Germanistisches Institut der Eötvös-Loránd-Universität Budapest

Einführung in die Computerlinguistik

Zusammengestellt von
Pál Uzonyi

Budapest 1994

Pál Uzonyi

Einführung in die Computerlinguistik

Eötvös-Loránd-Universität

Budapest, 1994

Lektoren: Ferenc Kiefer
László Hunyadi
Renate Heim

ISSN 0138-9055

Felelős kiadó: Dr. Manherz Károly igazgató,
ELTE Germanisztikai Intézet, 1146 Budapest, Ajtósi Dürer sor 19-21.

Nyomtatta és kötötte a Dabas-Jegyzet Kft. 300 példányban
Felelős vezető: Marosi György ügyvezető igazgató
Munkaszám: 95-0026

Inhalt

1.
Hardware und Software: eine Einleitung für Laien

Beim Schreiben dieser Einleitung ging ich davon aus, daß die künftigen Leser des Buches Profis in Linguistik, jedoch Laien in Computertechnik sind. Deswegen möchte ich mich diesmal mit dem Phänomen *Sprache* nicht auseinandersetzen, dafür aber so viele computertechnische Grundbegriffe unter die Lupe nehmen, wie es auf diesen wenigen Seiten nur möglich ist.

Sehen wir uns zunächst das Wort "Computer" an: es ist ein englisches deverbales Substantiv, vom Verb *compute* abgeleitet; *compute* heißt rechnen, daher begegnet einem oft in der deutschen Terminologie auch die wortwörtliche Übersetzung *Rechner*. Aber sogar Laien wissen, daß ein Computer bei weitem nicht nur rechnen kann wie etwa ein kleiner Taschenrechner: ein durchschnittlicher Rechner kann gegenwärtig schreiben, zeichnen, musizieren, unterrichten usw. Die etwas besseren Computer steuern Raketen, entwerfen neue Computer, verstehen menschliche Sprachen - und das ist keine Fiktion mehr. Dabei funktionieren die modernsten Rechner nach wie vor aufgrund der elektrischen Binarität und sogar die kompliziertesten Prozesse im Innern des Computers setzen sich aus gleichartigen atomaren Schritten zusammen. Es wird nämlich immer nur eines geprüft: ist an der gegebenen Speicherstelle Strom vorhanden oder nicht? Die binäre Opposition "Strom-ja: Strom-nein" pflegt man mit 1 und 0 wiederzugeben. Diese atomare Information nennt man ein Bit. Komplexere Informationen ergeben sich aus Kombinationen von Einsen und Nullen. Die Grundeinheit der Bitkombinationen - wie etwa ein Molekül - besteht bei den geläufigen Computern aus 8 Bit und heißt Byte. Das Byte 01000001 repräsentiert beispielsweise den Buchstaben A.

Die ersten Rechner funktionierten noch im Dezimalsystem. Man experimentierte mit elektromechanischen Geräten bereits in den 30er Jahren, unter anderem in Ungarn (Kozma László, TU), in Deutschland (Konrad Zuse) und in den USA (1944, Mark I., Harvard Universität). Der erste richtig funktionierende elektronische Rechner wurde auch in den USA hergestellt, im Jahre 1946 (Elektronic Numerical Integrator And Calculator). Mit seinen 30 Tonnen und 18000 Röhren, die sich nur in einer mehr als 30 m langen Anlage beherbergen ließen, leistete der

"Urcomputer" weniger als die Armbanduhrrechner, die man heute für 300 Ft auf dem Flohmarkt kaufen kann (ENIAC kostete 10 Mill. $). Zur wesentlichen Beschleunigung der Weiterentwicklung hat vor allem der amerikanische Wissenschaftler ungarischer Herkunft John von Neumann beigetragen.

Nun, sehen wir uns einen Computer ein bißchen näher an. Das Gerät an sich, ohne Programme, wird Hardware genannt. Das Wort bedeutete im Englischen ursprünglich Werkzeug, Gartengerät wie Hammer, Spaten usw. Hardware nennt man heute aber nicht nur den Computer im engeren Sinne, sondern auch die ganze Maschinerie, die man an ihn koppeln kann, z.B. Drucker, Steuerknüppel usw. Der Hardware wird die Software gegenübergestellt, die "weiche Ware", d.h. alles, was man in einen Computer eingeben kann, wie z.B. die Programme, Datenmengen, überhaupt alle Arten von digitalisierten Informationen.

Die wichtigsten Bauteile der modernen Computer sind integrierte Schaltkreise (IC), die mit größtenteils automatisierten Präzisionsverfahren in einer miniaturisierten Form hergestellt werden. Wegen der kleinen Abmessungen werden sie Chips (Brocken) genannt. Sie leisten auf einigen Quadratzentimetern mehr als die ersten Geräte auf vielen Quadratmetern. Ein Computer hat mehrere Chips, die verschiedene Aufgaben haben können. Aus den Chips wird ein funktionsfähiges System zusammengestellt, das auf einer Platine befestigt wird. Diese Platine ist das Gehirn des Computers. Wenn sie also durch eine andere ausgetauscht wird, entsteht ein anderer Computer (mit einem Platinenwechsel kann man z.B. aus einem 286-er Rechner einen 386-er machen).

Der vielleicht wichtigste Chip auf der Platine ist die sogenannte Zentraleinheit oder CPU (Central Processing Unit), oft einfach Prozessor genannt. Die CPU kann mit den Daten operieren, die in den Speicherchips sind. Die Geschwindigkeit der Operationen hängt vor allem von der Taktfrequenz des Prozessors ab. Die neueren Zentraleinheiten der PCs sind mit mehr als 30 MHz getaktet (33, 40, 50 usw.).

Viele der Chips eines Computers sind Speicherbausteine, von denen einige bereits bei der Herstellung mit Software ausgefüllt werden. So

ist z.B. die Form der Buchstaben und anderer Zeichen, die auf Tastendruck auf dem Bildschirm erscheinen, auch in einem IC aufgeschmolzen. Derartige fest programmierte Chips sind der Festwertspeicher, der ROM-Speicher (Read Only Memory - "nur lesbar").

Eine andere Art der Speicherbausteine sind die RAM-Chips, in die z.B. Programme vom Nutzer (user) frei eingeschrieben werden können. Der RAM-Speicher ist (oder: die RAMs sind) Random Access Memory, d.h. Zufallszugriffsspeicher oder Randomspeicher.

Die Speicherkapazität wird in Bytes, KiloBytes oder MegaBytes gemessen. Ein K ist 1024 Byte, ein M ist tausend K ($1024 = 2^{10}$). Je größer der RAM-Speicher eines Rechners ist, desto kompliziertere, intelligentere Programme können darin ablaufen. Die RAM-Kapazität der kleinsten Heimcomputer (wie z.B. Commodore 16) beträgt 16 K, die kleinsten Personalcomputer (wie z.B. ein IBM-kompatibler AT) haben nicht weniger als ein Megabyte.

Beim Einschalten des Computers sind die RAM-Chips völlig leer. Lediglich die in den ROMs permanent enthaltenen Betriebssysteme werden aktiviert. Da beim Abschalten des Computers alle Daten und Programme in den RAMs gelöscht werden, müssen diese bei Bedarf woanders gespeichert werden.

Programme oder andere Dateien (Files) werden auf magnetischer Basis aufgezeichnet. Mit Hilfe der sogenannten externen Speicher werden diese auf Magnetbändern, Floppy-Disketten, Festplatten, Magnettrommeln usw. (sog. Speichermedien oder Datenträgern) fixiert, von denen sie sich dann jederzeit wieder in die RAMs laden lassen.

Den Abmessungen nach unterscheidet man zwischen Notebook ("Notizbuch" - wie ein großes Buch), Laptop (wie ein Aktenkoffer), Desktop Personal Computer (PC, den man auf einen Schreibtisch stellen kann) oder Tower (der etwas höher ist und deshalb öfters auf den Boden gestellt wird). Es gibt aber auch heute noch ganz große Maschinen etwa wie Kleiderschränke; das sind die Mainframes, die Supercomputer, die in Rechnerzentren stehen und deren Kapazität in GigaBytes (1000 M) gemessen wird.

Der Typ, den man am häufigsten an verschiedenen Orten vorfindet, ist der PC. Woraus besteht die Grundausstattung eines Desktop-Personal-Computers? Die Chips - so auch die CPU - befinden sich in einem Gehäuse. Darin sind meistens noch zwei externe Speicher untergebracht: eine Festplatte (Hard-Disc oder Winchester-Disk) und ein Diskettenlaufwerk (Floppy-Disk-Drive). Es gibt Festplatten mit verschiedener Kapazität: die meisten PC-s haben heutzutage eine 40 oder 80 MB-Festplatte, aber es kann auch mehr als 200 sein. Es ist eine andere Größenordnung als die Kapazität einer Floppy-Diskette, die gemeinhin maximal 1,44 MB beträgt (das ist typisch für die kleinere, 3,5 Zoll-Diskette), die 5,25 Zoll-Disketten werden meistens für 1,2 M oder 360 K formatiert. Eine Festplatte ist nicht nur geräumiger als eine Diskette, auch ihre Operationsgeschwindigkeit ist wesentlich höher.

Außerhalb des Gehäuses sieht man meistens noch eine Tastatur und einen Monitor (häufiger einfach nur "Bildschirm" genannt). Die Tastatur kann auch fest mit dem Gehäuse zusammengebaut sein (das ist z.B. bei kleinen Heimcomputern wie Commodore 64 oder bei Laptops und Notebooks der Fall). Sie ist einer elektrischen Schreibmaschine ähnlich. Die Tasten-Zeichen-Entsprechungen sind durch die ROMs festgelegt, aber mit entsprechender Software können sie leichthin geändert werden (damit z.B. Umlautbuchstaben einfacher zu schreiben sind).

Der Monitor kann monochromatisch (schwarz + weiß/grün...) oder farbig sein. Je nachdem, wie hoch die Auflösung des Bildschirms ist, unterscheidet man zwischen CGA (niedrigste Auflösung), EGA (etwas feiner, "angereichert") und VGA (Video-Graphic-Adapter); bei dem letzteren gibt es auch mehrere Stufen, von etwa 640 x 400 bis 1280 x 1024 oder mehr Bildpunkte. Bei den einzelnen Typen kann auch die Zahl der Farben unterschiedlich sein: von 16 bis mehrere Millionen. VGA ist nicht unbedingt farbig (vor allem bei Laptops/Notebooks): hierbei entsprechen den Farben verschiedene Graustufen (wie bei einem Schwarz-Weiß-Foto).

An den Computer kann man bei Bedarf einen Drucker anschließen. Die zwei Arten, die einem am häufigsten begegnen, sind der Matrix-Drucker und der Laser-Drucker. Matrix-Drucker sind billiger, aber dafür lang-

samer und schreiben bei weitem nicht so schön wie die Laser-Printer.
Externe Geräte, die man an Schnittstellen (Interfaces) des Rechners koppelt, wie z.B. die Drucker, sind Peripherie-Geräte. Als Peripherie gilt auch die sog. Maus, mit deren Hilfe man viele Programme leichter bedienen kann (vor allem bei der Auswahl von Menüpunkten).
Computer können als unabhängige Einzelgeräte ("stand-alone-units"), genutzt werden oder in einem Verbundsystem, in einem Netz(werk) eingesetzt werden, wo mehrere Computer als Terminale (mit oder ohne eigene Festplatten) an einen zentralen Speicher, an einen sog. Server angeschlossen sind. Die Bedienung einer Vernetzung ist ein wenig komplizierter als die eines alleinstehenden Rechners (z.B. logging-in/out, d.h. sich an/abmelden).
Um mit einem IBM-kompatiblen Personalcomputer umgehen zu können, muß man einige Wörter und Symbole erlernen: dies sind die Systemkommandos. Mit System ist das sog. Disk-Operationssystem (Plattenbetriebssystem), das DOS gemeint. Es ist auch eine Software, die beim Einschalten gestartet wird. Es gibt verschiedene Versionen mit kleinen Unterschieden. Das DOS ist nicht in den ROMs, sondern auf der HD oder einer FD. In den ROMs ist meistens nur ein BIOS, d.h. Basic Input Output System.
Wenn man das Gerät einschaltet, wird der Speicher überprüft, die Peripherien, danach meldet sich das BIOS und dann das DOS mit einem Prompt (Systemmeldung) und mit blinkendem Cursor (Positionsanzeiger). Die Laufwerke werden mit Buchstaben samt Doppelpunkt identifiziert: a: ist immer ein Diskettenlaufwerk, c: die Festplatte; wenn die Festplatte geteilt wird und/oder in einem Netz mehrere Festplatten zu erreichen sind, muß man weitere Buchstaben gebrauchen: e:, f:, y: usw.
Ein zweites Diskettenlaufwerk wird meistens b: genannt. Von einem Laufwerk aufs andere kann man umschalten, indem man den entsprechenden Buchstaben, Doppelpunkt, dann Enter drückt.
Die Informationen werden in der Form von Dateien gespeichert. Eine Datei hat immer einen Namen und meistens auch eine Erweiterung (extension). Grundsätzlich gibt es drei Erweiterungen, die darauf

hinweisen, daß man eine Datei direkt starten kann, also daß sie ein ausführbares (executable) Programm ist: *.EXE, *.COM und *.BAT. Zwischen Dateinamen und Erweiterung steht immer ein Punkt. Man kann ein Programm starten, indem man den Programmnamen eintippt, ohne Erweiterung, und dann Enter drückt. Es können auf diese Weise nur solche Programme gestartet werden, die auf dem Speichermedium im aktuellen Laufwerk vorhanden sind. Sonst bekommt man die Fehlermeldung des DOS: "Bad command or file name". Wie kann man erfahren, was sich auf einer Festplatte oder Diskette befindet? Mit dem Kommando DIR kann man so ein Inhaltsverzeichnis abrufen. Wenn nur DIR eingetippt wird, erscheint die Liste der Dateien des aktuellen Laufwerks (das aktuelle Laufwerk wird jeweils im Prompt angezeigt). Falls wir uns ein anderes Laufwerk ansehen wollen (z.B. das Diskettenlaufwerk a:), so müssen wir nach dem Kommando DIR auch den Laufwerknamen eingeben: DIR A:. Um ein unerwünschtes Weiterrollen bei zuviel Dateien zu verhindern, tippt man /P, so rollen die Zeilen erst nach einem Tastendruck weiter.

Directory ist also das Verzeichnis von Dateinamen. In diesem Verzeichnis können aber Namen auftauchen, die keine Dateinamen sind. Sie haben den Vermerk "DIR" bei sich, der bedeutet, daß es eine "Subdirectory", ein Verzeichnis im anderen Verzeichnis ist. Um in dieses Unterverzeichnis zu kommen, muß man die Directory wechseln, inden man CD (Change Directory) und den Namen der Subdirectory eingibt. Eine Subdirectory kann weitere Subdirectories enthalten usw., wie eine Matrjoschka-Puppe. Der volle Name einer Datei enthält demgemäß außer dem eigentlichen Namen auch die Reihe der Verzeichnisse, d.h. den Weg (Pfad, path), auf dem sie zu erreichen ist (z.B. c:\oktato\nemet\rektion.exe).

Aus einer Subdirectory kommt man in eine andere, indem man cd, Backslash (verkehrter Schrägstrich) und den anderen Directorynamen eingibt. Wenn man keinen Directorynamen angibt, kommt man wieder in das Hauptinhaltsverzeichnis (root directory) zurück.

Versuchen wir jetzt, eine Datei von der Festplatte auf eine Diskette zu kopieren. Wenn die Diskette noch nie gebraucht wurde, muß man sie

manchmal selbst formatieren (man kann auch solche Disketten neu formatieren, die Dateien enthalten, aber dann werden diese gelöscht). Das Kommando heißt: FORMAT und Laufwerksymbol - in diesem Fall ist es a: (man kann auch die Festplatte formatieren, aber das ist komplizierter). Ein Laufwerk mit 1,2 MByte formatiert die Diskette automatisch für 1,2 MB. Wenn man aber keine HD, sondern eine schwächere, eine DD-Diskette eingelegt hat, bekommt man bald eine Fehlermeldung. DD-Disketten formatiert man im 1,2 MB Laufwerk für 360 K, indem man dem Kommando "format a:" noch "/4" zufügt. Ein 360 KB Laufwerk kann Disketten nur für 360 K formatieren.

Wie kopiert man nun z.B. das Programm REKTION.EXE auf die soeben formatierte Diskette? Wir können zunächst ein neues Verzeichnis auf der Diskette eröffnen (das ist nicht unbedingt nötig), in das wir später z.B. weitere deutsche Lernprogramme kopieren wollen. Eine Directory (Verzeichnis) macht man mit MD (Make Directory) + Directoryname. Schalten wir auf a: um, dann schreiben wir "md nemet". Nun können wir mit dem Kopieren anfangen. Das Kommando heißt COPY. Das schreiben wir ein, dann - nach einer Leerstelle (space) - den Namen der zu kopierenden Datei mit dem zu ihr führenden Pfad, dann wieder space und den Weg zur Stelle, an die das Programm kopiert werden soll (copy c:\oktat\nemet\rektion.exe a:\nemet). Wenn man in demselben Verzeichnis ist, wo sich das zu kopierende File befindet, braucht man den Pfad nicht anzugeben (z.B. copy rektion.exe a:\nemet).

Wenn wir eine Datei nicht mehr brauchen, können wir sie löschen, u.z. mit dem Kommando DEL + Filename (z.B. del a:\nemet\rektion.exe). Ein leeres Verzeichnis läßt sich mit RD löschen (z.B. rd a:\nemet).

Mit dem Kommando TYPE + Filename kann man den ganzen Inhalt von Textdateien auf dem Bildschirm erscheinen lassen, mit PRINT + Filename werden dieselben Informationen zum Drucker geschickt und ausgedruckt. Mit CLS (Clear Screen, "Leere den Bildschirm") wird alles auf dem Bildschirm gelöscht.

Für diejenigen, die das DOS nicht nutzerfreundlich genug finden, hat man verschiedene Programme entwickelt, die mehr und schneller leisten als DOS, und dabei einfacher zu bedienen sind. Eines der populärsten

Programme dieser Art ist der Norton Commander. Da werden die Informationen mit Fenstern und Menüs verwaltet. Eine noch nutzerfreundlichere Oberfläche bietet das System Windows, das fast 100%-ig mit einer einzigen Maus bedient werden kann.

Im Zeichensatz des Computers sind 254 Zeichen: Ziffern, Buchstaben, Satzzeichen, mathematische Symbole, graphische Zeichen usw. Davon sind knapp 50 mit einem Tastendruck abrufbar, ebensoviel mit Shift.

Die direkt abrufbaren Buchstaben sind meistens nur die des englischen Alphabets, aber es gibt auch spezielle, z.B. deutsche Tastaturen. Jedes Zeichen hat eine international standardisierte Kodenummer, den sogenannten ASCII-Kode (American Standard Code for Information Interchange). Wenn man die Taste "Alt" festhält und die Nummer eines Zeichens eintippt (z.B. 129 für ü), dann die Alt-Taste wieder losläßt, erscheint an der Stelle des Kursors das betreffende Zeichen. Aber - wie oben schon erwähnt wurde - man kann die Tastatur mit Hilfe von speziellen Programmen umdefinieren. Nach dem Starten des umdefinierenden Programms kann man auch andere Zeichen mit je einer Taste auf den Bildschirm schreiben. Bei mit dem Programm KLAVGEN.EXE definierten Zeichensätzen kann man mit Ctr+Alt+F1 auf den ursprünglichen Zeichensatz umschalten, mit Ctr+Alt+F2 wieder auf den neuen.

Auf der Festplatte befindet sich immer ein File mit dem Namen AUTOEXEC.BAT, das nach dem Einschalten des Computers automatisch abläuft. Dieses File - wie überhaupt die Dateien mit der Extension .BAT - ist eine Verkettung von DOS-Kommandos, die nacheinander ausgeführt werden. Der Benutzer kann diese Mini-Programme ohne die Kenntnis irgendeiner Programmiersprache selber schreiben oder modifizieren; man braucht praktisch nur die DOS-Kommandos zu kennen, deren Zahl insgesamt nur ein paar Dutzende beträgt.

Da der Computer von vornherein in bedeutendem Maße zu militärischen Zwecken verwendet wurde, suchte man vom Anfang an nach Möglichkeiten, wie Software mit Hilfe anderer Software ge- oder zerstört werden könnte. So werden die ersten Computerviren entstanden sein, die dann auch in das Zivilleben der sich immer rascher computerisierenden Ge-

sellschaft Eingang fanden. Sie gehören zu unserem Alltag, es werden täglich neue Arten "hochgezüchtet". Von wem? Z.T. sind es Scherzbolde, die das Virusfabrizieren ohne Entgelt, als eine Art Hobby betreiben, z.T. aber Leute, die ihre Bestellungen möglicherweise von Softwarefirmen bekommen, die gleich darauf auch ein Viruskiller-Programm schreiben, das man legal verkaufen kann. Das gilt zum Glück nicht für alle Firmen. Es erscheinen fast monatlich neue Versionen von Virustötern auf dem Markt, die auch die neusten Viren mit Erfolg bekämpfen sollen.

Philosophiestudenten, die während des Studiums und später als Akademiker mit dem Computer überhaupt etwas zu tun haben, sind größtenteils Benutzer von Textverarbeitungsprogrammen. Die Bedienung eines solchen Programmes muß man vorher natürlich auch erlernen. Das Beherrschen derartiger Kenntnisse wird heutzutage immer mehr eine Voraussetzung zum vollwertigen Diplom.

Jedes Textverarbeitungsprogramm speichert die Texte in einem speziellen Format, so daß die Textdateien mit anderen Textverarbeitungsprogrammen nicht kompatibel sind (zum Glück gibt es auch Programme, welche die Texte von einem Format in ein anderes konvertieren). Die zur Zeit populärsten Programme sind MS-WORD, Word for Windows, Word Perfect. Für professionelle Zwecke, d.h. bei Verlagen, Redaktionen, Druckereien verwendet man oft auch das Programm Ventura.

Die meisten Textverarbeitungsprogramme haben verschiedene Versionen für die einzelnen Sprachen. Sie unterscheiden sich dadurch, daß die Kommunikationssprache des Programms (d.h. Menü, Rückmeldungen, Hilfe usw.), die Überprüfung der Rechtschreibung, Silbentrennung, der aktive Zeichensatz (Alphabet) der jeweiligen Einzelsprache entsprechen.

