dbsem.verb-class.lx'].
:-['dbsem.adjadv-class.lx'].

% Base-fähige LEX-Classes
entry_points([
   anaphoric_adjective_c,
   anaphoric_adverb_c,
   anaphoric_preposition_c,
   cardinal_c,
   ....
]).
```

Weiterhin lassen sich in der Nutzerdefinitionsdatei natürlichsprachliche Ersetzungen angeben, um Featurenamen und -werte durch Ausdrücke in einer nutzergerechten Form darstellen zu können. Defaultmäßig erscheint in der Abfrage von BLAT immer der Featurename selbst, sofern er nicht in der Nutzerdefinitionsdatei ersetzt wurde.

Das Format der Nutzerdefinitionsdatei läßt sich anhand eines Beispiels am besten verdeutlichen.

```
% Kommentarzeilen beginnen mit '%'
% Abfrageprioritäten
1. role_1:
2. predname:

% Natürlichsprachliche Ersetzungen
'_nein_' = 'Feature unnötig'
'top' = 'Genauer spezifizieren!'
'lemma:' = 'Lemma:'
'role_1:' = 'Role of first Argument:'
'role_2:' = 'Role of second Argument:'
'role_3:' = 'Role of third Argument:'
```

Zuerst wird nach dem Feature `sort_of_first_actant` gefragt, in der nächsten Klassifikation nach dem Feature `predname`. Auf diese Weise lassen sich die Abfragestrategien sehr flexibel aufbauen.

Die natürlichsprachlichen Ersetzungen unterstützen Benutzer, denen die aktuelle *LeX4-Class*-Struktur nicht geläufig ist, bei der Dateneingabe, da die speziellen Featurenamen in der Abfrage mit den Strings auf der rechten Seite ersetzt werden.

3.2.1 Aufruf von der Kommandozeile

`bin/makequery` *blah-entrypoints Abfragestrategie Nutzerdefinitionen*

Wobei:

blah-entrypoints	*LeX4*-Classes, denen Bases zugeordnet werden können
Abfragestrategie	Name der Datei, in die die Abfragestrategie geschrieben werden soll
Nutzerdefinitionen	Datei mit weiteren Nutzerdefinitionen.

3.3 Akquisitionstool BLAH

Dies ist der Kern der Akquisitionstools. BLAH dient in erster Linie zur Neueingabe von Lexikoneinträgen, kann aber auch zum nachträglichen Reeditieren vorhandener Einträge verwendet werden. Da BLAH nicht mit dem Standarddatenformat des Lexikonformalismus arbeitet, sondern aus Performanzgründen mit einem internen Format (`.plx`), welches außer den Featurewerten auch den Weg durch die Abfragestrategei mitabspeichert, können mit diversen Datenkonvertierungstools (cf. Abschnitte 3.4.1 (Konvertierung nach *LeX4*-Bases), 3.4.2 (Rekonvertierung von *LeX4*-Bases nach `.plx` und 3.4.3 (Anpassen alter `.plx`-Daten an eine neue Abfragestrategie)) alle *LeX4*-Einträge mit BLAH bearbeitet werden. Diese Konvertierungen erfolgen für den Nutzer jedoch völlig transparent. BLAH liest und erzeugt ausschließlich `.plx`-Dateien.

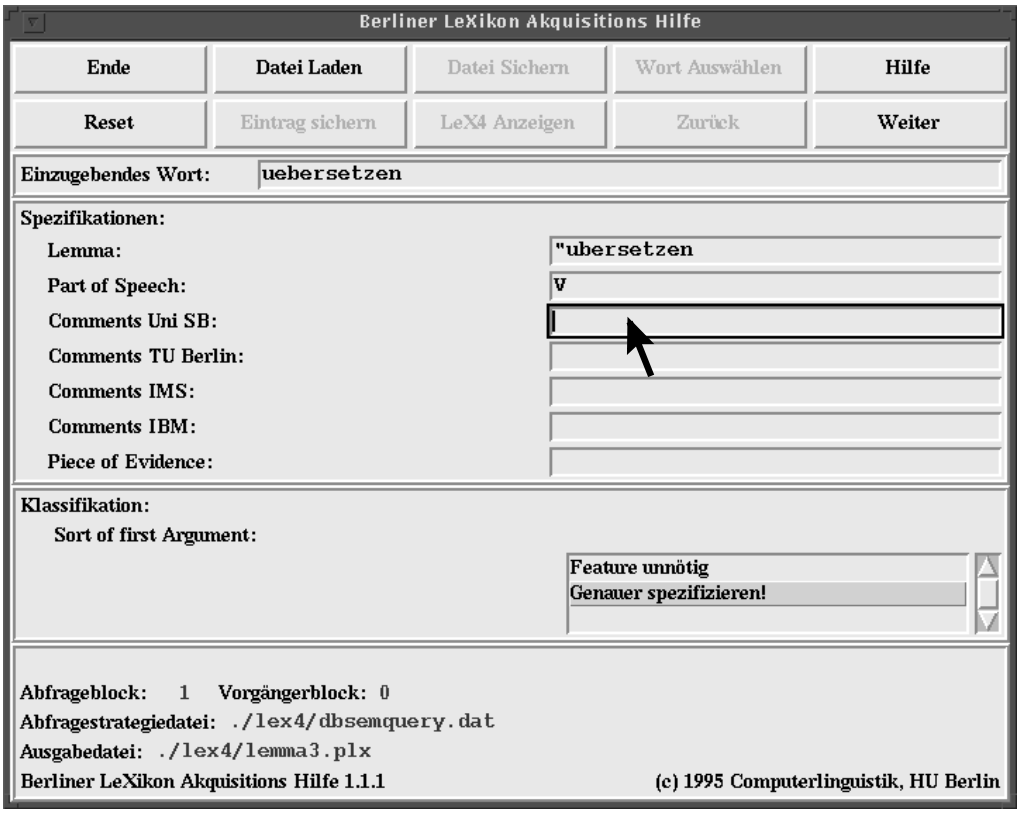

Abbildung 4. BLAH Oberfläche

Nach dem Aufruf aus BLAT über den Programmbutton *blah* erscheint ein weiteres Fenster. Da nicht alle Programmfunktionen zu jeder Zeit sinnvoll sind, sind die jeweiligen Buttons zu entsprechender Zeit inaktiv.

Die BLAH-Oberfläche ist in fünf Teilbereiche aufgeteilt, die folgende Funktion haben:

1. Funktionsbuttons: Alle Programmfunktionen werden mit den Funktionsbuttons aufgerufen. Teilweise sind jedoch zusätzlich Tastatur-Shortcuts implementiert (cf. Abschnitt 3.3.1.2). Die einzelnen Funktionen werden im folgenden Abschnitt erklärt.
2. Eintragname: In $\mathcal{LeX}4$ benötigt jeder Eintrag (*base*) einen Namen. An dieser Stelle kann der Name jederzeit geändert werden.
3. Spezifikationen: Aufgrund der Abfragestrategie werden soviel Features wie möglich durch BLAH von dem Benutzer erfragt. Falls ein Feature aufgrund der $\mathcal{LeX}4$-Class-Definition bereits einen (unveränderlichen) Wert hat, ist eine Änderung nicht möglich.

Beim Reeditieren von bereits bestehenden Einträgen werden die Abfragefenster mit dem momentanen Wert gefüllt, unterspezifizierte Features erscheinen mit einem leeren Fenster.

Es kann (besonders bei kurzen LEX4-Class-Strukturen) vorkommen, daß BLAH kein Feature zur Spezifizierung abfragt, sondern nur eine weitere Klassifikation braucht.

4. Klassifikationen: Hier trifft der Benutzer die Entscheidung, wie das folgende Abfragefenster aussehen soll. Aufgrund der LEX4-Class Strukturen ist an dieser Stelle immer nur eine Entscheidung sinnvoll.

   Ist eine Abfrage beendet worden, d.h. es gibt keinen weiteren Abfrageblock mehr, wird in diesem Teil der Name der LEX4-Class angezeigt, die der soeben eingegebene Eintrag aufgrund der Klassifikationsfolge erhält.