Der Mensch kann dem Computer Befehle geben, wozu eine Kommunikationssprache nötig ist. Die Sprache, die ein jeder Rechner versteht, ist der Maschinencode, den ein Mensch sehr schwierig bewältigen kann. Etwas näher der menschlichen Denkweise steht die Assembly-Sprache, die nicht nur mit binären Zahlen operiert und in der die einzelnen Symbole mitunter für komplizierte Kombinationen von Maschinencode-Schritten stehen.

In Assembly können heutzutage nur die besten Profis frei programmieren. Nunmehr herrschen die höheren Programmiersprachen vor wie Pascal, C oder Prolog. Ihre Symbole sind aus einer natürlichen Sprache, dem Englischen entlehnt, solche wie z.B. *write, do .. until, begin* usw. Diese Wörter können in Assembly mit wesentlich mehr Symbolen ausgedrückt werden, vom Maschinencode ganz zu schweigen. Aber da der Computer letzten Endes doch nur Maschinencode versteht, müssen die in höheren Programmiersprachen abgefaßten Programme in den Maschinencode übersetzt, kompiliert werden. Zu jeder Programmiersprache gibt es auch (mindestens) ein Compiler-Programm (der Compiler für Assembly heißt Assembler). Künftige Generationen der Computer sollen sogar in natürlichen Sprachen formulierte Aufgaben in Maschinencode übersetzen können. Die höheren Programmiersprachen sind nicht mehr so weit von diesem Ideal entfernt, es sind nämlich folgende Ausdrücke für Computer völlig klar: if result = the_number and if the_number > 0 then write "Okay." (eine kurze Beschreibung der Grundsätze der Programmiersprache BASIC befindet sich im Anhang 11).

2.
Bereiche der maschinellen Verarbeitung von natürlichen Sprachen

Beim Sprachgebrauch werden Gedanken kodiert übermittelt. Der Sprecher muß seine Gedanken kodieren, der Hörer - dekodieren. Die Kommunikationspartner müssen einen gemeinsamen Kode beherrschen - das ist die natürliche Sprache, von der sie Gebrauch machen. Wenn man den Kreis der potentiellen Kommunikationspartner einschränken will, muß man den allgemein bekannten Kode verändern. So ein Kode ist z.B. der **Geheimkode**, den man im Krieg gebraucht. Im II. Weltkrieg benutzte man bereits elektromechanische Rechner, die den Geheimkode feindlicher U-Boote entschlüsseln sollten. Dies war freilich kein unmittelbarer Kontakt zwischen Computer und Sprache. Aber zu einem unmittelbaren Kontakt ist es auch erstaunlich früh gekommen: 1949 verschickte Warren Weaver, Direktor bei der Rockefeller Stiftung einen Aufruf an 200 Wissenschaftler. Im Aufruf schilderte er die Idee der **Übersetzung** aus einer Sprache in eine andere mit der Anwendung des Computers. Vielen von den Adressaten gefiel der Gedanke, und sie machten sich auch gleich aktiv: es begann eine intensive Erforschung der Möglichkeit einer maschinellen Übersetzung. Nach der ersten Euphorie kam eine längere Periode der pessimistischen Resignation im Bereich der automatischen Übersetzung. In letzter Zeit erlebt jedoch auch dieser Teil der Computerlinguistik eine Renaissance.

Ein Übersetzungssystem hat auch solche Module, die - z.T. modifiziert - auch zu anderen Zwecken verwendbar sind. Für andere Anwendungsbereiche werden natürlich auch unabhängig von der MÜ spezifische Forschungen unternommen. Welches sind die anderen Bereiche der rechnerischen Verarbeitung von natürlichsprachlichen Daten?

Es sind vor allem vielleicht die speziellen **Datenbanken**, die man in ihrer herkömmlichen Form gemeinhin **Wörterbücher** nennt. Heutzutage sind kleinere Computerwörterbücher im Taschenrechner-Format vielerorts zu kaufen, u.z. zu einem fairen Preis.

Aber auch die Zusammenstellung von gedruckten Wörterbüchern - wie überhaupt die Erstellung von Drucksachen - ist heute ohne Computer kaum vorstellbar. Man kann die in beliebiger Reihenfolge eingegebenen Wörter jederzeit alphabetisch ordnen oder sogar die Umkehrung des Wörterbuchs ausführen, d.h. z.B. aus einem deutsch-ungarischen Wör-

terbuch ein ungarisch-deutsches machen, das allerdings meistens noch einer Überprüfung bedarf.

Die Lexikographie wurde sogar - eigentlich dank dem Computer - mit einem neuen Typ der Wörterbücher bereichert: mit dem **rückläufigen** Wörterbuch. Ein sogenanntes rückläufiges Wörterbuch kann mit vergleichsmäßig einfachen Programmen schnell und einfach aus jedem normalen Wörterbuch hergestellt werden. Wie ist so ein Wörterbuch aufgebaut, und was bezweckt man mit seiner Erstellung? Die Wörter sind darin nicht vom Wortanfang, sondern vom Wortende her geordnet, d.h. von rechts nach links (lateinisch: a tergo). Demnach stehen Wörter mit gleichem Wortausgang nacheinander. Gustav Muthmanns "Rückläufiges deutsches Wörterbuch" [27] enthält z.B. etwa 175000 Wörter, von denen *Samba* oder *Yucca* auf der ersten Seite, dagegen *Aufputz* oder *Abwehrgeschütz* auf der letzten Seite zu finden sind. Das ist auch eine alphabetische Ordnung, aber nicht die der Wörter, sondern die der Spiegelbilder der Wörter (vgl. Anhang 1). Woran ein Mensch mit Stift und Papier jahrelang arbeiten müßte, und dies wäre eine langweilige, mechanische, also unmenschliche Arbeit, das verrichtet eine Maschine innerhalb von ein paar Stunden.

Wozu dienen aber diese Wörterbücher? Das kommt auch darauf an, was noch mit den Wörtern zusammen eingegeben wird. Das rückläufige Wörterbuch der ungarischen Sprache [29] enthält z.B. praktisch alle morphologischen Informationen über die einzelnen Wörter (siehe Anhang 2). Ähnlich ist auch das Grammatische Wörterbuch der russischen Sprache konzipiert [53]. Beide Bücher sind offensichtlich für den kreativen Anwender gedacht, der nach Zusammenhängen zwischen Flexionstypen und Wortausgängen sucht.

Lexikographen bedienten sich auch früher u.a. der Methode, daß sie große Mengen von Texten untersuchten, um Wörter für ihre Wörterbücher zusammenzubringen. Dies war ziemlich zeitaufwendig, bis man den Computer auch dabei zu Hilfe rief. Heutzutage sind Riesenmengen von Texten auf Datenträgern gespeichert, z.T. in Datenbanken, z.T. in Textverarbeitungsdateien (z.B. in Druckereien). Diese Texte kann man jederzeit auch zur Herstellung von Wörterbüchern benutzen. Sollten sie

nicht ausreichen, so kann man heute schon gedruckte Texte mit Hilfe eines Scanners (d.h. "Abtasters") und eines Programms zur Zeichenerkennung direkt eingeben, d.h. ohne daß man sie wieder eintippt ("Zeichen" nennt man zusammenfassend Buchstaben, Ziffern, Interpunktionszeichen, mathematische Symbole und überhaupt alles, was im gesamten Zeichensatz des Computers enthalten ist).

Mit Computerunterstützung hat man den **Wortschatz einiger Dichter** inventarisiert, indem man sämtliche Werke dieser Dichter in den Computer eingegeben hat (z.B. Ady-Wörterbuch). Heute sind die Klassiker sowieso schon in den Datenspeichern der Druckereien.

Bei derartigen Bearbeitungen geht es darum, Lexeme zu fixieren, die in dem Korpus, d.h. in der gesamten untersuchten Textmenge mindestens einmal auftauchen. Man kann sich aber auch ein anderes Ziel setzen: es kann zusammengezählt werden, wie oft die einzelnen Wörter auftauchen. Das Resultat ist eine Häufigkeitsliste, oder - bei ausreichender Quantität - ein **Häufigkeitswörterbuch** (vgl. Anhang 3). Das kann sowohl im Sprachunterricht als auch in der maschinellen Bearbeitung der natürlichen Sprache (natural language processing) recht nützlich sein.

Da beide Wörterbuchtypen Lexeme enthalten sollen, die in den Texten durch ihre Wortformen repräsentiert werden, muß ein spezielles Programm in den flektierten Formen den jeweiligen **Stamm** finden. Bei solchen Sprachen, wo beispielsweise auch *itatgathatnának* eine Wortform ist, deren Stamm ein Buchstabe ist, kann diese Aufgabe mitunter recht "amüsant" werden.

Für linguistische Forschungen, aber auch für den Sprachunterricht ist es von großer Bedeutung, was in der Umgebung der einzelnen Wörter auftreten kann. Ein Computer, der die Wörter aus Texten extrahieren kann, vermag natürlich die jeweilige Umgebung dieser Wörter auch zu fixieren. Das Ergebnis nennt man **Konkordanzliste** (oder Konkordanzwörterbuch). Konkordanzprogramme funktionieren ziemlich mechanisch, indem sie eine festgelegte Anzahl von benachbarten Wörtern oder Zeichen mit dem Wort zusammen herausnehmen (siehe Anhang 4). Indessen kommt es aber nicht selten vor, daß die strukturell nächsten Formen topolo-

gisch entfernt sind (z.B. obligatorische Aktanten deutscher Verben). Diese Programme sind also mit einer menschlichen Intelligenz nicht zu vergleichen. Viel intelligentere Software braucht man hingegen, wenn der **Inhalt** eines Textes kurz resümiert werden soll. Und dies wird gegenwärtig auch weit und breit praktiziert.

Wie es in Prognosen von Computerexperten heißt, sollen künftige Rechnergenerationen in natürlichen Sprachen programmiert werden. Im Augenblick ist es noch eine Fiktion, aber nicht mehr weit von der Realität entfernt. Es gibt nämlich schon funktionierende **natürlichsprachliche Schnittstellen** (natural language interfaces), d.h. Programme, mittels derer die Computer in einer natürlichen Sprache kommandiert werden können. Solche Schnittstellen werden meistens an Datenbasisverwaltungsprogramme angeschlossen. Die Eingabe ist dabei schriftlich, denn eine **mündliche Eingabe** wäre hier ein Luxus.

Es gibt jedoch Gebiete, wo das "Hörverstehen" seitens des Computers schon wichtiger erscheint. Es gibt schon Roboter, die eine begrenzte Anzahl von Wörtern als Kommandos identifizieren und diese dann ausführen können (z. B. auf - rechts - links - stop usw.). Dazu braucht die künstliche Intelligenz nicht reicher zu sein als die eines Hundes.

Ein höheres Niveau der Intelligenz muß aber z.B. das japanische Übersetzungssystem besitzen, dessen Erprobung 1993 veröffentlicht wurde. Dieses System übersetzt nämlich aus dem Japanischen ins Englische und umgekehrt, und das alles mündlich, wie ein Dolmetscher. Es ist in ein Telephonsystem integriert, so daß sich ein Japaner mit einem Amerikaner nunmehr telefonisch verständigen kann, ohne die Sprache des anderen beherrschen zu müssen.

So ein System muß aber nicht nur mündliche Rede verstehen, sondern auch die Übersetzung verständlich aussprechen können. Dieser Aufgabe soll ein **Redesynthetisator** gerecht werden, der aufgrund einer Reihe von Graphem-Phonem-, bzw. Phonem-Phonzuordnungsregeln die schriftlich kodierten natürlichsprachlichen Daten sozusagen "laut vorlesen" kann. Die Synthese einer verständlichen (aber nicht unbedingt naturgetreuen) Rede ist wesentlich leichter zu schaffen als das Hörverstehen

seitens des Rechners. Schon ein einige hundert K großes Programm kann mit Hilfe des zur Grundausstattung gehörenden Tongenerators (also ohne zusätzliche Hardware) eine größtenteils verständliche Rede produzieren.

Linguistik wird sogar in die **Textverarbeitung** einbezogen: die Überprüfung der **Rechtschreibung** oder automatische **Silbentrennung** ist bei Sprachen mit reicher Morphologie (z.B. Ungarisch) ohne morphologische Analyse unvorstellbar.

Der sprechende Mensch drückt seine Gedanken, Beobachtungen, Gefühle usw. meistens mit einer Reihe zusammenhängender Sätze aus. Diese Tätigkeit ist ziemlich komplex, schwierig formalisierbar, trotzdem gab es bereits in den 70er Jahren funktionierende **Textgenerierungsprogramme.**

Die Implementationen von modellierten psychophysischen Phänomenen ist jeweils eine Erprobung des Modells, also ein Versuch, es zu bekräftigen. Es gibt aber meistens auch andere, praktische Zwecke, d. h. man sucht gleich nach Anwendungsmöglichkeiten in der Praxis. Wie kann Textgenerierungssoftware angewandt werden? Vor allem als ein Bestandteil der natürlichsprachlichen Schnittstellen von Datenbanken oder Expertensystemen (z. B. TEXT von McKeown[1] oder KAFKA, der Textgenerator des Expertensystems XCALIBUR[2]). Diese Texte sind Antworten auf die Fragen des Benutzers, die Informationen werden einer Wissensbasis entnommen, und eine interne Grammatik sorgt für richtige Strukturen. Die Texte sind nicht lang, sie können auch aus einem einzigen Satz bestehen.

Das Generierungssystem ANA[3] hat andere "Redeabsichten", es kann nämlich Berichte zu verschiedenen Themen abfassen, z.B. in den Bereichen Meteorologie, Wirtschaft, Handel, wobei es sich auf die aktuellen Werte der beobachteten Variablen stützt (Temperaturen, Börsen-Index usw.).

Der synthetisierende Teil der Übersetzungssysteme generiert letzten Endes auch Texte, aber die Ausgabe wird hierbei nicht nur von einem Ausgangsinhalt, sondern auch von formalen Eigenschaften des Quelltextes bestimmt (z. B. was möglichst mit einem Wort, mit einem Satz usw.

übersetzt werden soll).
Ein Programm, das sogar die Inhalte selber "ausdenkt", hat in der Alltagspraxis wenig Nutzen. Es kann nur fiktive Geschichten verfassen, aber Schriftsteller können es besser. Den theoretischen Wert solch eines Programmes darf man jedoch nicht unterschätzen, es muß nämlich eine Reihe von Prozessen simulieren, die im menschlichen Gehirn ablaufen, wie z.B. Folgerungen, Problemlösungen usw. So ein Programm ist beispielsweise TELLTALE von Correira[4].
Alle bislang erwähnten, nicht-linguistischen Anwendungen setzen linguistische Grundlagen voraus, und zugleich tragen sie - direkt oder indirekt - zu weiteren linguistischen Forschungen bei. Darüber hinaus gibt es auch solche Programme, die in erster Linie **linguistische Untersuchungen unterstützen** sollen. A-tergo-Listen oder Lexemhäufigkeitslisten werden auch vor allem von Linguisten benutzt, aber bei diesen Programmen kann man auch andere Anwenderkreise nicht ausschließen (Sprachen lernen, Verse oder Rätsel schreiben usw.). Nur Linguisten arbeiten dagegen mit Programmen, die **Phonemstatistiken, Phonemhäufigkeit, Phonemkonkordanz** liefern. (Letzteres ergibt die möglichen Phonemkombinationen, ohne deren Kenntnis ein vollständiges phonologisches Regelsystem nicht aufgebaut werden kann.) Mit Hilfe des Computers kann man auch nach anderen Arten von **Regeln suchen.** Oft ließe sich z.B. die linguistische Beschreibung vereinfachen, wenn man die Möglichkeit gewisser syntagmatischer Verbindungen ableiten könnte von formalen und/oder semantischen Merkmalen der beteiligten Elemente. Wenn es um große Mengen Wörter oder anderer Einheiten geht, kann die Suche nach Kookkurenzen recht langwierig werden. Deutsche Substantive z.B. lassen sich (fast) immer nur mit einer von drei parallelen Artikelformen verbinden. Inwieweit dies von ihrer Form und/oder Bedeutung ableitbar ist, kann ein Computer ermitteln, wenn man in ihn alle Substantive samt ihren Merkmalen eingibt. Auf dieses Thema wollen wir später noch zurückkommen.
Die Messung der **Lexemhäufigkeit** ist nur mit Hilfe eines morphologischen Programmes möglich, das aber ohne lange Stamm- und Affixlisten nicht funktionieren kann. Die **Häufigkeit von Wortformen** läßt sich

viel einfacher messen: es müssen nämlich nur die völlig übereinstimmenden Formen erkannt und zusammengezählt werden (vgl. Anhang 5). Derartige Listen kann auch ein kurzes Programm, wie z.B. Longman's Mini-Concordancer, zusammenstellen (siehe Anhang 6).

Bei phonologischen Untersuchungen kann man dem Computer den sprachlichen Stoff gleich in phonologischer Transkription angeben, aber bei Aufgaben, die großer Datenmengen bedürfen, ist das so energie- und zeitaufwendig, daß es sich überhaupt nicht mehr lohnt (vgl. Anhang 7). Wenn man dagegen einen Algorithmus herstellt, der die orthographische Schrift automatisch **in phonologische Schrift umsetzen** kann, so kann der Computer einen großen Teil der mechanischen Arbeit übernehmen. So ein Algorithmus enthält einerseits die regelmäßigen Graphem-Phonem-Entsprechungen (z.B. daß ein *p* vor einer Leerstelle immer einem /p/ oder die Graphemkombination *sch* einem /ʃ/ entspricht), andererseits aber eine Liste der Ausnahmen (Häus|chen, be|urteilen, be|enden, be|inhalten, wach|st - aber: wachs|t). Mit Hilfe einer morphologischen Analyse kann man die meisten Fälle richtig behandeln, aber die Analyse braucht auch eine Liste - die der Stämme -, sonst gibt es noch mehr Fehler als ohne Analyse (z. B. Be|in oder was|chen, wenn man einfach Affixe abtrennt).

In phonologisch transkribierten Texten kann man die einzelnen Phoneme zusammenzählen, und derartige Phonemstatistiken liefern Informationen über die Phonemhäufigkeit. Auch **onomatopoetische Untersuchungen** operieren oft mit Phonemen und nicht mit Graphemen. Onomatopöie oder Lautsymbolik liegt im Grenzgebiet von Literaturwissenschaft und Linguistik. Sie beschäftigt sich damit, welche Gefühle und unbewußte Assoziationen die Lautgestalt der Wörter im Sprecher/Hörer wecken kann. Die Phoneme werden isoliert bewertet, möglichst aufgrund der Meinungen vieler Versuchspersonen. Sie werden z.B. befragt, ob ein /i/ klein oder groß ist, bzw. wo es sich auf der Skala *klein-groß* befindet (weitere Merkmalpaare: männlich-weiblich, hart-weich, aggressiv-mild usw.). Auf diese Weise erhält man konstante Werte, welche die Phoneme charakterisieren. Einige Positionen wirken verstärkend, so z.B. der Anlaut oder die betonte Silbe. Den Positionen werden

spezielle Multiplikationszahlen zugeordnet. Wenn man dann die Merkmalwerte der Phoneme eines Textes addiert, erhält man eine lautsymbolische Charakterisierung des ganzen Textes. Bei literarischen, ästhetischen Untersuchungen kann z.B. auf diese Weise erforscht werden, inwieweit Klang und Inhalt eines Werkes übereinstimmen.

Phoneme werden natürlich meistens von linguistischen Aspekten aus ermittelt. So sind beispielsweise **phonotaktische Regelmäßigkeiten** der einzelnen Sprachen Phänomene, die vor allem für die Linguistik interessant sind. Mit Hilfe des Computers kann man sämtliche Phonemkombinationen auflisten, die in den verschiedenen Positionstypen vorkommen (Anlaut, Inlaut, Auslaut, Silbengrenze, Morphemgrenze usw.). /rt/ kann z.B. im deutschen Auslaut (Ort), im Inlaut (Torte), aber nie im Anlaut stehen (vgl. aber in slawischen Sprachen: rtęć (poln), rtut' (russ)).

Wenn die Phoneme durch Merkmalbündel repräsentiert werden, kann der Computer sogar **Verallgemeinerungen** vornehmen (z. B. [Son]+[Obstr.] im Anlaut ist ausgeschlossen, aber [Obstr]+[Son] nicht, siehe *klein, braun, schmeißt, Schnitt).*

Der Weg von den Phonemen zu den **Phonen**, d.h. Sprechlauten ist kürzer und einfacher als der Weg von den Graphemen zu den Phonemen. In der phonologischen Schrift sind nämlich schon die Morphemgrenzen mit einbezogen, was ohne eine vorangehende morphologische Analyse unvorstellbar wäre. Um von der Ebene der Phonologie auf die Ebene der Phonetik zu gelangen, braucht man nur noch die Koartikulationsregeln sowie das An- und Auslautgesetz. Laut einer Regel entspricht z.B. dem Phonem /g/ ein Phon [k] in der Wortform *sagt*. Die so erhaltene phonetische Transkription (samt intonatorischen Merkmalen) ist dann gut geeignet, als Grundlage einer synthetisierten Rede zu dienen.

Auf höheren Ebenen der Sprache (Morphologie, Syntax usw.) bedient sich die Computerlinguistik (und eigtl. die Linguistik überhaupt) nur recht selten phonologischer Repräsentationen: man operiert eher mit orthographischen Repräsentationen (in den meisten deutschen Grammatiken kann man z.B. ähnliches lesen: "die Endung der 2 P. Sg. ist statt *st* nur *t* nach *s*, *ß*, *z* und *x*"; dabei handelt es sich einfach jeweils

um ein [s] im Stammauslaut, das mit dem [s] der Endung verschmilzt).
In der Morphologie sind die morphotaktischen Möglichkeiten schon beschrieben, da kann man von Untersuchungen langer Texte wenig neue Informationen erwarten. Was noch verfeinert werden kann, das sind die Erklärungen für Verbindungsmöglichkeiten. Je mehr Erklärungen gefunden werden, desto einfacher, ökonomischer kann man die sprachlichen Formen beschreiben.
Der Computer kann bei der **Suche nach erklärenden Merkmalen** helfen.
Morphologische Regeln kann man grundsätzlich von drei **Regeltypen** ableiten. Zwei davon sind kontextabhängig (umgebungsabhängig), eine nicht.
Man würde nur sehr wenig Regeln brauchen, wenn eine Zeichenform immer mit demselben Inhalt verbunden wäre (also wenn es keine Polysemie und Homonymie gäbe) und ein Inhalt immer nur durch ein und dieselbe Form wiedergegeben werden könnte (also wenn es keine Synonymie gäbe). Dann würden Wortstellungsregeln genügen und die Regeln, die besagen, was den Inhalten der Bestandteile der Zeichenkombinationen eventuell noch zukommt.