5. Statusinformationen: Der unterste Teilbereich zeigt einige Informationen über den Status von BLAH an:

   - Abfrageblock/Vorgängerblock: Labelnummer aus der Abfragestrategiedatei des Fragenblocks, der momentan abgefragt wird
   - aktuelle Abfragestrategiedatei
   - Ausgabedatei: alle **neuen** Einträge schreibt BLAH in diese Datei
   - Versionsnummer.

3.3.1 Beschreibung der Funktionsbuttons und Tastatur-Shortcuts in BLAH

3.3.1.1 Funktionsbuttons

Ende: Bricht das Programm sofort ab. Der aktuell bearbeitete Eintrag, der noch nicht gesichert war, ist verloren.

Datei Laden: Eine `.plx`-Datei laden, um sie nochmal zu (re)-editieren.

Datei Sichern: Reeditierte Daten wieder in die `.plx`-Datei schreiben (inaktiv, solange keine Datei geladen wurde).

Wort Auswählen: Ein Wort zum Reeditieren auswählen (inaktiv, solange keine Datei geladen wurde).

Hilfe: Online-Hilfe

Eintrag Sichern: Einen neu eingegeben Eintrag in das Ausgabefile schreiben, welches beim Aufruf angegeben wurde. Im Falle von reeditierten Einträgen wird der Eintrag als verändert markiert und mit dem Funktionsbutton `Datei sichern` in die vorher geladene `.plx`-Datei zurückgeschrieben (inaktiv bis alle Eingabemenüs abgearbeitet worden sind).

LeX4 Anzeigen: Einen neu eingegeben Eintrag in LeX4-Format ansehen (inaktiv bis alle Eingabemenüs abgearbeitet worden sind).

Reset: BLAH in den Anfangszustand setzen.

Zurück: Beim Eingeben der Features ein Fenster zurückgehen (inaktiv, solange man sich im ersten Abfragemenü befindet).

Weiter: Beim Eingeben der Features zum nächsten Fenster gelangen (inaktiv, wenn man sich im letzten Abfragemenü befindet).

3.3.1.2 Tastatur-Shortcuts

Taste:	entspricht Button:
`ESC`	*Ende*
`F1`	*Hilfe*
`RETURN`	*Weiter* bzw. *Eintrag Sichern* je nachdem, welcher Button aktiv ist
`CTRL-RETURN`	*Zurück*
`CTRL-X` `CTRL-F`	*Datei Laden*

3.3.2 Eintrag reeditieren

Um bereits eingegebene Einträge zu editieren, muß die entsprechende `.plx`-Datei in BLAH geladen werden. Nach Anklicken des *Datei Laden*-Buttons und Eingabe der zu reeditierenden `.plx`-Datei erscheint folgendes Auswahlfenster[2]:

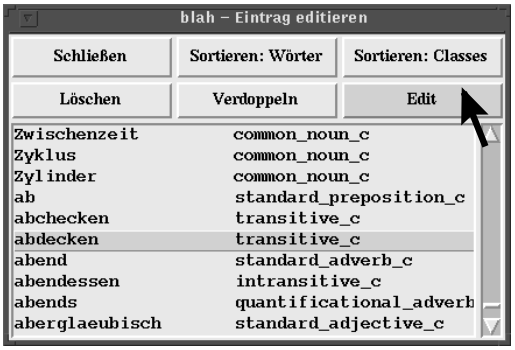

Abbildung 5. Auswahlmenü für zu editierende Einträge

[2]bei größeren `.plx`-Dateien (> 1000 Einträgen) kann das Laden unter Umständen noch 30 Sekunden dauern)

Der zu reeditierende Eintrag wird hier ausgewählt und kann durch Doppelklick bzw. Klick auf *Edit* nach BLAH geladen werden. Wenn man anschließend durch die Abfragemenüs geht, erscheinen an den jeweiligen Stellen die Werte der Features, die in dem Eintrag bereits eingegeben worden sind. Änderungen und neue Spezifizierungen sind so leicht vornehmbar. Klickt man im *Klassifikations*fenster auf eine andere Verzweigung als die ursprüngliche, kann ein Eintrag leicht umklassifiziert werden. Allerdings sind diejenigen Features unterspezifiziert, die nach der neuen Klassifikation noch abgefragt werden. Nachdem alle Abfragemenüs durchlaufen sind, kann man den reeditierten Eintrag mit *Eintrag Sichern* sichern und mit *Wort Auswählen* einen neuen Eintrag auswählen. Um die geänderten Einträge physisch in die ursprüngliche Datei zu schreiben, ist der *Datei Sichern*-Button anzuklicken.

3.3.2.1 Beschreibung der Funktionsbuttons und Tastatur-Shortcuts des Wortauswahlmenüs

Schließen: Das Auswahlfenster wird geschlossen.

Sortieren: Bases: Alphabetisches Sortieren der Einträge im Auswahlfenster

Sortieren: Classes: Alphabetisches Sortieren der Einträge im Auswahlfenster nach ihren LaTeX-Class-Namen

Edit: Ausgewählten Eintrag reeditieren

Verdoppeln: Kopie des ausgewählten Eintrags machen

Löschen: Ausgewählten Eintrag löschen

Taste:	entspricht Button:
ESC	*Ende*
RETURN oder *Doppelklick*	*Edit*

3.3.3 BLAH-Aufruf von der Kommandozeile

`tcl/blah.tcl` *Abfragestrategiedatei* `.plx`-*Ausgabedatei* `[-debug]`

Wobei:

Abfragestrategiedatei	Datei, aus der die Abfragestrategie gelesen wird
`.plx`-*Ausgabedatei*	Datei, in die die **neu**eingegebenen Einträge geschrieben werden
`-debug`	Optional: Während des Einlesens der Abfragestrategie wird der Abfragebaum nach `stdout` geschrieben.

3.4 Datenkonvertierungstools

Da BLAH nur mit einem Format arbeitet, wurden einige Konvertierungstools entwickelt und implementiert, die die Bearbeitung beliebiger LeX4-Einträge mit den Akquisitionstools ermöglichen.

3.4.1 `plx`→`lx` Konvertierung

Der Programmbutton `plx->lx` ruft die LeX4-Bases-Konvertierung auf, d.h. die Einträge im `.plx`-Format werden in reguläre LeX4-Bases konvertiert, um dann mit dem Lexikonformalismus weiterverarbeitet zu werden.

3.4.1.1 Aufruf von der Kommandozeile

`Plx2lx .plx-`*Datei* *LeX4*`-`*Base-Datei* `[-b|-s]`

Wobei:

`.plx-`*Datei*	BLAH-Ausgabedatei, wird hier eingelesen
LeX4`-`*Base-Datei*	Reguläre LeX4-Base-Datei '`-`' anstelle einer Datei liest/schreibt von/nach `stdin` bzw. `stdout`
`-b`	Optional: `.lx`-Datei wird nach Bases sortiert
`-s`	Optional: `.lx`-Datei wird nach Classes und Bases sortiert

3.4.2 `lx`→`plx` Rekonvertierung

LeX4-Bases können unter der Voraussetzung, daß sie zu den LeX4-Classes gehören, aus der die aktuelle Abfragestrategie generiert wurde, jederzeit wieder in das `.plx`-Format rekonvertiert werden, um mit `blah` reeditiert zu werden. Das mit dem Programmbutton `lx->plx` zu startende Programm bestimmt einen Weg durch die Abfragestrategie. Gelingt dies aufgrund von nicht vom System vorgenommenen Änderungen nicht, erscheint eine Fehlermeldung.

3.4.2.1 Aufruf von der Kommandozeile

`bin/striplx` *LeX4-Base-Datei* *Abfragestrategie* `.plx`*-Datei*

Wobei:

LeX4-Base-Datei	LeX4-Base-Datei, die gelesen wird
Abfragestrategie	Datei, aus der die Abfragestrategie gelesen wird
`.plx`*-Datei*	In das `.plx`-Format konvertierte Ausgabedatei

3.4.3 plx-Update

Durch Änderung der LEX4-Classes ist eine Neugenerierung der Abfragestrategie erforderlich. Da die Einträge der `.plx`-Dateien Informationen über den Weg durch die Abfragestrategie enthalten, ist ein Abgleich mit der neuen Abfragestrategie nötig. Dies leistet das Programm, welches der `plx`-*Update*-Button aufruft. Stellt das Programm fest, daß in der neuen Abfragestrategie im Eintrag vorhandene Features obsolet sind, wird eine Warnung ausgegeben, ebenso, falls neue Features hinzugekommen sind, die in den alten Einträgen nicht spezifiert sind.