Z.B. *Giftgas*	- Gas, das giftig ist / Gift, das Gas ist
Flaschengas	- Gas, das in einer Flasche ist
Gasflasche	- Flasche zum Aufbewahren von Gas (Stahlgefäß)

Aber sprachliche Zeichen sind polysem/homonym und synonym, also brauchen wir noch viele Regeln, um entscheiden zu können, welche von den synonymen Formen in den einzelnen Umgebungen (eigtl. Kontexten) zu wählen sind. Z.B. -s, -n, -ns im Genitiv Sg.:
des *Käses, Jungen, Namens*
(Hier sind natürlich nicht die Stämme, sondern die Endungen synonym; sie haben ein und dieselbe grammatische Bedeutung.)
1. Regeltyp: Die kontextabhängige Auswahl der Elemente in den einzelnen Positionen ist durch eine Verkettung von elementaren Regeln zu beschreiben. Die elementare Regel operiert jeweils mit 4 Komponenten, von denen nur 3 bekannt sind, die vierte wird von der Regel bestimmt.

Form 1	Form 2	← "Grammatik des Sprechers"
Inhalt 1	Inhalt 2	← "Grammatik des Hörers"

mein	est st t ?
MEINEN	2.P.Sg.Ind. Präs. Akt.

arbeit|est
heiß|t

← Sprecher

fähr	t
FAHREN	3.P.Sg.Ind.Präs.Akt. 2. P. Pl. " 2. P. Pl. Imp. ? Part. Perf.

← Hörer

ihr fahrt
fahrt!
studiert

Derartige minimale Kontexte reichen nicht immer aus, die einzige richtige Variante zu finden; manchmal vermindern sie bloß die Zahl der Möglichkeiten. In diesen Fällen muß man den Kontext erweitern (z.B. Fahr + t - *Fahrt ihr mit?; Die Fahrt dauert eine Stunde*).

Es ist auch (1) eine Regel, aber nicht ökonomisch genug; Wenn wir alle möglichen Umgebungen der Endungen *-st*, *-est* und *-t* untersuchen, können wir Verallgemeinerungen vornehmen.

(1) "Wenn der Stamm *arbeit* ist, dann kommt die Endung *est* dazu".

Dann können wir statt einer langen Liste mit solchen Verben wie *arbeiten, melden, warten, senden, atmen, rechnen* usw. nur ihre gemeinsamen Merkmale nennen; dentaler Verschlußlaut am Ende oder

$$\begin{bmatrix} +\text{Konsonant} \\ -\text{Liquid} \end{bmatrix} \begin{bmatrix} +\text{Nasal} \end{bmatrix}$$

(nur bei unverändertem Stammvokal, vgl. lädst, trittst)

Solche Merkmale sind direkt beobachtbar, man kann sie ohne zusätzliche Informationen über die kontextbildenden Elemente ermitteln.

2. Regeltyp: Diese Regel stützt sich auf eine andere Art von Merkmalen. Solche Merkmale sind nicht direkt im Kontext beobachtbar, aber als grammatische Informationen sind sie im Lexikon der Sprecher mit den Elementen eng verbunden. So z. B. die Valenz bzw. die Rektion:
haben Nom, Acc; *gehören* Dat, Nom,
oder der Konjugationstyp (d.h. z.B. ob ein Verb den Umlaut bekommt: *klagen, klagst; schlafen, schläfst*).
Diese Merkmale sind paradigmatische Merkmale des kontextbildenden Elementes, die besagen, welche Ausfüllungsvariante des Paradigmas zum betreffenden Element paßt. Jedes Paradigma besteht nämlich aus Leerstellen, die ausgefüllt werden sollen.
3. Regeltyp: Dieser Regeltyp ist nicht kontextabhängig. Einige synonyme Gruppen können ökonomischer beschrieben (gespeichert) werden, indem man gewisse Glieder der Gruppe von anderen Gliedern derselben Gruppe ableitet. In der deutschen Grammatik spielt dieser Regeltyp eine weniger bedeutende Rolle als im Ungarischen oder im Russischen. Eine der wenigen Regeln dieser Art lautet folgenderweise:
"Wenn ein Adjektiv- oder Verbalstamm auf *-el* auslautet, hat er auch eine Variante ohne *e*." Z.B. sammel/n, samml/e
dunkel, dunkl/e
Es erinnert der erste Regeltyp am ehesten an eine Art Erklärung, weil dabei sprachliche Größen aufgrund bestimmter, von ihnen unabhängig existierender Eigenschaften ausgewählt werden. Diese Eigenschaften sind phonologische, morphologische, semantische Merkmale eines sprachlichen Elementes, die als solche auch sonst vorhanden sind, unabhängig von der Existenz anderer Elemente.
Beim zweiten Typ hängt die Wahl von solchen Eigenschaften ab, die von den auszuwählenden Größen bestimmt werden: Eigenschaften wie Valenz oder Flexionstyp existieren nur, um das Element mit anderen richtig zu verbinden. In konkreten Fällen können die Grundtypen der Regeln allerdings auch gemischt auftreten.

> Wie man derartige Regeln mit Computerunterstützung suchen kann, möchte ich an einem konkreten Beispiel illustrieren. Vor einigen Jahren habe ich versucht, mit Hilfe eines Großrechners Zusammenhänge in der Bildung von russischen denominalen Adjektiven zu

finden. Das Thema hatte mich seit langem beschäftigt. Es kam mir nämlich merkwürdig vor, daß solche Adjektive tagtäglich entstehen, aber sie werden von den Substantiven mit verschiedenen Suffixen abgeleitet, wobei die Auswahl des einen oder des anderen Suffixes scheinbar ad hoc vor sich geht. Im Deutschen (aber meistens auch im Ungarischen) werden in diesen Fällen zusammengesetzte Wörter gebildet (z.B. Bücherregal). Eine globale Regel der Adjektivbildung, die die Entscheidungen der Wortschöpfer erklären könnte, gab es jedenfalls nicht.

Meinen Untersuchungskorpus entnahm ich dem Akademischen Wörterbuch in 4 Bänden. Dort habe ich mehr als 6,5 tausend denominale Adjektive gefunden. Interessanter waren für mich jedoch nicht die Adjektive, sondern ihre motivierenden substantivischen Stämme. Ich habe all diese Substantive in den Computer eingegeben. Die Wörter habe ich in 10 semantische Klassen eingeordnet (z.B. Mensch, Institution usw.). Außer dem semantischen Merkmal habe ich auch den stilistischen Wert des Wortes codiert. Die formalen Merkmale hat ein spezielles Unterprogramm aufgrund der eingegebenen Wortform expliziert und jeweils in einem Rekord mit dem Substantiv zusammen gespeichert. Diese Merkmale waren: der letzte Buchstabe des Substantivs (im Russischen steht dieser Buchstabe im engen Zusammenhang mit dem Deklinationstyp des betreffenden Wortes); Zahl der Silben, Position der betonten Silbe (dies mußte extra angegeben werden) und die phonologischen Merkmale der letzten drei Phoneme des Stammes. Letzteres setzte eine Graphem- Phonem-Konversion voraus. Für die russische Sprache läßt sich solch ein Algorithmus leichter erstellen als für die deutsche, aber sehr leicht war es auch nicht (besonders die Opposition "weich-hart" wird durch die Orthographie sehr kompliziert ausgedrückt).

Bei jedem Substantiv mußte unbedingt noch eine Angabe stehen: das Suffix, mit dem von ihm ein Adjektiv gebildet wird.

Der Computer hatte nun die Aufgabe, diejenigen Merkmale auszuwählen, die immer oder fast immer mit demselben Suffix zusammen vorkommen. Dabei mußte der Computer auch die Hierarchie der Merkmale berücksichtigen; die Merkmale bestimmten nämlich Wortmengen, d.h. Gruppen der Wörter, in denen sie enthalten waren. Wenn eine Wortmenge eine kleinere Wortmenge enthält, d.h. völlig einschließt, dann muß das bestimmende Merkmal der kleineren Menge ignoriert werden. Z.B. eine gewisse Phonemverbindung am Enden des Stammes kommt in 20 Wörtern vor, und sie bekommen alle das Suffix *-ow-*; aber die 20 Wörter sind alle einsilbig, ebenso wie noch weitere 200, und die einsilbigen stehen auch immer mit *-ow-*. Folglich fällt die Phonemverbindung als Merkmal weg, und es bleibt nur die Einsilbigkeit.

Es waren freilich wenig solche Merkmale zu finden, die immer und ausschließlich mit demselben Suffix vorkamen. Zuerst wurden diejenigen ausgewählt, welche die besten Prozentverhältnisse aufwiesen, d.h. bei denen verhältnismäßig wenig Ausnahmen entstanden. Dann wurde je-

weils nur noch der Rest untersucht. Mit diesem Verfahren wurde eine algorithmische Regel aufgestellt, die mehr als 90% der 6,5 tausend Adjektive erzeugt, d. h. vom jeweiligen Substantiv ableitet.
Aufgrund dieses Algorithmus kann man mit einem hohen Sicherheitsgrad auch neue Bildungen vorhersagen, so daß die Regel z.B. auch bei der automatischen Übersetzung neuer deutscher Komposita ins Russische angewandt werden kann.

3.
Computergestützter Sprachunterricht (CALL): ein Überblick

Typen der Unterrichtsprogramme

Im Fremdsprachenunterricht sind ziemlich viele Medien eingesetzt worden, für die das Interesse nach einer kurzen Euphorie-Phase oft stark abgenommen hat. Dies passierte u. a. dem Sprachlabor oder dem "Programmierten Unterricht" der 60-er Jahre. Der Computer jedoch, der bereits beim Programmierten Unterricht eine wichtige Rolle spielte, ist nach einer Pause in den Unterrichtsprozeß zurückgekehrt, und dies ist schon ein Beweis dafür, daß er nicht einfach als eines der Unterrichtsmedien anzusehen ist. Der Computer, der vor kurzem im Unterricht wieder erschienen ist, ist freilich kein Computer mehr aus den 60-er Jahren: ein PC ist viel leichter zu bedienen, und wegen des Preissturzes gibt es gegenwärtig in den einzelnen Schulen sogar in Ungarn meistens mehr Rechner als damals im ganzen Land.

Daß Computer im Sprachunterricht trotzdem recht selten benutzt werden, liegt einerseits an einer Schwellenangst der Lehrer, andererseits an mangelnder Software. Es fehlen außerdem die entsprechenden Lehrpläne, die u. a. die Unterrichtsformen bestimmen, die ein effektives Kombinieren von computerisierten und sonstigen Lehrstoffen ermöglichen.

Der Struktur und Funktion nach kann man die CALL (Computer Assisted Language Learning) Programme in Typen einteilen. Die meisten der bislang hergestellten CALL-Programme lassen sich am besten in der Einzelarbeit einsetzen. Es ist also kein Wunder, daß die Lehrer ratlos sind, wenn sie in der Stunde Programme aus dem jetzigen Angebot verwenden wollen.

Es gibt aber auch solche Programme, die nur einen Rechner im Klassenzimmer benötigen, und dabei der ganzen Klasse gleichzeitig eine Möglichkeit zur Aktivität bieten. Die **in der Unterrichtsstunde** einsetzbaren Programme gehören grundsätzlich zu zwei Typen: der eine unterstützt den Lehrer, der andere ersetzt ihn in gewissen Phasen des Unterrichts.

Zum ersten Typ gehören u. a. die sog. **Demonstrationsprogramme**, die Prozesse, Gesetzmäßigkeiten, Regeln und andere Phänomene veranschaulichen sollen. Sie werden vor allem in den naturwissenschaftlichen

Fächern verwendet (z.B. ballistische Kurven, Simulationen von Kettenreaktionen in der Atomphysik oder in ökologischen Mikrosystemen).
Im Sprachunterricht läßt sich nicht vieles mit Computern veranschaulichen; zu speziellen Unterrichtszwecken kann man eventuell bei Fortgeschrittenen Häufigkeitslisten erstellen.
Dem Lehrer können in der Stunde auch solche Programme behilflich sein, die zur Klassenarbeit Themen und andere **Stimuli** liefern. In der Sprachstunde können es z.B. zufällig generierte Konversationsvarianten sein. Dabei kommunizieren die Lerner nicht mit dem Computer, sondern miteinander und mit dem Lehrer, dessen Anwesenheit hier unerläßlich ist. Das Programm kann als eine Art Abenteuerspiel konzipiert werden, wo die kollektiven Entscheidungen als solche jeweils diskutiert werden müssen (B. Jones: GRANVILLE). Aber es kann auch ein Textgenerierungsprogramm sein, wie unsere KURZKRIMIS (dieses Programm wollen wir später, beim Thema "Textgenerierung" unter die Lupe nehmen).
Dem Lehrer können indirekt auch für **Einzelarbeit** konzipierte Programme helfen, indem sie einen Teil der Klasse beschäftigen, so daß der Lehrer mit dem Rest der Klasse intensiver arbeiten kann. Wenn jedoch alle Schüler mit dem Computer arbeiten und das Programm die falschen und die richtigen Lösungen kommentiert bzw. Regeln und andere Arten der Hilfe bietet, ist die Anwesenheit des Lehrers überflüssig.
Während man gegen den Einsatz des Computers in der Unterrichtsstunde damit argumentieren kann, daß es genügend Lehrer gibt und der Computer ihre Arbeit nicht ergänzt sondern ersetzt und somit die Arbeitslosigkeit fördert, kann man ähnliche Argumente gegen Lernprogramme für die Einzelarbeit außerhalb der Stunde nicht anführen. Es gibt nämlich schon jetzt mehr Computer in den Privathaushalten als Hauslehrer.
Diese Programme dienen meistens zur **Übung** und zum **Selbsttest**. Die neueren Programme können antizipierte **Fehler des Lernenden kommentieren, analysieren, Hilfe leisten**. Der Schwierigkeitsgrad der jeweiligen Aufgabe hängt oft von den früher begangenen Fehlern ab. Ein großer Vorteil des Computers ist dabei, daß der Lernprozeß weitgehend

vom Lerner bestimmt werden kann, was im schulischen Unterricht nicht der Fall ist. Der Lernende kann den Stoff in einem ihm passenden Tempo bearbeiten, er kann zwischen Schwierigkeitsstufen frei wählen, das Lernen jederzeit unterbrechen und dann fortsetzen, und dabei braucht er sich wegen der Fehler nicht vor den Mitschülern und dem Lehrer zu schämen; den Computerlehrer kann er sogar ohne Hemmungen beschimpfen. Mit dem Computer kann auch das langweilige Büffeln interessanter werden, schon wegen der prompten Bewertung. Wenn man dazu noch ein paar spielerische Tricks verwendet, kann sogar ein Der-die-das-Einpauken zu einem spannenden Spiel werden. Vor einigen Jahren haben wir für Heimcomputer wie Sinclair und Commodore ein Programm mit dem Namen Lexi-trap (Autoren: P. Uzonyi und L. Agócs) geschrieben. Eine gelungene Graphik und Toneffekte sorgten dafür, daß das Tontaubenschießen auf Wörter nicht nur Kinder, sondern auch Erwachsene stundenlang an den Computer fesseln konnte.

Ein Lernprogramm wird freilich noch zu keinem spannenden **Spiel**, wenn wir es einfach mit Zeichnungen und Musik ausrüsten. Ein Spiel muß z.B. Regeln haben, die ein Wetteifern mit sich selbst, mit einem Rekord oder/und mit anderen Spielern ermöglichen. In einem Lernprogramm sollte dabei die Leistung nicht davon abhängen, wie geschickt und schnell der Lerner mit den Tasten oder dem Steuerknüppel (Joystick) umgehen kann. Andererseits aber kann ein Programm auch dann zur Aneignung von lexikalischen und grammatischen Daten dienen, wenn das Spielergebnis nicht nur von den Sprachkenntnissen abhängt. Dies trifft auch für unsere SPOKER-Serie zu. In diesem Pokerspiel gewinnt nämlich nicht unbedingt derjenige, der mehr Vokabeln kennt, sondern der die Jetons geschickter setzt und schlauer blufft. Der Lerner ist jedenfalls interessiert daran, daß er sich die neuen Wörter merkt, denn so hat er mehr Chancen.

Die früheren Übungsprogramme waren primitiv in dem Sinne, daß sie die Lösungen mit "Richtig" oder "Falsch" bewerteten, kein Hilfe-File enthielten und keine Optionen boten. So ein Programm war DEUTADJ, das wir vor etwa 10 Jahren geschrieben haben, noch vor dem Anfang der Schulcomputer-Aktion im ungarischen Schulsystem. Mag das Programm

noch so primitiv sein, ich habe es vor kurzem doch für PC-s adaptiert, da es unser erstes Programm war, das aus Wörtern bzw. Morphemen selbständig Syntagmen herstellte (z. B. mit + dessen + neu + em + Tisch).

Satzgenerierung in CALL-Programmen[5]

Wieviel Typen von Sprachlernprogrammen auch immer entwickelt werden, ihnen ist gemein, daß sie mit konkreten sprachlichen Daten operieren. Übungsprogramme sollten dem Lernenden Sätze oder Texte in einer großen Anzahl liefern. Verfasser von solchen Programmen müssen also nach Methoden suchen, die ermöglichen, Sätze und Texte aus weniger Elementen herzustellen als die Gesamtzahl der Elemente in den Texten.
Es gibt ein einfaches Verfahren, das meines Wissens zuerst in unseren Lernprogrammen konsequent verwendet wurde. Nehmen wir die folgenden drei Sätze:

(1) Hans ging heute vormittag einkaufen.
(2) Sie soll Dienstag um eins zum Chef.
(3) Vati wollte auch ins Zentrum.

Aus diesen 17 Wörtern lassen sich leicht weitere 240 Sätze zusammenstellen, indem man die Wörter, die in derselben Position sind, miteinander austauscht, d.h. die möglichen Kombinationen herstellt. Diese Sätze lassen sich in 5 Positionen teilen, die aber nicht alle unbedingt ausgefüllt sind (hier ist es die 4.).

(4)	1.	2.	3.	4.	5
	Hans	ging	heute	vormittag	einkaufen.
	Sie	soll	Dienstag	um eins	zum Chef.
	Vati	wollte	auch	-	ins Zentrum.

Wenn wir den Positionen noch je ein Wort zuordnen, werden wir 1024 Sätze haben.

Man gewinnt weitere Kombinationen, wenn man auch die Wortfolge ändert. Z. B. die 16 (2^4) Sätze, die durch die folgenden 2 Sätze repräsentiert werden können, ergeben noch weitere 32 Sätze mit anderer Wortstellung.

(5)	1	2	3	4
	Er	fährt	am 12. Juli	nach Dresden.
	Die Gruppe	kommt	erst morgen	dorthin.

Es sind noch die Wortfolgen "3.2.1.4." und "4.2.1.3." möglich. In diesen Beispielen kann ein beliebiges Wort von jeder Position gewählt werden, aber die Repräsentationen können auch Verzweigungen enthalten. Auf diese Weise kann eine Art primitive Subkategorisation vorgenommen werden.

(6)				
Gestern	angelte	Klaus		zwei Stunden.
Diesmal	telefonierte	Herr Klein		eine Viertelstunde.
	dauerte	die Besprechung		
		der Waldlauf		
		es		

Was für Übungen kann man mit Hilfe dieser Sätze zusammenstellen? Vor allem braucht man sie **als Kontext** zu grammatischen oder lexikalischen Übungen. Z. B. Endungen, Artikel oder andere Wörter können weggelassen werden, die der Lerner in die Lücken hineinschreiben soll.
Die **Fehleranalyse** kann durch das kombinierende Verfahren auch zuverlässig gemacht werden. Das Programm SATZBAU2 (P. Uzonyi - Z. Papp) kann mehr als hunderttausend richtige und mehrere Billionen falsche Sätze generieren, es werden den einzelnen Positionen nämlich auch die antizipierten falschen Formen zugeordnet - wie *Kleid* im folgenden Beispiel:

Der Mann ist alt.
Dieser Wagen war da. (16 richtige und 8 falsche Sätze)
(Kleid)

SATZBAU2 weist auf den Charakter des begangenen Fehlers hin, indem es den Lerner im Telegrammstil an die verfehlte Regel erinnert (z. B. "Modalverben ohne "zu" verwenden!"). Das Programm kann 40 Fehlertypen identifizieren, und es kann je eine von diesen Erklärungen zu jedem einzelnen Fehler der vielen Billionen Sätze hinzufügen.
Es gibt Antworten, die gleichzeitig mehreren Fehlertypen angehören.
Z. B. "Ich weiß nicht, wohin er gegangen haben": *Falsches Hilfsverb und Nichtbeachtung der Kongruenz.* Außerdem kann vorkommen, daß der Schüler in der Tat eine andere Regel verfehlt, nicht die, welche das Programm als Erklärung angibt (z.B. "du eßt" - es wird möglicherweise ein Kongruenzfehler angezeigt, wobei der Schüler einfach nicht weiß, daß es ein Verb mit e-i-Wechsel ist).
SATZBAU2 identifiziert den Fehler jeweils nur in einer Hinsicht, was dazu führen kann, daß die nächste Antwort des Lerners von einem anderen Aspekt aus noch immer falsch wird. So kommt er schrittweise zur richtigen Lösung (z. B. erste Antwort: ..., daß er gegangen haben: *Anderes Hilfsverb*!; zweite Antwort: ...daß er gegangen sind: *Nichtbeachtung der Kongruenz*!). Es gibt auch Programme, die alle zutreffenden Fehlermeldungen gleichzeitig aufzählen.
Wenn Programmautoren keine fertigen Sätze speichern wollen, können sie nicht nur das Generieren verwenden. Sätze oder Texte können auch vom Benutzer (d.h. vom Lernenden oder seinem Lehrer) eingegeben werden. Solche Software nennt man **Autorenprogramme**. Bei der Konzipierung des Autorenprogrammes muß man berücksichtigen, daß es benutzerfreundlich sein muß, denn Benutzer verbringen ihre Zeit oft nicht gern mit Dateiauffüllung, besonders dann, wenn die Prozedur ihnen zu kompliziert scheint.
Mit Autorenprogrammen werden wir uns noch in einem anderen Kapitel beschäftigen.

Textgenerieren für CALL

Generierte Texte kann man auch im Unterricht verwenden. Der Lerner kann z.B. die Aufgabe bekommen, daß er in einem Text die Verflech-

tungsmittel finden und diese näher bestimmen soll. Dabei ist es einerlei, worum es sich im Text handelt, Hauptsache, daß er ein zusammenhängender, also kohärenter Text ist und das Programm die betreffenden Verflechtungsmittel auch identifizieren kann. Ein Generierungssystem mit Wissensbasis, Lexikon, Komponente der logischen Operationen, Grammatik usw., wie sie die o. a. Systeme haben, für diesen Zweck einzusetzen wäre wohl eine ziemlich luxuriöse Lösung. Es ist wesentlich einfacher beispielsweise, wenn man eine Anzahl Sätze zusammenbringt, die in demselben Themenkreis gebraucht werden können, aber ohne Fügewörter keine inhaltlichen Beziehungen zueindander aufweisen. So läßt sich ihre Reihenfolge beliebig verändern, außerdem kann man beliebige Konjunktionen oder Konjunktionaladverbien einsetzen und dadurch kopulative, adversative, konsekutive und andere Verhältnisse zwischen den Sätzen markieren. In diesem generativen Programm können also Sätze wie (1) und (2) nicht gebraucht werden, weil sie auch ohne Konjunktion ein kausales Verhältnis implizieren.

(1) Die Lehrerin tadelte den Schüler.
(2) Der Schüler hatte die Hausaufgaben nicht gemacht.

Die Sätze (3) und (4) lassen dagegen verschiedene Fügewörter zu.

(3) Herr Müller wurde zum Werkmeister befördert.
(4) Die Produktivität der Werkstatt sank um 10 %.

Wenn wir z.B. das Konjunktionaladverb "trotzdem" vor (4) einsetzen, wird dem Inhalt eine Präsupposition beigefügt, nach der man von Herrn Müller erwartet, daß er die Produktivität nicht sinken läßt. Wenn wir dagegen "nämlich" einsetzen, muß der Leser daran denken, daß Herr Müller nach der Beförderung die Produktivität steigern soll. Mit "deshalb", "außerdem" usw. wird den zwei Sätzen, die eigtl. auch als ein kurzer Text angesehen werden können, jeweils ein anderer Sinn verliehen. Wenn man die Sätze vertauscht, können weitere Bedeutungen ausgedrückt werden. Bereits 5 derartige Sätze, die ihrerseits auch aus kleineren Segmenten generiert werden können, ergeben eine so große Anzahl von Texten, daß es eine kombinatorische Explosion ge-

nannt werden kann (unser Programm ANALIZIS generiert ungarische Sätze auf diese Weise; Autoren: P. Uzonyi, L. Agócs).
Im Programm KURZKRIMIS (P. Uzonyi, F. Megyery, Z. Papp)[6] ist es nicht mehr ganz egal, wovon die Texte handeln. Die Maschine erzeugt eine Anzahl von Kriminalgeschichten mit Hilfe des Zufallszahlgenerators. Mit diesen Kurzgeschichten kann ein Lehrer vieles machen: ihren Inhalt erzählen lassen, Fragen stellen oder stellen lassen, die Geschichte fortsetzen lassen usw. Das Programm selbst ermöglicht zweierlei Anwendungen ohne Lehrer: der Lernende kann den Text Wort für Wort rekonstruieren (jeder Buchstabe wird zuerst durch ein Sternchen ersetzt), oder - was auf der grundlegenden Idee des Programms basiert - die Geschichte lesen und die zwei Verhörten finden, deren Aussagen einander widersprechen. Damit kann man das verstehende Lesen üben, während die Rekonstruktion den Wortschatz und die grammatische Kompetenz festigt.
Ein bedeutender Teil der sprachlichen Daten von KURZKRIMIS befindet sich als sequentielle Files auf der Festplatte. Jedes File beinhaltet je einen Aussagentyp, aus dem dann mehrere konkrete Aussagen zu generieren sind. Als erster Schritt kombiniert das Programm aus den insgesamt 60 Typen 5 so,daß es dabei auch gewisse Schlüsselinformationen berücksichtigt und dafür sorgt, daß von den 5 Aussagen zwei sich unbedingt widersprechen. Bevor das Programm die Files zu kombinieren beginnt, generiert es aus den Daten im operativen Speicher eine Grundsituation, d.h. wer, wann, wo, womit ermordet wurde. Davon hängt ab, welche Files nicht auszuwählen sind (wenn z.B. das Opfer eine Frau ist, kann das File nicht gewählt werden, in dem die Braut des Ermordeten erzählt, wie sie sich mit ihm gestritten hat). Davon hängt auch ab, welche Werte die Variablen in den Aussagen annehmen können. Wegen der Textkohärenz ist es unerläßlich, daß alle über dieselbe Person sprechen, oder daß die von verschiedenen Zeugen erwähnten Zeitpunkte koordiniert sind, usw.
Die Aussagentypen sind Mengen von kurzen Texten, deren Repräsentation den oben dargestellten Satzmengen-Repräsentationen ähnlich ist. Sie enthält keine Abzweigungen, nur Variablen, die von der Grundsituation

abhängige Werte annehmen (z.B. "ihn" oder "sie", abhängig davon, ob ein Mann oder eine Frau ermordet wurde). Man kann bereits zwischen Satzlänge und der Anzahl möglicher Elemente in den einzelnen Positionen eine Relation entdecken, die dem umgekehrten Verhältnis ähnlich ist. Bei Texten ist es noch auffallender. Obwohl wir Synonymität nicht sichern, sondern eher meiden wollten, war es ziemlich schwer, Elemente mit verschiedenen Inhalten in den einzelnen Positionen aufzuzählen; die Vielzahl der Kombinationen ergibt sich hier vor allem aus der Variation synonymer Ausdrücke (lief/rannte, Hund/Köter, eine Bitte abschlagen/nicht erfüllen usw.). Jedoch, wo es möglich war, brachten wir verschiedene, manchmal sogar gegensätzliche Inhalte zusammen (z.B. ich war müde/trotzdem gar nicht müde).

Das Multimedia-Programm TALK (P. Uzonyi - P. Palotai) generiert auch Texte zu Unterrichtszwecken. Im Unterschied zu früheren Generierungsprogrammen stellt es Hörtexte her, deren Lautgestalt nicht vom Computer synthetisiert wird, sondern aus von muttersprachlichen Sprechern aufgenommenen Textsegmenten zusammengefügt wird. Da die Segmentgrenzen mit Satzgrenzen zusammenfallen, klingt der vom Computer zusammengestellte Text auch intonatorisch richtig. Der Zweck der Generierung ist die Überwindung der begrenzten Speicherkapazität der Festplatte, auf der die digitalisierten Tondateien gespeichert werden. Mit diesem Trick kann man anhand einer Aufnahme, die kaum länger als eine halbe Stunde dauert, eine Million Hörtexte zusammenstellen, die im Durchschnitt mindenstens 3 Minuten lang sind, d. h. wenn wir alle Kombinationen der Reihe nach abspielen wollen, würde das etwa 50000 Stunden (mehr als 5 Jahre!) dauern.

Diese immense Quantität kann jedoch verhältnismäßig leicht produziert werden. Man braucht nur 6 mal 10 solche Textteile abzufassen, die in einem Matrix mit 6 Spalten so verteilt werden können, daß jedes Element der ersten Spalte mit jedem der zweiten Spalte fortgesetzt werden kann, was auch für die anderen Spaltenpaare gilt. Die untenstehenden Textteile A1, A2 und A3 sind Elemente aus der ersten Spalte, in der noch weitere 7 Elemente sind; B1, B2 und B3 sind aus der

zweiten, C1, C2 und C3 aus der dritten Spalte (es gibt noch drei Spalten: D, E und F).

A1
An einem Spätnachmittag im Herbst kam Herr Müller früher als gewöhnlich nach Hause. Deshalb beschloß er, sich noch vor dem Abendessen etwas in der Stadt umzusehen.

A2
Der Uhrmachermeister, Herr Schneider, verschloß sorgfältig die verzogene kleine Tür seines Geschäftes in der Birnbaumstraße und ging dann mit schnellen Schritten in Richtung Marktplatz.

A3
Der dänische Physiker, mit dessen Beschattung man mich beauftragt hatte, blieb am Donnerstag zu Hause und verließ erst gegen vier Uhr die Pension.

B1
Über der Stadt hing schon seit Tagen bewegungslos eine graue Wolkendecke. Deshalb dämmerte es auch an diesem Tag früher, und zwar noch bevor der Spitzenverkehr am Nachmittag einsetzte.

B2
In den Straßen der alten Kleinstadt Althausen erschienen schon die Menschen, die nach getaner Arbeit nach Hause eilten.

B3
Es war ein typischer Herbsttag voll von trauriger Stimmung, wenn aus den niedrig hängenden Wolken jeden Augenblick die Regentropfen fallen können.

C1.
Der besagte Mann, der inzwischen das schöne Pflaster der Fußgängerzone beschritt, hielt plötzlich inne und schaute in die Höhe. Aller-

dings konnte man nicht wissen, ob er die gegenüberliegende Fassade oder den Himmel auskundschaftete.

C2.
Unser Freund hat inzwischen die Bachgasse erreicht. Es war ersichtlich, daß er in Gedanken versunken war und sich nicht um die protzigen Schaufenster kümmerte. Nur vor dem Antiquariat blieb er stehen, aber auch hier betrachtete er nicht die Bücher, sondern - es sah wenigstens so aus - die gegenüberliegenden Gebäude.

C3
Die Bäume auf dem langgestreckten Wilhelmsplatz hüteten noch ihre letzten Blätter. Die nackten Baumkronen verbargen nichts mehr vor den oben kreisenden Krähen und vor dem unten vorbeigehenden, in die Höhe schauenden einsamen Mann.

Diese neun Elemente lassen 27 ABC-Kombinationen zu, wie z. B. A1, B1, C1 oder A2, B1, C3 usw. Die ganze Matrix generiert dementsprechend 10^6 ABCDEF-Kombinationen anhand der 60 Elemente. Mit den Texten kann man verschiedene Übungen machen: Diktat, Lückentext, Wahlübung, Frage und Antwort, richtig und falsch. Zu den 60 Textteilen gehören je 3 Varianten von jedem Übungstyp, was die Variabilität des Programmes weiter erhöht. Glossar, Übersetzung, illustrierende Bilder und grammatische Hilfe machen TALK wirklich lernerfreundlich. Mittels des Autorenmoduls kann der Lehrer aus dem Stoff beliebige Kombinationen erstellen und abspeichern, sowie die eigenen Ton-, Bild- und Textdateien eingeben und zu Lernsoftware zusammenfügen.

Sprachspezifische Lernprogramme[7]

Die bisher vorgestellten Programme hatten keinen kontrastiven Charakter, d. h. die Muttersprache des Lerners spielte bei der Zusammenstellung der Übungen keine Rolle. Unterschiede zwischen Sprachen können einerseits systematisch, d.h. mit Regeln beschreibbar sein,

wie z.B. die Abweichung deutscher Genitivkonstruktionen von den ungarischen Äquivalenten (Wortfolge, morphologische Markierung), andererseits können sie individuell, d.h. lexikalischer Natur sein, wie z.B. unterschiedliche Rektionen. Die letztere Art der Abweichungen braucht offenbar mehr Übung und Drill. Zu diesem Zweck, namentlich für das Einüben deutscher verbaler Rektionen, haben wir vor ein paar Jahren das Programm REKTION geschrieben (P. Uzonyi - Zs. Ilosvay). Der potentielle Nutzerkreis, für den das Programm gedacht ist, ist eindeutig ungarisch. Es wurden nämlich solche Verben zusammengestellt, die andere Kasus oder Präpositionen regieren als die häufigsten Übersetzungen der Rektionen ihrer ungarischen Äquivalente (z.B. die ungarische Endung "-ban/ben" wird meistens als "in + D" ins Deutsche übersetzt). Es wurden noch einige Verben mit gleicher Rektion ins Material aufgenommen, da die Lösung sonst zu leicht geworden wäre, denn der Lernende hätte diese Möglichkeit von vornherein ausschließen können. Diese Sätze werden ähnlicherweise generiert wie bei SATZBAU 2.

Der Benutzer kann zwischen vier Stufen wählen, die der Verteilung des Wortschatzes durch die Lehrbücher der 4 Gymnasialklassen entsprechen. Man muß dann auch von drei Übungstypen einen auswählen:

1. Ergänzung, 2. Verbesserung fehlerhafter Sätze, 3. von vier Sätzen ist nur einer richtig, den muß man auswählen. Allerdings ist nicht empfehlenswert, die letzten zwei Typen in der Anfangsphase des Lernens zu benutzen. Das gilt natürlich im allgemeinen für alle Arten von Korrekturübungen, denn man kann sich an die falschen Formen gewöhnen.

Das Programm macht eine Fehleranalyse, außerdem wird dem Lerner auch auf eine andere Weise Hilfe geleistet, indem auf Wunsch die Liste der Verben samt Rektionen angezeigt wird. Die Schüler, die sich prüfen wollen, können auch den Test-Zweig wählen.

Multimedia

Die schriftliche Kommunikation ist zwar eine sehr wichtige Variante des Sprachgebrauchs, aber eine Sprache beherrschen heißt doch vor allem, daß man sich mündlich verständigen kann. Deswegen suchte man auch im CALL schon vom Anfang an nach Möglichkeiten der mündlichen Kommunikation zwischen Lernern und Maschinen.
Wenn man den Rechner mit Tonband oder Videogeräten verbindet, kann der Lernende nicht nur schriftliche, sondern auch akustische und/oder zusätzliche visuelle Stimuli bekommen. Da er mit seinen Entscheidungen den Verlauf des Programmes beeinflussen kann, handelt es sich hier auch um interaktive Software, die man gemeinhin interaktives Audio bzw. interaktives Video nennt.
Die Art der Verbindung ist hardware-abhängig. Das eine Extrem ist, wenn zwischen dem Abspielgerät und dem Computer keine technische Verbindung besteht (d.h. weder ein Kabel noch irgendwelche infrarote Fernsteuerung). An gewissen Punkten des Programmablaufs erscheinen dann einfach Anweisungen an den Benutzer auf dem Bildschirm, z.B. daß er den Recorder stoppen oder zu einer angegebenen Statusnummer spulen soll. Als das andere Extrem könnte etwa bezeichnet werden, wenn auditive und/oder visuelle Effekte ohne externe Geräte, d. h. nur mit Hilfe der Grundausstattung produziert werden. Mit jedem Computer wird eine graphische Karte (z.B. Hercules, CGA, VGA) mitgeliefert, die nicht nur Standbilder, sondern auch Animation präsentieren kann, wobei die nötigen digitalisierten Informationen auf einer Diskette oder Festplatte gespeichert werden. Der eingebaute primitive Lautsynthetisator ist imstande, nicht nur Piepsen oder andere Geräusche, sondern auch menschliche Rede (wenn auch in einer schlechten Qualität) anhand entsprechender Dateien zu reproduzieren. Falls eine bessere Qualität und eine nutzerfreundliche Bedienung der auditiven und/oder visuellen Technologie erzielt werden sollen, muß man zusätzliche Hardware verwenden. Dies kann einerseits eine Schnittstelle (interface) sein, die ermöglicht, daß Kassettenrecorder, Video-

geräte usw. direkt vom Programm gesteuert werden, andererseits Steckkarten wie Soundblaster, Voicecard, Videoblaster usw., mit deren Hilfe Ton- und/oder Videoaufnahmen auf der Festplatte abgespeichert werden können. Zum Abspielen benötigt man beim letzteren nur den Bildschirm des Computers und einen Kopfhörer oder Lautsprecher. Ein großer Vorteil ist, daß die einzelnen Segmente (z.B. Sätze) als Files gespeichert werden und als solche jederzeit erreichbar sind, so daß eine mündliche Interaktion leicht simuliert werden kann (z.B. bei einer falschen Lösung sagt der Computer mit tadelnder Intonation, daß es falsch war, und bei der richtigen Antwort kann er den Lerner mündlich loben, oder wenn die Videobilder vom Lehrer auch auf der Festplatte sind, kann der Lerner bei falscher Antwort nicht nur eine tadelnde Stimme, sondern auch den tadelnden Gesichtsausdruck und die Geste eines Schulmeisters erleben). Ein Nachteil ist allerdings, daß auf einer 20 MByte Festplatte kaum mehr als eine halbe Stunde digitalisierte Rede gespeichert werden kann, von digitalisierten Videobildern ganz zu schweigen.
Um unter anderem auch dieses Problem zu lösen, haben wir vor kurzem das o. a. Programm mit dem Namen TALK entwickelt, das aus isoliert aufgenommenen Sätzen 1000000 Hörtexte kombinieren kann.
Eine andere Lösung des Speicherproblems kann sein, Ton und/oder Bild auf einer CD-ROM zu speichern, deren Kapazität größer ist als die der geläufigen Festplatten und die ebenso leicht austauschbar ist wie eine Diskette. Ein Nachteil ist dabei, daß die Herstellung von CD-Aufnahmen zur Zeit noch sehr kostspielig ist.

Autorenprogramme

Der Lehrer kann auch selber Lernsoftware produzieren und dadurch erreichen, daß die Programme dem jeweiligen Grad der Sprachkenntnisse der Schüler, dem aktuellen Lehrstoff usw. angepaßt werden. Ein Sprachlehrer ist natürlich kein Programmierer, so daß man von ihm nicht erwarten kann, daß er Programme schreibt. Für diesen Zweck hat

man Autorenprogramme entwickelt. Das sind vorgefertigte Programmrahmen, die mit den verschiedensten Lehrstoffen gefüllt werden können, und zwar ohne die geringsten Programmierkenntnisse. Wenn ein Autorenprogramm nutzerfreundlich genug ist, kann man mit ihm leichter umgehen als mit einem Textverarbeitungsprogramm. Im Autorenprogramm ist bis auf die Übungsinhalte alles vorprogrammiert, so z.B. das Format der Lernereingaben, das Verfahren der Auswertung, ein Rahmen für Rückmeldungen und Kommentare usw.

Das Autorenprogramm ist für zwei Typen von Anwendern gedacht: für den Lehrer, der die Übungen erstellt, und natürlich für den Lerner, der dann mit diesen Übungen arbeitet. Manchmal sind es zwei verschiedene Programme, oder es gibt zwei Versionen: eine, die sowohl Autorenteil als auch Übungsteil enthält, und eine ohne Autorenteil, also für den Lerner (z.B. ECLIPSE). Es kommt auch vor, daß das Programm eigentlich nur den Lernerteil enthält, und die Übungen muß man mit Hilfe eines beliebigen Texverarbeitungsprogrammes schreiben. Hauptsache, daß es ein Textfile mit ASCII-Code ist, und sowohl seine speziellen Symbole als auch die Struktur dem Format entsprechen, das durch das Rahmenprogramm interpretierbar ist (z.B. die CALIS-Script im Computer Assisted Language Instruction System der Duke University - siehe Anhang 8). Das Symbolinventar und die Syntax dieser "Autorensprache" muß man sich aneignen, was nicht zu schwierig ist, zumal Begleithefte mit der Beschreibung der Sprache zum Nachschlagen zur Verfügung stehen.

Es gibt verschiedene Autorenprogramme, je nachdem, wieviel und welche Übungstypen angeboten werden, wie komplex die möglichen Algorithmen sind (z.B. Randomisierung oder Verzweigungen realisierbar sind), oder ob Fehlerbehandlung, Hilfe, Glossar usw. erstellbar sind (letztere können z.T. auch vorprogrammiert sein).

Der häufigste Übungstyp ist die Ausfüllung von Lücken. Es gibt hierbei auch verschiedene Varianten. Es ist für den Lehrer vielleicht die einfachste Variante (die allerdings von geringstem Nutzen ist), wenn das Programm bestimmt, welche Wörter gelöscht werden. Der Lehrer gibt nur eine Zahl an, die den Abstand zwischen den Lücken festlegt (z.B. jedes n-te Wort) oder die Zahl der Wörter bestimmt, die vom Programm zufällig ausgewählt und weggelassen werden sollen. Lezteres läßt sich gezielter anwenden, wenn man die Vokabeln irgendwie markieren kann, die überhaupt vom Programm gelöscht werden können (z.B. Lückenspiel - P. Uzonyi und Z. Papp).
Ebenso leicht ist die Aufbereitung von solchen Lückentexten, in denen wortartspezifische Lücken mit einem Tastendruck erzeugt werden können. Dies bezieht sich aber nur auf die wenigen Wortarten, in die nur kleinere Gruppen von Wörtern gehören, wie z.B. Präpositionen, Artikel, Partikeln usw. (z.B. Eclipse). Das Programm muß die Wörter der Texte nur mit diesen kurzen Listen vergleichen. Wegen der Homonymie kann es aber zu Fehlinterpretationen kommen (Z.B. das Präfix *ab* wird ausgeblendet, wenn die Präpositionen gelöscht werden sollten, weil die Form in der Liste der Präpositionen auch enthalten ist.).
Eine fehlerlose Wortartspezifizierung kann freilich auch erzielt werden, indem man jedes Wort des Textes mit dem Kode seiner Wortart versieht, aber diese Möglichkeit muß auch vorprogrammiert werden (siehe das Programm "Arbeit mit Texten" von T. Hassert).
Eine recht schwierige, aber meistens auch unterhaltsame Aufgabe ist es für den Lernenden, wenn alle Wörter eines Textes ausgeblendet werden. So einen Text kann man nur in dem Falle erschließen, wenn man vorher den Text lesen kann oder wenn eine Hilfe jederzeit erbeten werden kann. Bei der Entschlüsselung des Textes wird eine Reihe von Fähigkeiten auf die Probe gestellt, u. a. die grammatische und die lexikalische Kompetenz, Kenntnis der Textverflechtungsmerkmale usw.
Die intelligentesten Lückenübungen kann man durch die Bestimmung jeder einzelnen Lücke erstellen. In diesem Fall ist es nämlich möglich, daß die Lücken den eigenen Bedürfnissen angepaßt werden (z.B. wenn sie nicht in Texte sondern in isolierte Sätze gesetzt werden)

und - wenn es vom Rahmenprogramm zugelassen ist - jede Lücke mit den entsprechenden parallelen richtigen und den antizipierten falschen Lösungen versehen wird. Außerdem kann jede Lösung einen individuellen Kommentar erhalten (siehe z.B. CALIS Anhang 8).
Es sind natürlich nicht alle falschen Antworten vorhersagbar. Man kann sich z.B. vertippen, und es ist irritierend, wenn ein längerer Satz wegen eines Tippfehlers im ganzen als falsch bewertet wird. Deswegen hat man in eine neuere Version von CALIS einen Teil eingefügt, der den Namen "Spellcheck" hat, und nicht antizipierte falsche Lösungen mit der gespeicherten richtigen Lösung vergleicht und vertippte, fehlende oder überflüssige Buchstaben im Kontext anzeigt. Außerdem kann man in CALIS mittels einer Anweisung erlauben, daß der Unterschied zwischen Klein- und Großschreibung ignoriert wird oder das Fehlen von Interpunktionszeichen auch toleriert wird.
Diese Übungen sind vielseitig, lehrhaft, können in der Einzelarbeit effektiv gebraucht werden, aber ihre Erstellung ist eine ziemlich zeitaufwendige und bei weitem nicht mechanische Arbeit.
Auch Multimedia-Programme können als Autorenprogramme konzipiert werden. Dabei müssen die Programmierer dafür sorgen, daß der Lehrer auf die einfachste Weise Ton- und Bilddateien herstellen und diese mit Texten verknüpfen kann (siehe z.B. TALK).