Von der alten `.plx`-Datei wird ein Backup erstellt, und unter dem Namen *alter_name*`.plx.`*Monat*`-`*Tag*`.`*Stunde*`-`*Minute* im gleichen Verzeichnis gesichert.

3.4.3.1 Aufruf von der Kommandozeile

`bin/plxplx` *alt*`.plx` *Abfragestrategie* *neu*`.plx`

Wobei:

alt`.plx`	Aus dieser Datei wird gelesen.
Abfragestrategie	Datei, aus der die Abfragestrategie gelesen wird.
neu`.plx`	In diese Datei werden die der neuen Abfragestrategie angepaßten Einträge geschrieben.

3.4.4 Lexikonformalismus

Der Programmbutton LEX4 ruft den Lexikonformalismus auf. Zuerst liest die LEX4-Maschine die Datei, die in BLAT unter *LEX4-Startdatei* angegeben ist. In dieser Startdatei werden alle weiteren Dateien für die Lexikongenerierung geladen, d.h. LEX4-Class- und -Base-Dateien.

Die in BLAT unter *Ausgabe LEX4-Base-Datei* angegebene Datei wird gegenwärtig nicht automatisch in die LEX4-Maschine geladen, sondern muß mit dem Dateisystem von LEX4 geladen werden. Dazu sei auf [2] verwiesen.

3.4.4.1 Aufruf von der Kommandozeile

`bin/lexfor` *LEX4-Start-Datei*

Wobei:

LEX4-Start-Datei	Startdatei für die LEX4-Maschine, von der aus die LEX4-Classes und -Bases geladen werden.

3.4.5 $\mathcal{LaTeX}$-Ausgabe Transformation

Der Lexikonformalismus erzeugt aufgrund der Klassenstruktur eine Ausgabedatei, in der zu jeder Base eine Featurestruktur abgespeichert ist. Entspricht dieses Format nicht dem der Applikation des Lexikons, können diese Featurestrukturen mit $\mathcal{LaTeX}$-TRAFO[4] in jedes beliebige Format transformiert werden.

3.4.5.1 Aufruf von der Kommandozeile

`bin/trafo` *Traforegeln* *$\mathcal{LaTeX}$-Ausgabe* *Trafo-Ausgabe*

Wobei:

Traforegeln	Datei,aus der die Transformationsregeln gelesen werden
$\mathcal{LaTeX}$-Ausgabe	Ausgabe des Lexikonformalismus'
Trafo-Ausgabe	Konvertierte Ausgabe des $\mathcal{LaTeX}$-TRAFOs.

A Dateiformate

Die Daten, die von den oben beschriebenen Programmen gelesen und geschrieben werden, haben fest definierte Formate, die im folgenden erläutert werden. Die meisten Formate sind jedoch intern, d.h. sie werden nur von Programmen gelesen bzw. geschrieben. Der Nutzer sollte also im Normalfall keine Notwendigkeit sehen, sich diese Dateien anzusehen.

A.1 *.query - Abfragestrategie

Das Format der Abfragestrategie, die von `lqg` erzeugt und von `blah.tcl`, `lxplx` sowie `plxplx` gelesen wird, sieht folgendermaßen aus:

```
% Kommentarzeilen beginnen mit einem '%'
mark 1 {    % Jeder Block beginnt mit 'mark'
            % und einer Nummer, sowie '{'
   spezifikation {  % Kann auch leer sein
       (1) 'd_object:' "direktes Objekt:"
       (4) 'numerus:' "Numerus:" {
                        'sg' "Singular" / 'pl' "Plural"
           }

   }
   klassifikation {  % Darf nie leer sein
```

```
        (3) 'syn:particletype:' "-" {
            'top' "-" goto 3 /
            '_nein_' "-" goto 4
        }
    }
}    % Der gesamte Block muss auch mit '}' enden

...

mark 23 {
    spezifikation {
    }
    klassifikation {
        (0) return trans_verb_c %  Der Eintrag ist eingegeben, und bekommt
                                %  die LaTeX-Klasse 'trans_verb_c'
    }
}

endinput %  Nach jedem Block kann mit 'endinput'
         %  das Lesen abgebrochen werden.
%% EOF %%
```

Sowohl der `spezifikation`s-Block als auch der `klassifikation`s-Block sind folgendermaßen ausgebaut:

`(`*n*`)`	Fragetyp (s.u.)
`'String'`	Featurename (in einfachen Anführungszeichen)
`"`*String*`"`	Natürlichsprachliche Umschreibung (wird in Nutzerdefinitionsdatei angegeben), um unverständliche Featurenamen zu erläutern (in doppelten Anführungszeichen)

Beim Fragetyp `(4)` folgt nach `"String"` noch die Liste der Auswahlmöglichkeiten, die durch '`/`' voneinander getrennt werden:

`'String'`	Feature
`"`*String*`"`	Natürlichsprachliche Umschreibung

Bei `klassifikation` folgen noch die Verweise auf andere Abfrageblöcke, welche wiederum mit '`/`' voneinander getrennt sind:

`'String'`	Feature
`"`*String*`"`	Natürlichsprachliche Umschreibung
`goto` *nn*	Folgender Abfrageblock

Wenn keine natürlichsprachliche Umschreibung vorliegt, erscheint sie als `"-"`.

Die Fragen im `spezifikation`s-Block kann aus folgenden Typen bestehen:

Typ	Erläuterung
`(0)`	`subject:nonlex_case` (Feature über die Klasse schon spezifiziert)
`(1)`	Feature hat den Wert '`top`' in der Klasse
`(3)`	Beliebige Auswahl aus einer Liste (Noch nicht implementiert)
`(4)`	Genau eine Möglichkeit aus einer Liste

Der `klassifikation`s-Block kann aus folgenden Typen bestehen:

Typ	Erläuterung
`(0)`	Keine weitere Klassifikation, LEX4-Klasse wird übergeben
`(3)`	Gehe zur angeklickten Marke

A.2 *.plx - Internes base-Format

Um bereits vorliegende Einträge editieren zu können, müssen diese nach BLAH geladen werden. Da das originäre LEX4-Format in `Tcl` schwer interpretierbar ist, werden die *base*s in ein internes Format konvertiert. Die Ausgabe von BLAH ist in jedem Fall eine `.plx`-Datei. Das Format dieser Dateien ist abhängig von der Abfragestrategie, mit deren Hilfe die Daten eingegeben bzw. konvertiert worden sind. Ein Eintrag befindet sich in einer einzelnen Zeile.

Inhalt	*Format*
LEX4_Keywort	`String`
Marke_der_Abfragstrategie	`integer`
Anzahl_Spezifikationen	`integer`
Anzahl_Spezifikationen * { Feature Value }	`String%String%`
Nächste_Marke	`integer`
oder 0 LEX4_Klasse	`integer String`
Abfragestrategiedateiname	`String`
Prüfsumme	`long_integer`

A.3 Andere Formate

`*.def` Die Nutzerspezifikationsdatei ist in Abschnitt 3.2 beschrieben.

`*.lx` Alle Dateien mit diesem Suffix sind originäre LEX4-Dateien. Dabei kann es sich um *class-* oder um *base*-Dateien handeln. Eine genauere Beschreibung findet sich im LEX4-Handbuch [2].

`*.pl` Prolog-Dateien. Die LEX4-Startdatei und die Traforegeldatei sind ebenfalls als Prolog-Datei gekennzeichnet. Sie enthalten jedoch auch LEX4-spezifische Prädikate. Genaueres entnehme man bitte [2] und [4].