4.
Computerisierte Wörterbücher

Computerwörterbücher dienen vor allem zum Zwecke der menschlichen Übersetzung. Es gibt auch solche sprachlichen Datenbanken, die andere Funktionen haben, wie z.B. Überprüfung der Rechtschreibung, maschinelle Übersetzung usw.

Unter menschlicher Übersetzung soll man einfach die herkömmliche Übersetzungsarbeit verstehen, die nicht vom Computer, sondern vom Menschen verrichtet wird. Dabei können dem Übersetzenden - sei er Sprachlernender oder professioneller Übersetzer - computerisierte Wörterbücher in nicht geringem Maße behilflich sein.

Was kann aber ein Computerwörterbuch (im weiteren: CW) leisten, was einem traditionellen Wörterbuch nicht zumutbar ist? Daß der Übersetzer nicht in verschiedenen Bänden minutenlang herumzublättern braucht, ist nur eine der vielen Dienstleistungen eines CW; sogar das primitivste Wörterbuchprogramm kann die Übersetzung samt den zusätzlichen Informationen im Nu auf dem Bildschirm anzeigen, wenn man ein Wort eintippt. Es ist übrigens ein jedes Datenbasisverwaltungsprogramm imstande, auch derartige sprachliche Daten zu speichern und zu finden. Für einen Computer ist es nämlich einerlei, ob er Namen von Waren speichert, die man auf Lager hat, mit Stückzahl und Preis, oder aber Wörter der deutschen Sprache mit ihren ungarischen Übersetzungen.

Ein echtes CW kann aber auch solche Operationen durchführen, zu denen ein herkömmliches Datenbasisprogramm nicht fähig ist. So eine Operation ist beispielsweise die Zurückführung einer flektierten Form auf die konventionelle Form, die das entsprechende Lexem repräsentiert (wie z.B. Infinitiv oder Nominativ Singular). Ein guter Übersetzer tut das freilich auch selber, d.h. bei der Abfrage tippt er nicht die im Text stehende Form, sondern die Grundform ein, die allerdings bei völlig unbekannten Wörtern oft nur eine hypothetische Form sein kann. Die Eingabe kann jedoch auch anders vor sich gehen, wobei die Lemmatisierung (d.h. die Zurückführung auf die Grundform) nicht überflüssig ist.

Die neueren CW lassen sich nämlich mit Textverarbeitungsprogrammen verknüpfen. Hierbei kann man die abzufragenden Vokabeln im Text mit

Cursor oder Maus direkt auswählen, und die ausgewählten Wörter sind nicht selten flektierte Formen.
Die übrigen vorteilhaften Eigenschaften der CW sind für Datenbasisverwaltungsprogramme im allgemeinen charakteristisch. So z.B. die Möglichkeit der Erweiterung und Korrektur in jedem Moment, was bei einem "Papierwörterbuch" nicht realisierbar ist.
Falls der Übersetzer einen Fehler im CW entdeckt oder eine zusätzliche Bedeutung für ein vorhandenes Wort findet, kann er die Daten modifizieren, aber er kann auch neue Wörter eintragen, die im CW fehlen.
Datenbasisverwaltungsprogramme, wie auch die meisten CW, können nicht nur nach Stichwörtern suchen. Die Suche kann sich auch auf die anderen Felder des Rekords erstrecken (Der Rekord ist eine Einheit der Datenbasis, z.B. ein Wort mit seinen Äquivalenten in der Zielsprache, und die Felder sind Teile des Rekords, die vom Programm vereinzelt behandelt werden.). Dies funktioniert ähnlicherweise wie der Search-Befehl der geläufigen Textverarbeitungsprogramme.
Eine intelligentere Suche lassen bei weitem nicht alle Programme zu. Dies kann sich z. B. auf das gleichzeitige Vorhandensein von mehreren Bedingungen beziehen. Zwischen den Bedingungen kann hierbei ein kopulatives ("und") oder ein disjunktives ("oder") Verhältnis bestehen. Man kann beispielsweise die Einträge eines deutsch-ungarischen CW suchen, in denen am Wortanfang auf der deutschen Seite "auf", auf der ungarischen "fel" steht (z.B. aufstehen - felkel). Wenn wir noch einen logischen Operator, die Negation gebrauchen, können wir noch komplexere Aufgaben formulieren. Man kann z. B. diejenigen Einträge suchen lassen, in denen parallel zu "fel" kein "auf" steht (z. B. felfedez - entdecken).
Ein erweiterbares CW muß die eingegebenen Wörter in alphabetische Reihenfolge ordnen. Der Algorithmus der Ordnung ist vom Alphabet der betreffenden Sprache abhängig. Im deutschen Alphabet gibt es z.B. mehr Buchstaben als im englischen, vom ungarischen ganz zu schweigen. Daß dabei einige Unterschiede traditionell ignoriert werden, macht die Sache noch komplizierter (z. B. Umlaut im Deutschen, Länge im

Ungarischen - aber o und ö gelten beim letzteren als verschieden). Wegen der zweigliedrigen Buchstaben des ungarischen Alphabets muß der Algorithmus extra Prozeduren vorsehen, z. B. bei einem *n*: wenn ihm ein *y* folgt, kommt diese Kombination nach allen *n*, weil der Buchstabe *ny* im Alphabet nach dem *n* steht. Da aber ein einfacher Algorithmus die Morphemgrenzen nicht detektieren kann, können einige Wörter doch falsch eingeordnet werden. Z.B. *házsor* kommt nach *hazug*, obwohl es hier kein *zs* ist: *z* und *s* gehören verschiedenen Morphemen an. Ebenso wie bei der automatischen Silbentrennung bei Textverarbeitungen, hilft hier auch ein umfangreiches Wortstammverzeichnis, das allein immerhin noch nicht ausreicht: man braucht noch ein langes Inventar der Einzelfälle. So ein Fall is z. B. *haltenden*, das nichts mit *enden* zu tun hat (wie etwa *voll-enden*) und folglich als *hal-tenden* zu trennen ist. In einigen Textverarbeitungsprogrammen, die keine automatische Silbentrennung machen, kann man sogenannte virtuelle Bindestriche verwenden. Diese werden zu realen Bindestrichen nur in dem Fall, wenn sie am Ende einer Zeile sind. Die alphabetische Ordnung kann auch vervollkommnet werden, indem man virtuelle Symbole einfügt, die dann auf dem Bildschirm oder im Druck nicht erscheinen (z.B. *ház*sor*).

Die größten Datenbanken sind natürlich nicht für die Arbeitssituation "ein Mensch - eine Maschine" konzipiert worden. Dank der Datenfernübertragung können gleichzeitig viele Übersetzer, die an verschiedenen Orten vor ihren Terminalen sitzen, ein zentrales Computerwörterbuch benutzen. In diesem Fall kann der Benutzer meistens nicht direkt Veränderungen an den zentralen Daten vornehmen, aber er kann seine Vorschläge dem Team mitteilen, dessen Aufgabe die Korrektur und Erweiterung der Datenbasis ist.

5.
Maschinelle Übersetzung

Die Datenbasis in den maschinellen Übersetzungssystemen

In jedem Übersetzungssystem, auf welchem Prinzip es auch immer aufgebaut ist, kann man zwei Teile unbedingt finden: eine Analyse und eine Synthese. Analyse läßt sich viel schwieriger automatisieren als Synthese, da die Maschine dabei letzten Endes einen Text, der von Menschen an andere Menschen geschrieben worden ist, bis zu einem gewissen Grade verstehen muß. Solche Texte sind indessen voll von mehrdeutigen Ausdrücken, und der Computer muß jeweils die einzige Bedeutung auswählen, die zum Kontext, Stil, Thema usw. paßt. Bei der Synthese geht er jedoch von einem festgelegten Inhalt aus, zu dem er nur noch die entsprechenden Ausdrucksformen finden muß, was viel leichter ist, als Ausdrücke zu verstehen (beim Menschen ist es in der Regel umgekehrt).

Bei der Übersetzung stützt sich das System auf ein internes Lexikon, auf das eigene CW. Es ist jedenfalls anders aufgebaut als die für menschlichen Gebrauch gedachten Computerwörterbücher. Das Lexikon zur maschinellen Übersetzung (im weiteren: LMÜ) muß ermöglichen, daß das Programm die lexikalischen Einheiten des Quelltextes identifiziert, die syntaktische Rolle dieser Einheiten und die Satzstrukturen erschließt, dann die entsprechenden Strukturen und Lexeme der Zielsprache findet und zum Schluß die morphologisch richtigen Formen herstellt. Dazu braucht man morphologische und syntaktische Regeln, die zum Teil mit diesen Informationen operieren.

Die wichtigsten Charakteristika der LMÜ der verschiedenen Systeme weisen keine wesentlichen Unterschiede auf. Zur morphologischen Analyse ist z.B. ein Morphinventar nötig, d.h. eine vollständige Liste der Varianten aller Stämme und Affixe (z.B. *brech*, *brich*, *brach*, *bräch*, *broch*; *st*, *t*, *est*). Wenn einem Morph der Quellsprache immer ein und dasselbe Morph der Zielsprache entsprechen würde, wäre die automatische Übersetzung ziemlich einfach zu lösen. Der Weg zum Äquivalent in der anderen Sprache ist demgegenüber voll von Verzweigungen, an denen jeweils die eine oder die andere Richtung gewählt werden muß.

Falls die Form eines Morphs mit Hilfe des Inventars bereits identifiziert worden ist, aber diese Form die Realisierung mehrerer Morpheme sein kann, gelangt der Algorithmus gleich zu einer Verzweigung. So ein Morph ist beispielsweise *reich*. Das Programm muß aufgrund entsprechender Informationen entscheiden, ob es in die Richtung eines verbalen oder eines nominalen Lexems weitergehen soll. Die Lösung liegt scheinbar auf der Hand: wenn rechts von ihm eine verbale Endung steht, dann ist es ein Verbalstamm, mit einer adjektivischen Endung dagegen ist es ein Adjektiv. Das Problem dabei ist nur, daß manche Formen sowohl als verbale als auch als adjektivische Affixe auftreten können. So z. B. *e* oder *en*: *ich reiche etw., die reiche Frau* usw. Die Überprüfung der Verträglichkeit der Endung mit dem Flexionstyp des Stammes kann in einigen Fällen auch helfen. Z. B. *s* ist eine substantivische Endung, die aber nicht nach jedem Substantiv stehen kann; das Morph *Reich* ist der Stamm zweier Substantive, die aber verschiedenen Deklinationstypen angehören, so daß einige Endungen (u.a. *s*) bei der Entscheidung des Automaten ausschlaggebend sind ((*des*) *Reichs* und (*des*) *Reichen*). Die Endung *en* kann dagegen nichts entscheiden (sie reichen, die reichen Menschen, einen Reichen, in den Reichen). In diesem Fall hilft meistens die syntaktische oder gegebenenfalls semantische Untersuchung des Kontextes.

Nach der Identifizierung des Wortes muß man auch seine syntaktische Rolle bestimmen. Auf dieser Ebene der Analyse gelten Formen mit verschiedenen Bedeutungen als Repräsentanten eines Lexems, wenn sich die verschiedenen Bedeutungen an dieselbe Wortart und an denselben Flexionstyp knüpfen (z. B. *haben* als Vollverb und als Hilfsverb). Unter syntaktischer Rolle soll man dabei nicht nur die Regens- bzw. Dependensrolle verstehen, sondern auch die Valenzrolle, also die Einordnung in die Gruppe der vom Prädikat bestimmten Mitspieler (Aktanten), bzw. in die der freien Angaben. Diese Unterscheidung ist nicht nur für dependenzgrammatisch ausgerichtete Übersetzungsmodelle charakteristisch, denn - möglicherweise mit einer anderen Terminologie - alle Systeme müssen die vom Prädikat bestimmten formalen Merkmale und die durch andere Faktoren bestimmten Formen unterschiedlich behan-

deln, sonst können die Konstrukte falsch übersetzt werden. Dazu muß man im LMÜ bei jedem Lexem seine möglichen Valenzstrukturen (Satzbaupläne) angeben.
Die Wörter haben oft mehrere potentielle Valenzstrukturen, was die Analyse einerseits erschweren, andererseits manchmal erleichtern kann. Die Analyse wird dadurch erschwert, daß viele Mitspieler fakultativ sind, und daher ist es mitunter problematisch zu entscheiden, welche der potentiellen Valenzstrukturen im Satz vorliegt und ob eine nach fakultativem Valenzmitspieler aussehende Form keine freie Angabe ist.

(1) Er arbeitete am Schreibtisch.
(2) Er arbeitete am Wasser.
(3) Er arbeitete an der Dissertation.

Das Verb *arbeiten* bestimmt beispielsweise 2 Formen im Satz: einen Nominativ und die Präposition *an* mit dem Dativ. Da der Präpositionalkasus hierbei weglaßbar ist, wird der Satz (1) zweideutig: der Schreibtisch kann sowohl Objekt als auch Ort der Arbeit sein. Weder ein Mensch noch eine Maschine können ohne Kontext entscheiden, ob die Übersetzung ins Ungarische "íróasztalon" oder "íróasztalnál" sein soll. Die Sätze (2) und (3) bereiten dagegen dem Übersetzer kein langes Kopfzerbrechen, weil ein Mensch weiß, daß *Wasser* in der Regel kein Arbeitsprodukt und *Dissertation* kein Arbeitsplatz sind. Was kann aber der Computer mit diesen Sätzen anfangen? Das hängt auch vor allem vom verwendeten LMÜ ab, in dem auch außersprachliche Kenntnisse gespeichert werden können, und zwar als semantische Merkmale der Wörter (z.B. Wasser: 'Stoff', 'flüssig', 'Ort' usw.).
Die verschiedenen Valenzstrukturen eines Wortes können die Analyse gegebenenfalls auch erleichtern, indem sie die passende Bedeutung auswählen helfen.

(1) Die Partei besteht seit fünf Jahren.
(2) Die Kundin besteht auf ihrer Bitte.

(3) Das Problem besteht in diesem Widerspruch.
(4) Die Dissertation besteht aus vier Kapiteln.
(5) Der Student besteht die Prüfung.

Die Sätze (1), (4) und (5) sind eindeutig wegen der Form der Aktanten, d.h. hier kann die identifizierte Valenz helfen. Ohne semantische Untersuchung der Aktanten sind jedoch die Sätze (2) und (3) nicht eindeutig, weil *bestehen* in der Bedeutung 'vorhanden sein' mit vielen lokalen Präpositionen stehen kann, wie z.B. im Satz (6).

(6). Die stärkste Résistance bestand *im* Raumgebiet/*auf* dem Territorium Frankreichs.

Hierbei können wieder nur die semantischen Merkmale des engeren oder weiteren Kontextes behilflich sein.
Die Angabe der Valenz ist natürlich nicht nur wegen der Lösung einiger Mehrdeutigkeitsprobleme wichtig, man muß nämlich die Valenzstruktur ebenso übersetzen wie die in die Leerstellen eingesetzten Lexeme. Die Übersetzung einer Valenzstruktur ist eine zu ihr parallele Valenzstruktur der Zielsprache, wie z. B. in (7).

(7) deutsch: *brauchen* + Nom + Akk
ungarisch: *szüksége van* + NAK + RA

Ein LMÜ muß auch derartigen Äquivalenzen gerecht werden.
Von einem Wort der Quellsprache führt ein sich mehrfach verzweigender Weg zu den Wörtern der Zielsprache, die als Übersetzungen in Frage kommen. Die Wegweiser bei den Verzweigungen sind die im Lexikon gespeicherten Valenzstrukturen, semantische Merkmale, außerdem stehende Wortverbindungen (z. B. *in Führung gehen - átveszi a vezetést* (etwa: "die Führung übernehmen") oder *vezetéshez jut*("zur Führung kommen")). Falls am Ende eines Weges immer noch mehrere Formen stehen, muß die Umgebung in der Zielsprache bei der Wahl helfen, wie z. B. das Genus des Substantivs, das im Deutschen einen der drei Artikel wählt, wenn ein determiniertes Substantiv aus dem Ungarischen übersetzt werden soll.

Ein Lexikon zur maschinellen Übersetzung muß folglich auch die Informationen enthalten, die von der Quellsprache nicht abhängen, aber ohne die eine syntaktisch richtige Verbindung der Elemente unmöglich wäre (im Deutschen z. B. die Genera, die unregelmäßigen Verbstämme, die adjektivischen Steigerungsformen mit Umlaut, die Verben, die nur samt einem Korrelat einen Nebensatz regieren können, usw.).

Automatische und interaktive Übersetzung mit Hilfe des Computers[8]

Die computerisierten Wörterbücher fanden schnell Eingang in den Alltag der professionellen Übersetzung. Es gab nur eine recht geringe Anzahl von Firmen, die auf dem ursprünglichen Ideal der völlig automatischen Übersetzung bestanden, und einige von ihnen haben funktionierende Systeme entwickelt, die eine Prä- und/oder Postedition benötigen und trotzdem nicht teurer sind als die völlig menschliche Übersetzung.

Zwischen dem niedrigsten und dem höchsten Grad der maschinellen Aktivität, d.h. zwischen Datenbanken und automatischen Übersetzungssystemen, sind unendlich viele Übergangsgrade vorstellbar, von denen manche auch realisiert worden sind.

Die Rechentechnik entwickelte sich in letzter Zeit mit einem beschleunigten Tempo, demzufolge müssen die Linguisten die skeptischen Meinungen der fünfziger - sechziger Jahre auch neu überlegen. Heute weisen Operationsgeschwindigkeit und Speicherkapazität eine derart verheißungsvolle Tendenz auf, daß wir uns bald auf keine technischen Hindernisse mehr berufen können. Die Fachleute, die sich mit maschineller Übersetzung, bzw. mit der künstlichen Intelligenz befassen, müssen möglicherweise mit einer Herausforderung der Technik fertigwerden: es kann nämlich passieren, daß die technologischen Voraussetzungen zum Funktionieren einer der natürlichen ähnlichen künstlichen Intelligenz eher zur Verfügung stehen werden als ein funk-

tionsfähiges Modell der Intelligenz. Wenn es doch nicht dazu kommen wird, ist das der Tatsache zu verdanken, daß es bei der Erarbeitung der neueren Computergenerationen immer schwieriger wird, Forschungen der Mikroelektronik und die der künstlichen Intelligenz voneinander zu trennen.

Maschinelle Übersetzung ist freilich etwas mehr und zugleich etwas weniger als das, was man herkömmlich als künstliche Intelligenz bezeichnet. Sie ist mehr, denn sie benötigt eine spezielle natürlichsprachliche Schnittstelle, die zwei Sprachen beherrscht: aus der einen kann sie Informationen gewinnen, und denselben Inhalt kann sie in der anderen Sprache wiedergeben. Sie ist auch weniger als die künstliche Intelligenz im weiteren Sinne, denn logische Operationen, Folgerungen führt sie nur in dem Maße durch, das zu einer adäquaten Übersetzung erforderlich ist.

Während das Übersetzungsprogramm die Input-Stringkette in eine Output-Stringkette verwandelt, muß es eine Reihe algorithmischer Schritte zurücklegen, die z.T. Verzweigungen sind. Die Wahl ist meistens nicht frei, sondern bei einer Verzweigung ist nur eine Richtung möglich, wenn die optimale Übersetzung erreicht werden soll (die Optionen sind nur dann frei, wenn es nach Meinung der Autoren mehrere optimale Lösungen gibt). Die Auswahl einer nicht freien Richtung ist jeweils eine Folgerung, zu deren Prämissen das Programm grundsätzlich auf zweifache Weise kommen kann: entweder durch einen unmittelbaren Eingriff der natürlichen Intelligenz (sprich: des Menschen) oder mit der Benützung des Kontextes und einer Datenbasis von außersprachlichen Kenntnissen.

Diese zwei Verfahren schließen einander nicht aus, also lassen sich kombinieren. Die Stufen der Automatisierung der Übersetzung entsprechen letzten Endes den verschiedenen Verteilungsmöglichkeiten der beiden Informationsquellen.

Auf der untersten Stufe der Automatisierung sind die maschinellen Wörterbücher, die gewisse Verzweigungen des Übersetzungsalgorithmus in der Form von Äquivalentenlisten darstellen und möglicherweise auch einige Voraussetzungen zur Wahl der einzelnen Äquivalente expli-

zieren, es ist jedoch der Übersetzer, der das Vorhandensein der Voraussetzungen im betreffenden Text überprüfen muß. Bei einer um eine Stufe höheren Automatisierung ist es möglich, Wörter auch aufgrund ihrer flektierten Formen zu finden, denn das Programm führt diese auf ein Lexemsymbol zurück. Verzweigungen kann man hierbei grundsätzlich auf zweifache Weise behandeln: entweder werden die möglichen Lösungen dem Menschen in einer Menüliste angeboten (z.B. meine: 1. meinen (Verb); 2. mein (Pronomen)), oder die Maschine benutzt ihre eigene "Intelligenz", was im Falle eines computerisierten Wörterbuches wohl als Luxus gilt; z.B. im Satz (1) kann die Form "Bandes" entweder als das Lexem "der Band" oder aber - woran der Mensch normalerweise nicht denken würde - als "das Band" identifiziert werden.

(1) Am Ende des dritten Bandes befindet sich ein Literaturverzeichnis.

Ein Computer muß die letztere Version anhand einer komplizierten Kontextuntersuchung ausschließen, was ohne außersprachliche Kenntnisse gemeinhin nicht zu bewerkstelligen ist (Z.B. wo sich ein Literaturverzeichnis überhaupt befinden kann: es kann das Buch, die Seite, der Band sein - oder vielleicht auch das Magnetband?).