B Systemdateien

Hier findet sich eine Übersicht über alle ausgelieferten Dateien:

`./Blah/bin/blah2lex`	`plx-lx`-Konverter
`afs2tk`	Graphikprogramm für BLAT
`blahwish`	Erweiterter `Tcl/Tk`-Interpreter für BLAH
`lexfor`	Shellscript für LeX4-Aufruf
`lqg`	Abfragestrategiegenerator
`lxplx`	`lx-plx`-Konverter
`makequery`	Shellscript für `lqg`-Aufruf
`plxplx`	`plx`-Updater
`sec2date`	Hilfsprogramm für BLAT
`showcl`	Hilfsprogramm für BLAT
`striplx`	Präprozessor für `lx-plx`-Konverter
`trafo`	Shellscript für TRAFO-Aufruf
`./Blah/bin/LeX4_1.qof`	LeX4
`LeX4_2.qof`	LeX4-Zusatzprädikate
`lqg_1.qof`	Präprozessor für Abfragestrategiegenerator
`settings.pl`	LeX4-Systemsteuerung
`strip.qof`	Präprozessor für `lx-plx`-Konverter
`trafo.qof`	TRAFO
`./Blah/src/Makefile`	
`Makefile.local`	Lokale Anpassungen zum `Makefile`
`afs2tk.c`	Graphikprogramm für BLAT
`blah.c`	`Tcl`-Erweiterungen für BLAH
`blah.h`	c-Headerdatei für `Tcl`-Erweiterungen
`blah2lex.c`	`plx-lx` Konverter
`blahread.c`	`.plx`-Leseroutinen
`blahread.h`	c-Headerdatei für `.plx`-Leseroutinen
`blahtool.c`	Hilfsroutinen für BLAH
`blahwish.c`	Mainfile für `Tcl`-Erweiterungen
`lqg.h`	c-Headerdatei für Abfragestrategie-Generator
`lqg_file.c`	Abfragestrategie-Generator
`lqg_main.c`	Abfragestrategie-Generator
`lqg_matr.c`	Abfragestrategie-Generator
`lqg_tree.c`	Abfragestrategie-Generator
`lxplx.c`	`lx-plx` Konvertierer
`matrix.c`	Abfragestrategie-Generator
`matrix.h`	c-Headerdatei für Abfragestrategie-Generator
`plxfeature.h`	c-Headerdatei für Hilfsroutinen für BLAH
`plxplx.c`	`plx`-Updater
`plxrekurs.c`	Hilfsroutine für Konverterprogramme

`sec2date.c`	Hilfsprogramm für BLAT
`showcl.c`	Hilfsprogramm für BLAT
`./Blah/tcl/blah.icon`	Bitmap für `X11` Icon
`blah.tcl`	BLAH
`blat.tcl`	BLAT
`./Blah/Blah`	Shellscript für BLAH-Aufruf
`Plx2lx`	Shellscript für `plx-lx`-Konverter Aufruf
`README`	
`blat.dbssetup`	Beispiel Setup-Datei für BLAT

C Literatur

[1] Buhr, Peter: LQG - Generierung lexikalischer Abfragestrategien. Humboldt-Unversität zu Berlin (forthcoming 1996).

[2] Gebhardi, Gunter; Heinecke, Johannes: Lexikonformalismus $\mathcal{LEX}4$ — Sprachbeschreibung. Verbmobil Technisches Dokument 19. Humboldt-Unversität zu Berlin 1995.

[3] Gebhardi, Gunter; Heinecke, Johannes: Substantivflexion in $\mathcal{LEX}4$. Ein Applikationsbericht. Verbmobil Memo 62 Humboldt-Unversität zu Berlin 1995.

[4] Gebhardi, Gunter: $\mathcal{LEX}4$-TRAFO Version 2. Verbmobil Technisches Dokument. Humboldt-Unversität zu Berlin (forthcoming) 1996.

[5] Heinecke, Johannes; Gebhardi, Gunter: Konjugation der Verben im Lexikonformalismus. Verbmobil Memo 63. Humboldt-Unversität zu Berlin 1995.

[6] Ousterhout, John K.: Tcl und Tk. Addison-Wesley: Bonn u.a. 1995.