Die Übersetzung mit Hilfe eines maschinellen Wörterbuchs kann als rechnergestützte menschliche Übersetzung bezeichnet werden. Die Ausgabe ist dabei nämlich so weit vom Format einer fertigen Übersetzung, daß die Arbeit an diesem Output keinesfalls "Posteditieren" zu nennen ist. Die allgemein verbreitete Technik, die freilich immer unter dem Niveau der neuesten Spitzentechnologie liegt, ermöglicht schon jetzt mehr als die Speicherung von lexikalen Daten.

Auf PC-s, die nunmehr zu unserem Alltag gehören, können auch solche Programme laufen, die aus einem Input-Text einen Output-Text herstellen, d.h. Programme, die natürlichsprachliche Texte analysieren und synthetisieren. Der Mensch kann dabei vor, nach und/oder während der Übersetzung eingreifen.

Als Präeditieren kann man zweierlei bezeichnen: sowohl geringfügige Modifizierungen am Quellentext als auch seine Umformulierung in eine quasi-natürliche Sprache, die eigentlich als eine eigenartige höhere

Programmiersprache angesehen werden kann.
Posteditieren kann sowohl kleinere stilistische Korrekturen als auch die "Umkodierung" einer für Laien unverständlichen Rohübersetzung in einen gemeinverständlichen Text bedeuten.
Den Typus der MÜ, bei dem der Mensch während des Programmablaufs (auch) eingreifen muß, nennt man gemeinhin interaktive maschinelle Übersetzung (IMÜ). Sie hat mehrere Untertypen, von denen ich nun einen kurz vorstellen möchte. An der Kossuth-Lajos-Universität (Debrecen) hat man vor ein paar Jahren mit der Entwicklung einiger experimenteller Module eines Systems begonnen, das dem Nutzer in der Ausgangssprache Fragen stellt, wenn es zu Verzweigungen kommt, wo eine automatische Fortsetzung nicht möglich ist (siehe auch bei Dragalin-Hunyadi-Uzonyi [4]). Im Idealfall wird es also zu einer Software, mit welcher der Nutzer aus seiner Muttersprache in eine für ihn unbekannte Sprache übersetzen kann, u.z. ohne Prä- bzw. Posteditieren im herkömmlichen Sinne (ähnlicherweise wie bei Wood&Chandler, [51]). Das Lexikon dieses Übersetzungsprogrammes ist in nutzerfreundlicher Weise zu erweitern, dabei reicht es allerdings nicht immer aus, wenn man nur die Quellsprache beherrscht.
Bei der IMÜ kann vorkommen, daß sie wegen der zu vielen Fragen umständlicher und langsamer vor sich geht als die traditionelle Übersetzung. Dies kann auf verschiedene Weise behoben werden. Man kann sich beispielsweise eine eindeutige Formulierungsart aneignen, d.h. die Ersetzung von mehrdeutigen Wörtern und Strukturen durch eindeutige Ausdrücke, die das Programm jeweils empfiehlt. Das ist letzten Endes die Kombinierung der Präeditierung mit der Interaktivität (es ist dabei natürlich günstiger, wenn der Ausgangstext bereits in der ersten Fassung vom Nutzer formuliert wird, also keine Umformulierung nötig ist).
Eine Reduzierung der Zahl der Fragen kann auch erreicht werden, wenn das Programm den Kontext untersucht. Da sich Fragestellung und Kontextuntersuchung nicht ausschließen, sind sie miteinander kombinierbar. Die zwei Extreme sind hierbei, alles erfragen und nichts erfragen. Letzteres ist mit der automatischen Übersetzung identisch und

als idealisierter Endpunkt einer Entwicklungsstrategie vorstellbar. So eine Strategie ermöglicht, daß bereits in einer frühen Etappe ein funktionsfähiges Übersetzungssystem zustande kommt, in dem man die interaktiven Teile Schritt für Schritt, durch Erweiterung der sprachlichen und außersprachlichen Wissensbasis (und parallel dazu der Hardwarekapazität) mit automatischen Prozeduren austauscht.

Einige Verzweigungen werden vom Programm automatisch behandelt. Ein Teil der Entscheidungen zwischen Morpheminhalten ist z.B. anhand morphotaktischer Regeln zu treffen. Diese Phänomene kann man an deutschen Beispielen ebensogut illustrieren wie an ungarischen:

(2) leer/t - Verbalstamm
leer/es - Adjektivstamm
lehr/en - Verbalsuffix

Was bei der morphologischen Analyse nicht entscheidbar ist, wird auf die Ebene der Syntax gebracht, wo die syntaktische Struktur weitere kontextuelle Informationen liefern kann:

(3) Sie leeren ihre Gläser.
Sie nehmen die leeren Gläser.

Gewisse morphologische Mehrdeutigkeiten lassen sich nicht einmal im Laufe der syntaktischen Analyse aufheben (z.B. bei der o.a. Form "Bandes"). In diesem Fall folgt eine Frage, ebenso wie bei mehrfachen Interpretierbarkeiten von syntaktischen Strukturen wie (4).

(4) der Wagen des Mannes, der vor dem Haus steht

Wenn man (4) z.B. ins Ungarische übersetzt, muß entschieden werden, ob das Bezugswort *Wagen* oder *Mann* ist, denn es gibt unterschiedliche Relativpronomina für Personen und Nicht-Personen.
Also eine Maschine, die ohne menschliche Hilfe, aber schneller,

billiger und nicht schlechter als der Mensch übersetzen kann, scheint Fachleuten unserer Tage wieder realisierbar, während sie unlängst noch als eine Art Perpetuum Mobile angesehen wurde. Bis zur Realisierung jedoch - sollte das Tempo der Entwicklung noch so beschleunigt werden - muß man lange Jahre warten. Die allmähliche Umschaltung von Interaktivität auf automatische Betriebsart ermöglicht, daß bei mangelnder menschlicher Übersetzungskapazität der Rechner mit menschlicher Unterstützung schon jetzt übersetzt, wobei einzuräumen ist, daß es anfangs wesentlich schlechter und langsamer funktioniert als das angestrebte Ideal.

6.
Linguistische Formalismen in der maschinellen Sprachverarbeitung

Die strengen, nach mathematischer Exaktheit strebenden Methoden der strukturalistischen Schulen, insbesondere die der amerikanischen deskriptiven Linguistik, schufen günstige Bedingungen zur Formalisierung natürlichsprachlicher Gegebenheiten, mit denen ein Computer andernfalls nichts hätte anfangen können.

Computerlinguisten waren dann auch von Chomskys generativer Transformationsgrammatik ([5, 6]) begeistert, aber die Hoffnungen auf eine elegante Lösung der MÜ mittels Transformationen haben sich bald zerschlagen. Die ursprüngliche Theorie wurde vom Autor mehrmals umgearbeitet, bis ein Formalismus entstand, der kaum mehr an die erste Version erinnerte. Dies war die Rektions- und Bindungstheorie ([7]), die 1981 erschien. Es gibt zwar einige Parser (Programme für automatische syntaktische Analyse), die auf dieser theoretischen Basis aufgebaut sind (z.B. [49, 8]), aber in der Computerlinguistik konnte die Transformationsgrammatik nicht Fuß fassen. Das liegt z.T. daran, daß bei Chomsky weder Form noch Inhalt als Ausgangspunkt gelten, sondern Satzstrukturen, die dazwischen sind und semantisch bzw. phonologisch interpretiert werden sollen. Dieses System entspricht weder den psychischen noch den maschinellen Prozessen der Analyse und Synthese von Sätzen. Ein anderer Grund für nur spärliche Implementationen dürfte sein, daß die Transformationen die Verarbeitung mitunter ziemlich umständlich machen können. Deswegen konnte nicht einmal die sog. generative Semantik ([23]) in die Computerlinguistik Eingang finden, obgleich sie mit den Transformationen jeweils vom Inhalt zur Form gelangte.

In den sechziger Jahren veröffentlichte S. Lamb die Grundzüge einer sog. stratifikationellen Grammatik ([24]), die - ebenso wie die generative Semantik - die Form mit dem Inhalt verbindet, aber nicht nur in einer, sondern in beiden Richtungen funktioniert. Zwischen den phonetischen Repräsentationen und den begrifflichen Vorstellungen liegen mehrere Schichten (strata), und die Grammatik beschreibt die Verbindung der Elemente der einzelnen Schichten mit denen der benachbarten Schichten. Dieses System kann auch den Kommunikationsprozeß teilweise modellieren, weil ein Impuls vom beliebigen aktivierten

Element das System von Schicht zu Schicht durchlaufen kann. Trotz dieser auffallenden Ähnlichkeit mit elektronischen Mechanismen machte der stratifikationelle Formalismus auch keine Karriere im Bereich der maschinellen Sprachverarbeitung.

Die Grundsätze der Dependenzgrammatik und ihrer wichtigsten Komponente, der Valenztheorie sollen bereits in den 30-er Jahren aufgetaucht sein (vgl. [33], SS. 1-2.), jedoch wurde die Aufmerksamkeit der Linguisten aus vielen Ländern erst nach dem Erscheinen zweier Arbeiten von L. Tesnière [34, 35] auf die Abhängigkeitsgrammatik gelenkt. Vor allem in Deutschland wurde sie populär, so daß die bedeutendsten syntaktischen Beschreibungen des Deutschen seitdem in diesem theoretischen Rahmen entstanden sind (vgl. [17, 10] usw.). Eines der ersten deutschen MÜ-Systeme SUSY (*S*aarbrücker *Ü*bersetzungs*s*ystem, siehe [25]), funktioniert auch mit dem dependenzgrammatischen Formalismus. Dabei werden durch die Transferkomponente nicht nur die Lexeme, sondern auch ihre Valenz in die Zielsprache übersetzt. Die Ausgabe der Analyse ist unabhängig von der Zielsprache, nur der Transfer ist sprachspezifisch.

Während Dependenzgrammatik eine z.T. auch zu Unterrichtszwecken verwendete Forschungsrichtung ist, gibt es auch solche Theorien, die eher für rechnerische Implementierung gedacht sind. Die Phrasenstrukturgrammatiken liegen mehreren Parsern zugrunde. Für das Deutsche wurde vor allem die generalisierte Phrasenstrukturgrammatik (GPSG) angewandt (siehe [13, 36, 50]). Sie ist eine kontextfreie Grammatik, die keine Transformationsregeln enthält. Die ursprüngliche Version mußte später mit einer Wortstellungskomponente erweitert werden, damit man auch solche Sprachen verarbeiten konnte, deren Wortstellung freier ist als die des Englischen (z.B. Deutsch). Man hat außerdem vorgeschlagen, bei der Analyse auch die Valenz in Betracht zu ziehen (es ist kaum verwunderlich, daß dies in der deutschprachigen Computerlinguistik vorgefallen ist - s. [36], S. 143). In der deutschen MÜ arbeitet man auch mit GPSG (s. [16]).

Die theoretischen Ansätze der GPSG wurden u.a. in der HDPSG (Head-driven Phrase Structure Grammar) weitergeführt ([54]). Sie

stellt zugleich eine sog. "Kopfgrammatik" dar, d.h. die syntaktischen und gewisse semantische Merkmale der Elemente eines Syntagmas werden durch das dominierende Glied, den Kopf bestimmt, und bei der Analyse (Parsing) wird zunächst nach diesem Kopf gesucht. Somit können die anschließend identifizierten Elemente gleichzeitig syntaktisch und semantisch charakterisiert werden (siehe auch [30], S. 368).

Es gibt eine Reihe Grammatiken, die eine Operation gebrauchen, welche gemeinhin Unifikation genannt wird. Das sind die sog. Unifikationsgrammatiken (UG). Bei der Unifikation versucht man, zwei Größen der Grammatik zu vereinigen. Sie lassen sich nur dann verknüpfen, wenn ihre gemeinsamen Kategorien dieselben Werte haben (wie etwa bei der morphologischen Kongruenz, aber Unifikation ist in der UG ein generelles Verfahren). Die GPSG war auch eine Art UG, ebenso wie die Funktionale Unifikationsgrammatik (FUG, [19]). Die funktionalen Theorien der 80er Jahre haben die z. T. traditionellen grammatischen Funktionen wie Prädikat, Subjekt, Objekt usw. in die generative Linguistik als Neuerungen eingeführt (die kommen allerdings auch in der Semantikkomponente der GPSG vor).

Eine andere, bei Computerlinguisten beliebte funktionale Theorie, die auch mit Unifikation operiert, heißt lexikalisch-funktionale Grammatik (LFG), u. z. wegen der stark angereicherten lexikalischen Komponente ([3]). Sie hat auch keine Transformationsregeln, sogar passivische Verbformen werden im Lexikon gespeichert, und regelmäßige Zusammenhänge zwischen Aktiv- und Passivformen werden durch lexikalische Redundanzregeln wiedergegeben.

Die LFG wird auch in der MÜ der deutschen Sprache verwendet ([31]). Da diese Grammatik die in generativen Grammatiken geläufige Konstituentenstruktur der Oberfläche mit einer Funktionalen Struktur (F-Struktur) verbindet, braucht man mit Hilfe der Transferkomponente nur die F-Strukturen - und natürlich die Lexeme - zu übersetzen. Ein Teil des Transfers kommt somit in die Komponenten der Analyse und Synthese, bzw. in das einsprachige Lexikon. Im zweisprachigen Lexikon des Transfers regieren die äquivalenten Lexeme der beiden Sprachen meistens ziemlich ähnliche Funktionen, weil diese Ebene den

universellen thematischen Rollen wie Agens, Patiens usw. näher liegt als die Ebene der Oberflächenkasus (oder anderer Realisierungen von Funktionen). Rohrer zeigt am Beispiel des Satzes *Maria denkt an den kleinen Jungen*, wie F-Strukturen übersetzt werden (vgl. [31], SS. 85-86.).

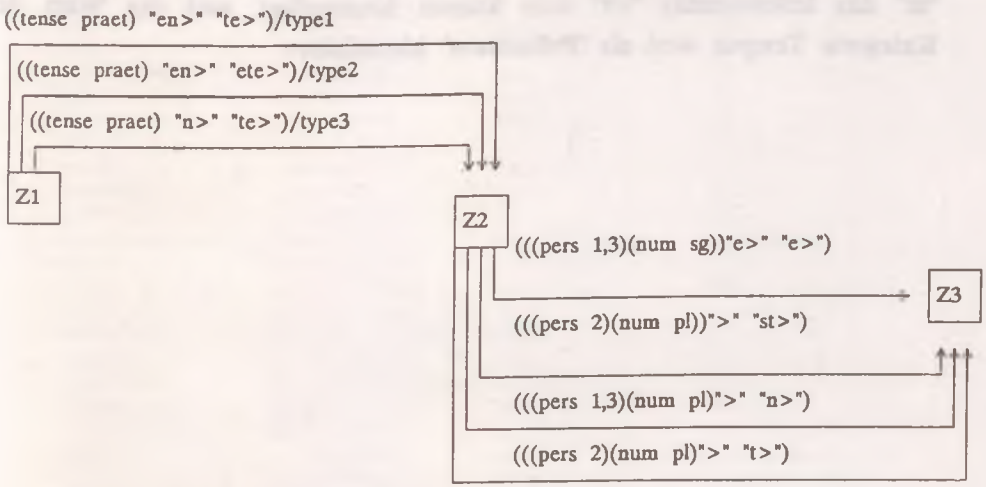
```
[ PRED      'denken ,<(↑SUBJ) (↑AN_OBJ)>'
  SUBJ      [ PRED 'Maria'
              CASE NOM ]

  AN_OBJ    [ PCASE AN
              CASE AKK
              SPEC DEF
              NUM SG
              PRED 'Junge'

              ADJ [PRED 'klein'] ]
]
```

Die F-Struktur des entsprechenden englischen Satzes *Mary thinks about the little boy:*

```
[ PRED      'think <(↑SUBJ) (↑ABOUT_OBJ)>
  SUBJ      [PRED 'Mary']

  ABOUT_OBJ [ PCASE about
              PRED 'boy'

              ADJ [PRED 'little']

              NUM SG
              SPEC the ]
]
```

Der Wert von PRED (predicate) ist immer ein Lexem. Bis auf das oberste PRED stehen die anderen in eckigen Klammern als Attribute von Funktionen, zusammen mit den anderen Attributen, die z.T. grammatische Kategorien (Kasus, Numerus usw.) des betreffenden Lexems angeben. Neben der Kategorie steht ihr Wert (z.B. Kasus Akkusativ). Am wichtigsten ist das oberste PRED, dessen Wert das regierende Lexem der Phrase ist, samt seiner funktionalen Valenz. Diese Funktionen erscheinen auch unter dem PRED, rechts von ihnen werden in den Klammern ihre Attribute aufgezählt.
Bei der Übersetzung wird zuerst ein Äquivalent zum obersten PRED gesucht. Die Zahl der Valenzstellen muß dabei mit der des Äquivalentes übereinstimmen. Die Funktionen bestimmen teilweise auch die Werte ihrer Attribute (z.B. PCASE about). Dann müssen die Lexeme (PRED), die diese Funktionen verkörpern, auch übersetzt werden (z.B. 'Junge' → 'boy'), sowie die anderen Attribute mit ihren Werten (z.B. SPEC DEF → SPEC the).
Die MÜ mit LFG scheint allerdings ohne die Valenz auch nicht funktionieren zu können, denn auf der S. 86 steht bei Rohrer ([31]) folgendes: "Im zweiten Schritt wird im Transferlexikon der passende Eintrag für das Verb 'denken' gesucht, der die gleiche Valenz wie in der vorliegenden F-Struktur hat."
Es gibt auch solche Grammatiken, die sogenannte endliche Zustandsautomaten darstellen. Hierzu gehören die verschiedenen ATN-Grammatiken (augmented transition networks, z.B. [52]), und auch die bekannte Zwei-Ebenen-Morphologie (two-level morphology) von Koskenniemi [22].
Endliche Automaten sind gut geeignet, morphologische Synthese und Analyse von Wortformen durchzuführen. Der endliche Automat von Paulus ([55]) beispielsweise analysiert und synthetisiert (generiert) deutsche und französische Verbformen. Im Deutschen wird die Aufgabe durch den Stammvokalwechsel erschwert, weil die den Stammvokal ersetzenden Vokale nicht vorhersagbar sind und der auszutauschende Vokal nicht immer leicht lokalisierbar ist (z.B. bei anerkennen kann eine Ersetzungsregel "e → a" nicht mechanisch funktionieren). Das

Problem wird mittels spezieller Lexikoneinträge gelöst, indem "der Stammvokal durch einen neutralen Platzhalter ersetzt und an das Wortende verschoben" wird ([55], S. 343).
Ein endlicher Automat besteht aus Zuständen. Es gibt einen Zustand, zu dem kein Weg führt (der also Ausgangszustand ist), und einen, von dem kein Weg weiterführt (der also Endzustand ist). Zwischen diesen können weitere Zustände sein. Zwei Zustände können durch mehrere, verschiedene Wege (Kanten) verbunden werden. Die verschiedenen Wege bedeuten verschiedene Zustandsveränderungen, die durch Regeln beschrieben werden. Ein Weg wird auf Grund der Merkmale des betreffenden Lexems gewählt, indem man diese Merkmale mit den bei den Kanten angegebenen Merkmalen unifiziert.
Der untenstehende Automat synthetisiert (und analysiert, mit entgegengesetzter Pfeilrichtung) die Präteritalformen von deutschen schwachen Verben. Im Lexikon muß bei jedem Verb angegeben werden, ob es zum Typ 1 (z.B. mach/en - mach+te), zum Typ 2 (z.B. meld/en - meld+ete) oder zum Typ 3 (z.B. handel/n - handel+te) gehört.



Z1 ist der Ausgangszustand der Synthese (Generierung) und - mit verkehrten Pfeilen - der Endzustand der Analyse von Präteritalformen.

Dementsprechend ist Z3 Endpunkt der Synthese und Ausgangspunkt der Analyse. Die Regel an der oberen Kante zwischen Z1 und Z2 hat folgende Bedeutung: Wenn die Ausgangsform der Infinitiv eines Verbs des Typs 1 ist und sein Tempus den Wert "Präteritum" erhalten soll, muß "en" am Wortende durch "te" ersetzt werden. Die untere Kante zwischen Z2 und Z3 enthält folgende Regel: Wenn das Verb im Zustand Z2 die kategoriellen Werte "2. Person" und "Plural" erhalten soll, muß an das Wortende noch ein "t" gefügt werden.
Bei der Analyse ist Z3 der Ausgangspunkt. Die Regel der unteren Kante funktioniert diesmal in der anderen Richtung: Wenn am Ende des Wortes ein "t" ist, dann wird es eliminiert, und die Werte der Kategorien Person und Numerus werden als "2. Person" und "Plural" identifiziert. Nach dem Zustand Z2 wird "te" oder "ete" abgetrennt. Der letzte Schritt dieser Analyse ist eigentlich Synthese, da die kanonische Form des Lexems, das Lemma auch hergestellt wird (das ist der Infinitiv). Dazu muß im Inventar der Stämme der dort angegebene Typ ausgesucht werden. Wenn das der Typ 1 ist, dann wird anstelle von "te" das Infinitivsuffix "en" zum Stamm hinzugefügt, und der Wert der Kategorie Tempus wird als "Präteritum" identifiziert.

Anhang

Anhang 1

Eine Seite aus dem rückläufigen Wörterbuch von Mater [26].

Motto	Rennauto	Reversierduo	nirgendswo
Risotto	Feuerwehrauto	Mandschukuo	anderswo
Putto	Überfallsauto	Quidproquo	sonstwo
brutto	Privatauto	Quiproquo	zwo
Auto	Mietauto	bravo	Yo-Yo
Spielzeugauto	Lastauto	wo	Embryo
Polizeiauto	Dienstauto	irgendwo	Bajazzo
Personenauto	Basuto	nirgendwo	Terrazzo
Raketenauto	Duo	allwo	Intermezzo

P

Kap	Reaktionsprinzip	Stroboskop	Heliotrop
Nordkap	Induktionsprinzip	Kaleidoskop	isotrop
Handikap	Personalitätsprinzip	Geoskop	Isop
Satrap	Rentabilitätsprinzip	Stereoskop	Ysop
Reep	Utilitätsprinzip	Laryngoskop	Isotop
Fallreep	Autoritätsprinzip	Nephoskop	stop
Piep	Identitätsprinzip	Stethoskop	schlapp
Step	Wirtschaftsprinzip	Bioskop	Geschlapp
Clip	Nützlichkeitsprinzip	Glimmlicht-	Julklapp
Flip	Leitprinzip	oszilloskop	Bärlapp
Pip	Alp	Anemoskop	Kolbenbärlapp
Trip	Hochalp	Uranoskop	Keulenbärlapp
Tip	Skalp	Galvanoskop	Tannenbärlapp
Geheimtip	Schulp	Ikonoskop	Alpenbärlapp
Tototip	Pulp	Chronoskop	knapp
Partizip	Zulp	Baroskop	geldknapp
Prinzip	Camp	Urethroskop	Papp
Stabprinzip	Kamp	Mikroskop	papperlapapp
Grundprinzip	Tramp	Ultramikroskop	Berapp
Parteiprinzip	Vamp	Metallmikroskop	Krapp
Regionalprinzip	Pomp	Elektronenmikroskop	Trapp
Moralprinzip	Lump	Elektronmikroskop	Wolfstrapp
Kausalprinzip	plump	Beobachtungs-	Schwapp
Fundamentalprinzip	Haderlump	mikroskop	Depp
Wahlprinzip	Gesinnungslump	Horoskop	Schlepp
Auswahlprinzip	Erzlump	Elektroskop	Ochsenschlepp
Planprinzip	Pump	Kondensator-	Krepp
Dopplerprinzip	Hutstump	elektroskop	Klipp
Führerprinzip	Olymp	Spektroskop	Geklipp
Entwicklungsprinzip	Epidiaskop	Gyroskop	Schnipp
Bevölkerungsprinzip	Teleskop	Zystoskop	Tripp
Leitungsprinzip	Spiegelteleskop	Zyklop	Süpp
Leistungsprinzip	Zenitteleskop	Mop	hopp
Lebensprinzip	Polariskop	Philanthrop	Galopp
Organisationsprinzip	Periskop	Misanthrop	Handgalopp

Anhang 2

Eine Seite aus dem rückläufigen Wörterbuch der ungarischen Sprache [29]. Die Zahlen sind Kodenummern für Wortart, Stammtyp, Flexionstyp, Wortbildungsstruktur.

	A	B	C	D	E	F	G
SZORNY	1	2	1	04	04	04	
HUNY	1	1	1	10	00	00	
BEHUNY	6	1	1	10	00	01	
LEHUNY	6	1	1	10	00	01	
KIHUNY	6	201	1	10	00	00	
ELHUNY	6	1	1	10	00	00	
SUNY	1	1	1	10	00	00	
LESUNY	6	1	1	10	00	01	
GÚNY	1	2	1	03	00	21	
ONGÚNY	2	2	1	03	00	01	9

1114

O

O	1	2	1	03	03	02	
CO	1	11		00	00	00	
ALTKREOO	9	2	2	33	33	02	
VULOO	1	6		00	00	00	
HOHOHO	1	10		00	00	00	
SOSTISSIMO	1	62	1	03	03	02	
NO	1	11		00	00	00	
PIANO	1	62	1	03	03	02	
ANNO	1	6		00	00	00	
NONO	2	11		00	00	00	
NONONO	3	11		00	00	00	
ALLEGRO	1	62	1	03	03	02	
PRO	1			00	00	00	
RECITATIVO	9	2	1	03	03	02	
INTERMEZZO	9	2	1	03	03	02	

15

Ó

ó[1]	1	2	1	03	03	02	
ó[3]	1	3	1	00	00	22	
ó[4]	1	102	1	10	00	00	
KAKAÓ	1	2	1	03	33	02	
PÁRAÓ[1]	1	2	1	03	03	02	
PÁRAÓ[2]	1	2	1	03	03	02	
ZSABÓ	1	2	1	03	03	02	
SZABÓ	1	32	1	03	03	02	1
FÉRFISZABÓ	3	2	1	03	03	02	1
SZŰRSZABÓ	2	2	1	03	03	02	1
BIMBÓ	1	2	16	03	03	02	
RÓZSABIMBÓ	2	2	1	03	03	02	
VIRÁGBIMBÓ	2	2	1	03	03	02	
KELBIMBÓ	2	2	1	03	03	02	
MELLBIMBÓ	2	2	1	03	03	02	
TERMŐBIMBÓ	2	2	16	03	03	02	
CSECSBIMBÓ	2	2	1	03	03	02	
DOBÓ	1	32	1	03	03	02	1
SÜLYDOBÓ	2	32	1	03	03	02	1
AVERBÓ	9	2	1	03	03	02	
GUBÓ	1	2	1	03	03	02	
MÁKOUBÓ	2	2	1	03	03	02	
SELYEMGUBÓ	2	2	1	03	03	02	

	A	B	C	D	E	F	G
LENGUBÓ	2	2	1	03	03	02	
CÉGÓ	1	2	1	03	03	02	
COCÓ	1	2	1	03	03	02	
DÓ	1	2	1	03	03	02	
ADÓ[1]	1	32	1	03	03	02	1
ADÓ[2]	1	2	16	03	03	02	1
JÖVEDELEM-							
TÖBBLETADÓ	3	2	1	03	03	02	9
ODAADÓ	6	3	1	00	15	02	1
MUNKAADÓ	2	23	1	03	03	02	1
HÁLAADÓ	2	32	1	03	03	02	1
ÓRAADÓ	2	32	1	03	03	02	1
ÉBADÓ	2	2	1	03	33	02	1
FÖLDADÓ	2	2	1	03	03	02	1
TŐKEADÓ	2	2	1	03	33	02	3
LEADÓ	6	32	1	03	03	02	1
RÁDIÓLEADÓ	7	2	1	03	03	02	1
DAGADÓ	1	32	1	03	03	02	1
MEGRAGADÓ	6	3	1	00	01	02	1
ELRAGADÓ	6	3	1	00	01	02	1
TAGADÓ	1	3	1	00	13	00	1
ISTENTAGADÓ	2	32	1	03	03	02	1
HITTAGADÓ	2	32	1	03	03	02	3
MEGADÓ	6	3	1	00	51	00	1
ÍNSÉGADÓ	2	2	1	03	03	02	3
VIGADÓ	1	32	1	03	03	02	1
HANGADÓ	2	32	1	03	03	02	1
RANGADÓ	2	32	1	03	03	02	1
FOGADÓ	1	32	1	03	03	02	1
VENDÉGFOGADÓ	2	2	1	03	03	02	1
ELFOGADÓ	6	32	1	03	03	02	1
SZÓFOGADÓ	2	3	1	00	00	22	1
ENYHADÓ	2	3	1	00	01	00	1
KIADÓ	6	32	1	03	03	02	1
CSOMAGKIADÓ	7	32	1	03	03	02	1
LAPKIADÓ	7	32	1	03	03	02	1
ZENEMŰKIADÓ	8	32	1	03	03	02	1
KÖNYVKIADÓ	7	32	1	03	03	02	1
RIADÓ	1	32	1	03	03	02	1
LÉGIRIADÓ	2	2	1	03	03	02	3
FEJADÓ	2	2	1	03	03	02	1
MÉRTÉKADÓ	2	3	1	10	00	00	3
LANKADÓ	1	32	1	03	03	02	1
HALADÓ	1	32	1	03	03	02	1
ELADÓ	6	32	1	03	03	02	1
FELADÓ	6	32	1	03	03	02	1
VISSZONTKLADÓ	7	32	1	03	03	02	9
TÁMADÓ	1	32	1	03	03	02	1
SZÁMADÓ	2	32	1	03	03	02	1
JÖVEDELEMADÓ	2	2	1	03	03	02	3
VAGYONADÓ	2	2	1	03	03	02	9
RÁDIÓADÓ	2	2	1	03	03	02	1
ELŐADÓ	6	32	1	03	03	02	9
SZAKELŐADÓ	7	2	1	03	03	02	1
FŐELŐADÓ	7	2	1	03	03	02	1
VISSZAMARADÓ	6	32	1	03	03	02	1
TÚLÁRADÓ	6	3	1	00	51	02	1
VÉRADÓ	2	32	1	03	03	02	1
KENYÉRADÓ	2	32	1	03	03	02	1
HÍRADÓ	2	32	1	03	03	02	1
VILÁGHÍRADÓ	3	2	1	03	03	02	1

Anhang 3

Die erste Seite der Liste der 1000 häufigsten Lexeme aus der ungarischen schöngeistigen Prosa (zusammengestellt von Mihály Füredi u. József Kelemen, 1983). (E1 = É, O1 = Ó, O2 = Ö, O3 = Ő usw., HSzf = Kodenummer der Wortart, Gyak = Häufigkeit.)

Lexéma	Szf	Gya	Lexéma	Szf	Gya
A	80	5582	MINTHA	92	00737
ES	91	0875	NAP	20	00700
NEM	263	0862	MAGA	252	00700
HOGY	292	0720	UTA1N	70	00696
VAN	310	0467	FEJ	220	00699
AZ	52	0441	U1R	20	00748
IS	91	0442	ARC	20	00682
DE	291	0363	NINCS	10	00682
EGY	280	0352	RA1	60	00679
VAN	310	0340	TE	152	00658
EZ	52	0320	MINDIG	60	00679
CSAK	63	0314	SZO1	20	00678
MA1R	63	0287	PEDIG	91	00685
S	291	0293	GONDOL	10	00672
MOND	10	0277	MEGY	10	00692
ME1G	63	0260	LA1NY	20	00713
TUD	10	0233	MINDEN	53	00662
HA	192	0212	MINDEN	52	00630
MAGA	152	0202	TESZ	10	00603
EZ	53	0192	KE1RDEZ	10	00628
EIN	52	0189	JO2N	10	00582
SEM	63	0181	SZERET	10	00569
OB	52	0178	E1LET	20	00554
KELL	10	0173	ASSZONY	20	00587
MI	552	0161	ELO3TT	70	00549
IS	63	0164	SEMMI	52	00539
MINT	292	0159	TALA1N	63	00536
MOST	60	0151	KEZD	10	00532
MERT	92	0146	E1V	20	00549
AKI	52	0148	E1PPEN	63	00525
NEKI	60	0145	E1REZ	10	00559
EMBER	20	0137	VALAMI	752	00517
UGY	60	0130	HANG	20	00515
ITT	60	0120	VALAMI	753	00515
VOLNA	11	0120	FE1RFI	20	00551
AKAR	10	0115	ALATT	70	00510
AKKOR	60	0116	DOLOG	20	00497
EGY	143	0113	EGYMA1S	52	00504
LESZ	10	0111	FIU1	20	00550
AZ	53	0103	MARAD	10	00457
LA1T	10	0100	OLYAN	53	00488
JO1	30	0101	JO1L	60	00454
AMI	652	01020	U1T	20	00462
KE1Z	20	00982	ANYA	20	00485
NAGY	30	00967	TART	10	00457
VAGY	91	00964	KO2ZO2TT	70	00442
AZTA1N	60	00985	HA1Z	20	00460
KE1T	43	00970	HOZZA1	60	00438
OTT	60	00904	SZE1P	30	00438
AMIKOR	60	00887	HELY	20	00434
MEG	191	00891	U1J	30	00444
NE1Z	10	00865	VA1R	10	00435
IDO3	20	00804	FELE1	70	00427
NE	163	00779	SZOBA	20	00431
SZEM	20	00776	ELSO3	43	00402
I1GY	60	00791	OLYAN	56	00421
LEHET	10	00781	APA	20	00494
VELE	60	00762	AJTO1	20	00453
KIS	30	00777	AD	110	00409
IS	96	00749	KE1R	10	00441
HA1T	63	00745	U2L	10	00416
NAGYON	60	00738	BESZE1S	10	00420
A1LL	10	00727	VESZ	110	00421
MAJD	163	00723	GYEREK	20	00413
			HISZ	110	00412

Anhang 4

Die vom Longman Mini-Concordancer zusammengestellte Konkordanzliste der Wörter *ich* und *die* aus einem Kurzkrimi (s. Anhang 9).

```
Concordance for "ich"  (25 lines)
Text: KRUGER.TXT

 Luerzere Briefe. Nachher sollt ich nur noch Herrn Breitner anrufen und
morgen frueh absagen. Das musste ich in einer Telefonzelle machen.   was
lassen: Um halb neun morgens hab ich ihn das letzte Mal gesprochen. Herr
 gab er mir die Autcschluessel. Ich fuhr noch  am Vormittag hin, und er
 der Motor. Die Reparatur konnte ich an Ort und  Stelle nicht machen, al
lle nicht machen, also schleppte ich den Volkswagen ab. Er steht auch
tellvertreter von Herrn Krueger: Ich  war kurz vor 22 bei Herrn Krueger.
r kurz vor 22 bei Herrn Krueger. Ich musste einige Papiere dringend
war nicht das    erste Mal, dass ich ihn zu Hause besuchte. Aber ich ruf
 ich ihn zu Hause besuchte. Aber ich rufe ihn natuerlich vorher immer an
h vorher immer an. und frage, ob ich nicht stoere. Auch diesmal machte i
 cht stoere. Auch diesmal machte ich es so. Herr  Krueger sagte im Telef
 Herr  Kruegar sagte im Telefon, ich solle vor 22 kommen, denn er bekomm
er, Nachbar von Herrn Krueger:   Ich war bei ihm an dem Tag. Er hat raem
 nach Haus kam, also um 16, ging ich rueber zu Herrn Krueger und gab ihm
 Herrn Krueger: An dem Tag hatte ich mehr als Pech. Mein Wagen sprang
 Wagen sprang      nicht an, als ich [illegible] um eins zu einem Patienten geruf
in Taxi zu bekommen. Also musste ich die Strassenbahn       nehmen. Aber w
se bahnschienen geworfen. Soviel ich weiss,  sind die zerrissenen Leitun
 mussten also   alle aussteigen. Ich bin dann zu Fuss losgegangen, denn
 dann zu Fuss losgegangen, denn ich wusste, dass Herr  Krueger in der
he wohnt und von dort aus wollte ich telefonieren, dass ich  etwas spae
 s wollte ich telefonieren, dass ich   etwas spaeter komme. Doch dann sa
was spaeter komme. Doch dann sah ich an der Ecke eine Telefonzelle, und
 Telefonzelle. und von dort habe ich angerufen.
```

```
Concordance for "die"  (9 lines)
Text: KRUGER.TXT

 den 5 verhoerten Personen war, die ihre Aussagen schon am    ersten Ta
eslegten. Der Taeter sagte nicht die Wahrheit, und wenn Sie die    Aussa
nicht die Wahrheit, und wenn Sie die   Aussagen aufmerksam lesen, werde
 noch Herrn Breitner anrufen und die  Fahrt fuer morgen frueh absagen.
n Taxi, und unterwegs gab er mir die Autcschluessel. Ich fuhr noch  am V
ttag hin, und pruefte den Motor. Die Reparatur konnte ich an Ort und  St
axi zu bekommen. Also musste ich die Strassenbahn       nehmen. Aber wie e
         entwurzelt und quer ueber die Strassenbahnschienen geworfen. Sovi
eworfen. Soviel ich weiss,  sind die zerrissenen Leitungen noch immer ni
```

Anhang 5

Die 120 häufigsten Wortformen aus der Kaeding-Zählung, die Ende des 19. Jh. stattfand und knapp 11 Millionen Textwörter umfaßte. Die Ergebnisse der manuellen Zählung, an der mehrere Jahre lang etwa 1000 Mitarbeiter beteiligt waren, wurden in den 70er Jahren von W.D. Ortmann et al. rechnerisch verarbeitet. Die abgebildete Seite befindet sich im Band "W.D. Ortmann: Hochfrequente deutsche Wortformen, I. Herausgegeben vom Goethe-Institut, München, 1975." Siehe auch [28]

RANGREIHE-SORTIERUNG 1. DIE - 120. VIEL

1.	DIE	349553	61.	ZUM	23278
2.	DER	341522	62.	ZUR	22794
3.	UND	320072	63.	KANN	22113
4.	IN	188078	64.	DOCH	21994
5.	ZU	172625	65.	VOR	21851
6.	DEN	138664	66.	DIESER	21811
7.	DAS	124232	67.	MICH	21334
8.	NICHT	114518	68.	IHN	20785
9.	VON	113201	69.	DU	20107
10.	SIE	102212	70.	HATTE	19929
11.	IST	96970	71.	SEINE	19718
12.	DES	96190	72.	MEHR	18549
13.	SICH	92945	73.	AM	18523
14.	MIT	91552	74.	DENN	18488
15.	DEM	89109	75.	NUN	17891
16.	DASS	87969	76.	UNTER	17636
17.	ER	87029	77.	SEHR	17293
18.	ES	86778	78.	SELBST	16911
19.	EIN	85919	79.	SCHON	16727
20.	ICH	82207	80.	HIER	16667
21.	AUF	80944	81.	BIS	16399
22.	SO	74273	82.	HABE	16105
23.	EINE	69304	83.	IHRE	16098
24.	AUCH	60750	84.	DANN	15545
25.	ALS	58331	85.	IHNEN	15254
26.	AN	55710	86.	SEINER	15067
27.	NACH	54760	87.	ALLE	14992
28.	WIE	51336	88.	WIEDER	14693
29.	IM	50770	89.	MEINE	14552
30.	FUER	50559	90.	ZEIT	14529
31.	MAN	44284	91.	GEGEN	14430
32.	ABER	44201	92.	VOM	13636
33.	AUS	40615	93.	GANZ	13548
34.	DURCH	40329	94.	EINZELNEN	13466
35.	WENN	40108	95.	WO	13145
36.	NUR	39507	96.	MUSS	13069
37.	WAR	39395	97.	OHNE	12895
38.	NOCH	39179	98.	EINES	12795
39.	WERDEN	39085	99.	KOENNEN	12709
40.	BEI	38844	100.	SEI	12553
41.	HAT	38159	101.	JA	12527
42.	WIR	37840	102.	WURDE	12510
43.	WAS	35220	103.	JETZT	11859
44.	WIRD	34589	104.	IMMER	11664
45.	SEIN	31462	105.	SEINEN	11647
46.	EINEN	31229	106.	WOHL	11356
47.	WELCHE	30974	107.	DIESES	11113
48.	SIND	30532	108.	IHRES	11014 1)
49.	ODER	30329	109.	WUERDE	10794
50.	UM	30206	110.	DIESEN	10558
51.	HABEN	28822	111.	SONDERN	10482
52.	EINER	28086	112.	WEIL	10473
53.	MIR	26804	113.	WELCHER	10372
54.	UEBER	25497	114.	NICHTS	10332
55.	IHM	25214	115.	DIESEM	10225
56.	DIESE	24706	116.	ALLES	10040
57.	EINEM	24232	117.	WAREN	9957 2)
58.	IHR	23790	118.	WILL	9811
59.	UNS	23765	119.	HERR	9688
60.	DA	23497	120.	VIEL	9553

- C 1 -

Anhang 6

Die vom Longman Mini-Concordancer zusammengestellte Liste der häufigsten Wortformen eines Kurzkrimis (siehe Anhang 9).

Anhang 7

Phonemhäufigkeit in Adys sämtlichen Gedichten [18]. ("[" steht für "í", "_" für "ű".)

A feldolgozás összes[tése

összes vers	=	1048		
összes szó	=	115564		
összes hang	=	552788		
összes C	=	327075	59.17 0/0	
összes V	=	225713	40.83 0/0	
összes V(pal)	=		115592	51.21 0/0
összes V(vel)	=		110121	48.79 0/0

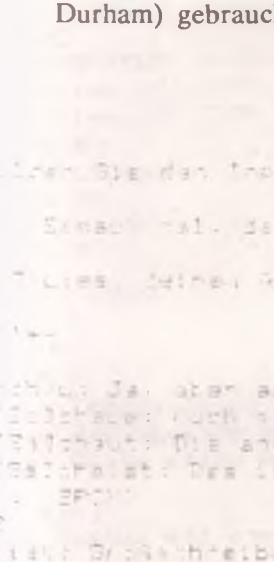
átlagos szóhossz=	4.783 hang/szó	1.953 V/szó	
V/C	= 0.690		
V(pal)/V(vel)	= 1.050		

a) betürendben

hang	db	0/0	vektor	hang	db	0/0	vektor
ä	1	0.00		x	14	0.00	
cz	22	0.00		y	38	0.01	
q	0	0.00		ch	15	0.00	
qu	1	0.00		cch	1	0.00	
w	5	0.00					
a	50132	9.07	417	lly	16	0.00	0
á	19782	3.58	164	m	26930	4.87	224
b	9077	1.64	75	mm	375	0.07	3
bb	1106	0.20	9	n	33216	6.01	276
c	1310	0.24	11	nn	1034	0.19	9
cc	28	0.01	0	ny	3235	0.59	27
cs	3834	0.69	32	nny	441	0.08	4
ccs	12	0.00	0	o	25449	4.60	211
d	14044	2.54	117	ó	5528	1.00	46
dd	311	0.06	3	ö	6899	1.25	57
dz	13	0.00	0	ő	4008	0.73	33
ddz	0	0.00	0	p	4639	0.84	39
dzs	5	0.00	0	pp	129	0.02	1
ddzs	0	0.00	0	r	23807	4.31	198
e	55383	10.02	460	rr	421	0.08	3
é	18909	3.42	157	s	21461	3.88	178
f	5075	0.92	42	ss	634	0.11	5
ff	17	0.00	0	sz	10286	1.86	85
g	13674	2.47	114	ssz	1135	0.21	9
gg	213	0.04	2	t	31692	5.73	263
gy	8261	1.49	69	tt	4070	0.74	34
ggy	51	0.01	0	ty	146	0.03	1
h	11304	2.04	94	tty	33	0.01	0
hh	5	0.00	0	u	6630	1.20	55
i	23235	4.20	193	ú	2600	0.47	22
[	2347	0.42	20	ü	3779	0.68	31
j	8759	1.58	73		1032	0.19	9
jj	301	0.05	3	v	12775	2.31	106
k	30655	5.55	255	vv	78	0.01	1
kk	504	0.09	4	z	8914	1.61	74
l	28453	5.15	236	zz	399	0.07	3
ll	1939	0.35	16	zs	545	0.10	5
ly	1712	0.31	14	zzs	1	0.00	0

Anhang 8

Eine kurze Übung, die nach einer Bearbeitung mit dem Hilfsprogramm AUTHOR im Rahmen des Autorenprogramms CALIS ((c) Duke University, Durham) gebraucht werden kann.



Erklärung der Symbole:

\...\	: Lücke (Text in Klammern ausblenden)
<...>	: Symbol zur Identifizierung der Lücke
!!!+-	: Anfang des Auswertungsteiles
#...	: Symbol der Lücke im Auswertungsteil
+	: andere richtige Lösung
-	: antizipierte falsche Lösung
;	: Anfang des Kommentars
[...]	: beliebiges von den Zeichen in Klammern
*	: beliebige Zeichenkette
&	: eine oder mehrere beliebige Zeichenketten
<SPCK>	: Fehleranalyse Buchstabe für Buchstabe (spellcheck)

AUTHOR erstellt daraus folgende Textdatei:

```
CLOZE
     Calis tools program version 3.77
  Existing text file is imperati.txt


Bilden Sie den Imperativ der Verben in Klammern!

\<1>Schau\ mal, da kommt dein Onkel! (schauen)

\<2>Lies\ deinen Aufsatz vor! (vorlesen)


  Current date and time are Tue Jan 11 11:59:25 1994


#0
!                                CLOZE  PROCEDURE
!
!
!   Please fill in each blank in the following text.  Every blank is
!   recoverable through contextual clues.  First, read through the text
!   then begin to fill in the blanks.  You may use the white arrow keys
!   to scroll through the text if necessary.
!
!   Type the correct word or phrase into the blank where the cursor app
!
!   then type [ENTER].  Your answer may be larger or smaller than the b
!
?
? Bilden Sie den Imperativ der Verben in Klammern!
?
? <UL>  1  </UL> mal, da kommt dein Onkel! (schauen)
?
? <UL> 2  </UL> deinen Aufsatz vor! (vorlesen)
?

#1
?<Q><CUR: 1:1>
?<Q><CUR: 4:2>       mal, da kommt dein <CUR: 4:2>
+Schau:<Q><CUR: 4:2><UL>Schau</UL> mal, da kommt dein
+schau:<Q><CUR: 4:2><UL>Schau</UL> mal, da kommt dein
&<D>          Ja, aber am Satzanfang sollten Sie kein kleines s schreib
+[Ss]chaue:<Q><CUR: 4:2><UL>Schau</UL> mal, da kommt dein
&<D>                  Auch möglich, aber viel seltener gebraucht.
-[Ss]chaut:
&<D>                       Die angesprochene Person ist 2.P.Sg.
-[Ss]chaust:
&<D>                              Das ist Indikativ!
-%:
%<D>                                        <EOC>
-<PASS>:<Q><CUR: 4:2><UL>  1  </UL> mal, da kommt dein
-<GIVEUP>:<Q><CUR: 4:2><UL>Schau</UL> mal, da kommt dein
%<D>                          The Correct Answer is: Schau
```

```
#2
?<Q><CUR: 6:2>      deinen Aufsatz vor'<CUR: 6:2>
+Lies:<Q><CUR: 6:2><UL>Lies<UL> deinen Aufsatz vor'
+lies:<Q><CUR: 6:2><UL>lies<UL> deiner Aufsatz vor'
%<D>                                  Großschreibung'
-[Llies*:
%<D>                                    e-i-Wechsel'
-*e:
%<D>                               Bei e-i-Wechsel kein Suffix'

-[Vvlor&:
%<D>                                   Trennbares Verb'
-&:
%<D>                                        <SPCK>
-<PASS>:<Q><CUR: 6:2><UL>2  <UL> deiner Aufsatz vor'
-<GIVEUP>:<Q><CUR: 6:2><UL>Lies<UL> deinen Aufsatz vor
%<D>                              The Correct Answer is: Lies

#end
```

Die vom AUTHOR hinzugefügten Symbole:

#0 : Anfang der Übung

#end: Ende der Übung

! : Zeile für Anweisungen

? : Zeile für Übung

<Q>: Fenster für Übung

<CUR>: Kursorposition

<UL>: Unterstreichen ein/aus

<D>: Fenster für Rückmeldungen

<PASS>: Überspringen

<GIVEUP>: Aufgeben

Anhang 9

Ein Text, der vom Programm KURZKRIMIS generiert worden ist ((c) Goethe-Institut, Autoren: P.Uzonyi, Z.Papp, F.Megyery).

Fall Nr 13237

Im Norden hatte man damals einen ungewöhnlich heißen Sommer. Das war 1986. Der vorliegende Mord hat der Mordkommission viel Arbeit verursacht, obgleich der Mörder unter den 5 verhörten Personen war, die ihre Aussagen schon am ersten Tag ablegten. Der Täter sagte nicht die Wahrheit, und wenn Sie die Aussagen aufmerksam lesen, werden Sie den Widerspruch auch selbst finden. Auszüge aus dem Ermittlungsprotokoll: Um 23 Uhr wurde der Intendant Ludwig Krüger in seinem Hausgarten tot aufgefunden. Man hat ihn mit einem Briefbeschwerer aus Marmor ermordet. Der Mord muß zwischen 22.00 und 22.30 Uhr geschehen sein.

Aussage Nr 1: GABI JUNG (26), Sekretärin, der Herr Krüger jeden Abend zu Hause diktierte: An diesem mißglückten Abend diktierte er mir nur zwei kürzere Briefe. Nachher sollte ich nur noch Herrn Breitner anrufen und die Fahrt für morgen früh absagen. Das mußte ich in einer Telefonzelle machen, weil das Telefon von Herrn Krüger seit drei Tagen nicht funktionierte.

Aussage Nr 2: KURT WURM (47), Autoschlosser: Um halb neun morgens hab ich ihn das letzte Mal gesprochen. Herr Krüger hatte der Tage einen neuen Wagen gekauft. Das Auto hat aber bald versagt: er wollte am Morgen des betreffenden Tages mit dem Auto zur Arbeit, aber er konnte den Motor nicht anlassen. Nachdem er es ein paarmal vergeblich versucht hatte, setzte er sich in ein Taxi, und unterwegs gab er mir die Autoschlüssel. Ich fuhr noch am Vormittag hin, und prüfte den Motor. Die Reparatur konnte ich an Ort und Stelle nicht machen, also schleppte ich den Volkswagen ab. Er steht auch jetzt noch in meiner Werkstatt.

Aussage Nr 3: BRUNO WISSEMANN (43), Stellvertreter von Herrn Krüger: Ich war kurz vor 22 bei Herrn Krüger. Ich mußte einige Papiere dringend unterschreiben lassen, und blieb nur etwa fünf Minuten. Das war nicht das erste Mal, daß ich ihn zu Hause besuchte. Aber ich rufe ihn natürlich vorher immer an, und frage, ob ich nicht store. Auch diesmal machte ich es so. Herr Krüger sagte im Telefon, ich solle vor 22 kommen, denn er bekomme Besuch.

Aussage Nr 4: HORST KOWALSKI (51), Schweißer, Nachbar von Herrn Krüger: Ich war bei ihm an dem Tag. Er hat nämlich wieder einmal ein Paket bekommen. Herr Krüger hat mich gebeten, solche Sachen für ihn zu übernehmen, wenn er nicht zu Haus ist. Der Briefträger weiß natürlich auch Bescheid. Als er nach Haus kam, also um 16, ging ich rüber zu Herrn Krüger und gab ihm das Ding. Dann hat er mich wie immer zu einem Glas Kognak eingeladen.

Aussage Nr 5: Dr. ARNOLD STOLZ (58), Internist, seit acht Jahren Hausarzt von Herrn Krüger: An dem Tag hatte ich mehr als Pech. Mein Wagen sprang nicht an, als ich so um eins zu einem Patienten gerufen wurde. Wegen des Unwetters war auch kein Taxi zu bekommen. Also mußte ich die Straßenbahn nehmen. Aber wie es das Mißgeschick wollte, hatte der Sturm einen Baum entwurzelt und quer über die Straßenbahnschienen geworfen. Soviel ich weiß, sind die zerrissenen Leitungen noch immer nicht repariert. Wir mußten also alle aussteigen. Ich bin dann zu Fuß losgegangen, denn ich wußte, daß Herr Krüger in der Nähe wohnt und von dort aus wollte ich telefonieren, daß ich etwas später komme. Doch dann sah ich an der Ecke eine Telefonzelle, und von dort habe ich angerufen.

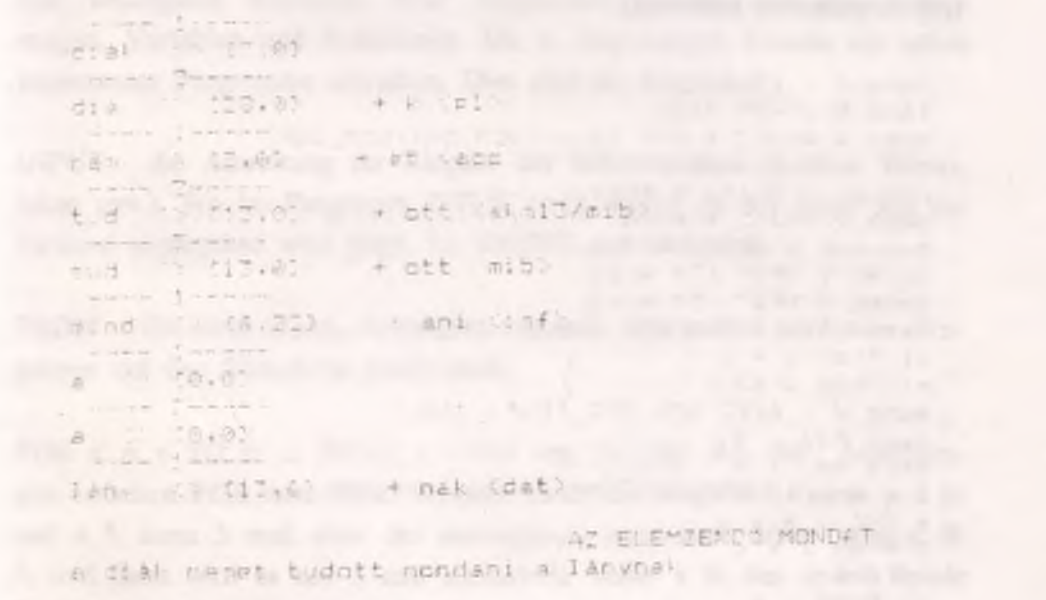
Anhang 10

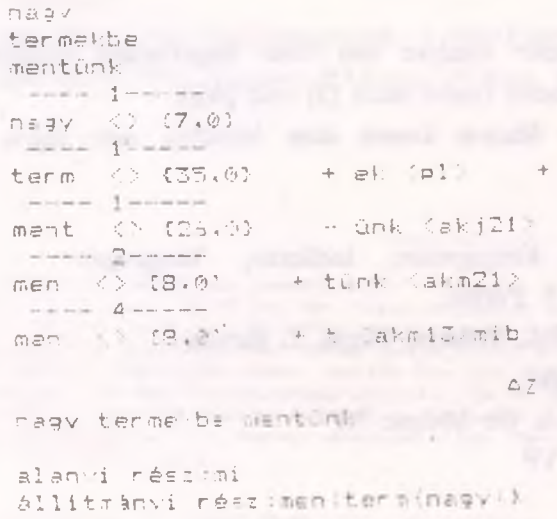
Morphologische und syntaktische Analyse von zwei ungarischen Sätzen durch einen experimentellen Parser (siehe auch [9] und [46]).

Übersetzung der Sätze: 1.*Der Student konnte dem Mädchen nein sagen.* 2. *Wir gingen in große Säle.*

Erklärung einiger Notationen:

akm13	: subjektive Konjugation, Indikativ, Vergangenheit, Singular, 3. Person
akj21	: subj. Konj., Ind., Präsens, Plural, 1. Person
mib	: Partizip Perfekt
alanyi r.	: Subjektteil, d.h. die höchste NP
állítmányi r.:	die höchste VP





Die zur Analyse der obigen Sätze verwendeten lexikalischen Einträge (mit homonymen Stämmen).

```
lány N . 17 .
lány N . -67 #lány
megy V AKJ13 * NOM_[dir]/NOM_DAT/NOM_INF
men V . -8 #megy
megyek V AKJ11 * #megy
négy V AKJ12 * #megy
megyünk V AKJ21 * #megy
mehet V VPOT 134 #megy
menes V VPOT -28 #megy
ment V . 26 NOM_ACC_[ABL]/NOM_ACC_[tel&l]
mi Pron . * .
mi Pron . 43 .
mond V . 4/22 NOM_ACC_[DAT]_[DEL]
nagy Adj . 7 .
nagg Adj . -17 #nagy
tud V . 13 NOM_ACC_[DEL]/NOM_INF/NOM_[ACC]_DEL
a Art . * .
a Pron . * .
az Art . * .
az Pron . * .
nem Op . * .
nem N . 2 .
dia N . 26 .
diá N . -28 #dia
diák N . 7 .
tér V . 2 NOM_dir
tér N . 45 .
ter N . -35 #tér
terem V . 17 NOM_[ACC]
term V . -34 #terem
terem N . 43 .
term N . -35 #terem
termet N . 2 .
```

Anhang 11

Über die Programmiersprache BASIC

Untersuchen wir nun einige "Vokabeln" einer künstlichen Sprache, die dem Programmierer mit Ausdrücken natürlicher Sprachen leichter umgehen helfen. Diese künstliche Sprache ist die Programmiersprache BASIC (Beginners All-purpose Symbolic Instruction Code). Diese Programmsprache war ursprünglich für Anfänger gedacht, und demgemäß konnte sie wesentlich weniger leisten als damalige höhere Sprachen wie FORTRAN, PL-1 usw. Zwischendurch hat man aber auch diese Sprache weiterentwickelt, immer bessere B-Compiler geschrieben, und heute sind die neusten Versionen von Quick Basic, Turbo Basic oder Visual Basic für die meisten Zwecke ebensogut oder mitunter sogar besser zu gebrauchen als Sprachen, die schon immer viel leisten konnten.
Die wichtigsten Elemente einer Programmiersprache sind die Anweisungen, Variablen und Funktionen. Mit 4 Anweisungen können wir schon interessante Programme schreiben. Dies sind die folgenden:

INPUT - die Anweisung zur Eingabe der Informationen (Zahlen, Wörter, Sätze usw.). Wo im Programm INPUT steht, wartet es, bis etwas mit der Tastatur eingegeben wird (eigtl. bis ENTER gedrückt wird).

PRINT - die nach dieser Anweisung stehende Information wird vom Programm auf den Bildschirm geschrieben.

FOR x = y TO z: ...: NEXT x - eine sog. Schleife, d.h. die Anweisungen zwischen FOR und NEXT werden mehrmals ausgeführt (wenn y 1 ist und z 5, dann 5 mal, aber der Anfangswert kann auch höher sein, z. B. 3, und dann wird es nur 3 mal wiederholt, wobei x in der ersten Runde 3, dann 4, und in der letzten 5 ist).

IF ... THEN - Verzweigung, d. h. in Abhängigkeit von bestimmten Bedingungen wird entschieden, welche Operation ausgeführt werden soll: die nach dem THEN, oder diejenige, die in der nächsten Zeile steht.

In den Programmen gibt es immer Wörter, deren Wert (Bedeutung) verändert werden kann; sie sind die Variablen. Eine Variable kann z.B.

für eine Zahl oder für eine Buchstabenkette stehen (im letzteren Fall muß am Ende des Variablensymbols in BASIC ein $-Zeichen stehen). Außerdem gibt es noch eine Reihe Symbole, die man Funktionen nennt. Für Linguisten sind diejenigen Funktionen besonders wichtig, die mit Zeichenketten, d.h. z.B. mit Wörtern, Morphemen, Sätzen usw. operieren können.

LEFT$ (a$,x) - bezeichnet x Buchstaben von der linken Seite des Wortes a$ (z. B. LEFT$ ("Haus", 2) bezeichnet "Ha").

RIGHT$ (a$,x) - bezeichnet x Buchstaben von der rechten Seite des Wortes a$ (z. B. RIGHT$ ("Haus", 2) bezeichnet "us").

MID$ (a$,x,y) - bezeichnet y Buchstaben, die auf der Position x beginnen (z. B. MID$ ("Haus", 3, 1) bezeichnet "u").

LEN (a$) - bezeichnet die Länge, d.h. die Zahl der Buchstaben im Wort a$.

INSTR (a$, b$) - gibt die Nummer der Position an, wo b$ im Wort a$ beginnt (wenn b$ im a$ nicht enthalten ist, wird die Nummer 0). Z.B. INSTR ("Haus", "au") ergibt 2.

So sieht z. B. ein Programm aus, das von drei eingegebenen Wörtern diejenigen auswählt, die auf -en auslauten:

```
FOR i = 1 TO 3
INPUT szo$(i)
NEXT i
FOR i = 1 TO 3
IF RIGHT$ (szo$(i), 2) = "en" THEN PRINT szo$(i)
NEXT i
END
```

Folgendes Programm, das vier eingegebene Wörter in rückläufige Reihenfolge ordnet, ist auch nicht viel komplizierter. Die Anweisung "STEP -1" bewirkt, daß die Schleifenvariable in jeder Runde nicht größer, sondern um 1 kleiner wird. Auf diese Weise kommen wir vom

letzten Buchstaben des Wortes schrittweise zum ersten, wodurch wir die Spiegelbilder der Wörter herstellen. Mit "+" können Zeichenketten zusammengefügt werden (hier wird das Spiegelbild jeweils um einen Buchstaben verlängert).
Mit Hilfe von ">" kann man die ASCII-Kodewerte der Buchstabenketten vergleichen, wobei immer das Wort "größer" ist, das in der alphabetischen Reihenfolge weiter hinten steht. In der zweiten Schleife, die in die erste eingebettet ist, werden Wörter vertauscht, wenn das gerade untersuchte Wort größer ist als ein von ihm rechts stehendes Wort. Nach dem Tausch beginnt die Prozedur von vorne, bis alle rechts stehenden Wörter größer sind als das untersuchte Wort. Anschließend wird das nächste rechts stehende Wort untersucht usw., bis alle 4 Wörter in der vorgesehenen Abfolge stehen.

```
FOR i = 1 TO 4
INPUT szo$(i)
FOR j = LEN(szo$(i)) TO 1 STEP -1
fszo$(i) = fszo$(i) + MID$(szo$(i), j, 1)
NEXT j
PRINT fszo$(i)
NEXT i
FOR a = 1 TO 3
FOR b = a + 1 TO 4
IF fszo$(a) > fszo$(b) THEN fszo$ = fszo$(b): szo$ = szo$(b):
fszo$(b) = fszo$(a): szo$(b) = szo$(a): fszo$(a) = fszo$ : szo$(a) =
szo$
NEXT b
NEXT a
FOR i = 1 TO 4
PRINT szo$(i)
NEXT i
END
```

Verweise

1. Siehe: McKeown, K. R., The TEXT System for Natural Language Generation: an Overview. In: ACL Proceedings, 20th Annual Meeting (1982), pp. 113-120

2. Siehe: Mauldin, M. L., Semantic-Rule Based Text Generation. In: COLING-84 (1984), pp. 376-380

3. Siehe: Kukich, K., Design and Implementation of a Knowledge-Based Report Generator. In: ACL Procedings, 21st Annual Meeting (1982), pp. 145-150

4. Siehe: Correira, A., Computing Story Trees. American Journal of Linguistics 6 (1980), pp. 135-149

5. Erschienen unter dem Titel "Speichern von Sätzen und Texten in Computer-Lernprogrammen", in: Germanistisches Jahrbuch DDR-UVR, Budapest, 1989

6. Siehe auch: Speichern von Sätzen und Texten in Computer- Lernprogrammen. In: Germanistisches Jahrbuch DDR-UVR, Budapest, 1989, 285-293

7. Siehe auch: P. Uzonyi, Sprachspezifische Übungen im computergestützten Unterricht, in: Unser Thema 7., Lektorat für dt. Spr. u. Lit., Budapest, 1990

8. Siehe auch: P. Uzonyi, Von der rechnergestützten menschlichen Übersetzung zur menschengestützten Rechnerübersetzung. In: Klaudy, K. - Kohn, J. - Molnár, K. - Szalai, L. (Hgg.), Transferre necesse est. Aktuelle Fragen der Übersetzung. BDTF, Szombathely, 1993, pp. 249-256

Literatur

1. Bar-Hiller, Y., The Present Status of Automatic Translation of Languages. In: Alt, F.L.: Advances in Computers 1., New York, 1960, pp. 91-163

2. Bátori I. - H.J. Weber, (Hgg.), Neue Ansätze in Maschineller Sprachübersetzung: Wissensrepräsentation und Textbezug. Max Niemeyer Verlag, Tübingen, 1986

3. Bresnan, J. W. (Hrsg.), The Mental Representation of Grammatic Relations, MIT Press, Cambridge, Mass., 1981

4. Busemann, S., Probleme der automatischen Generierung deutscher Sprache. HAM-ANS Memo 8, Universität Hamburg, 1982

5. Chomsky, N., Syntactic Struktures. Mouton, The Hague, 1957

6. Chomsky, N., Aspects of the Theory of Syntax. MIT Press, Cambridge, Mass., 1965

7. Chomsky, N., Lectures on Government and Binding. Foris, Dordrecht, 1981

8. Correa, N., A Binding Rule for Government-binding Parsing. In: Proceedings of COLING '88, Budapest, 1988

9. Dragalin, A. - Hunyadi, L. - Uzonyi, P., On some practical issues of an interactive machine translation system with standardized texts. In: Mezhdunarodnyj seminar po mashinnomu perevodu, Tezisy dokladov, Moskva, 1989, 99-100

10. Engel, U., Syntax der deutschen Gegenwartssprache. Berlin, 1977

11. Fix, H. - Rothkegel, A. - Stegentritt, E. (Hgg.), Sprachen und Computer. Festschrift zum 75. Geburtstag von Hans Eggers. AQ-Verlag, Dudweiler, 1982

12. Franken, G., Das Einsteigerseminar. MS-DOS 6.0. BHV Verlag, Korschenbroich, 1993

13. Gazdar, G. - Klein, E. - Pullum, G. K. - Sag, I., Generalized Phrase Structure Grammar. Blackwell, Oxford, 1985

14. Grüner, M. - Hassert, T., Einfürung in den computergestützten Sprachunterricht. Langenscheidt, München, 1992

15. Haller, J., Ein Algorithmus zur Reduktion syntaktischer Homographien in einem Informationssystem. AQ-Verlag, Dudweiler, 1980

16. Hauenschild, Ch., KIT/NASEV oder die Problematik des Transfers bei der Maschinellen Übersetzung. In: Bátori, I. - Weber, H. J. (Hgg.), Neue Ansätze in Maschineller Sprachübersetzung: Wissensrepräsentation und Textbezug. M. Niemeyer Vlg., Tübingen, 1986, pp. 167-195

17. Helbig, G. - Buscha, J., Deutsche Grammatik. Enzyklopädie Vlg. Leipzig, 1972

18. Jékel, P. - Papp, F., Ady Endre összes költői műveinek fonémastatisztikája. Akadémiai Könyvkiadó, Budapest, 1974

19. Kay, M., Functional Unificational Grammar: A Formalism for Machine Translation. COLING '84, 1984, pp. 75-78

20. Kiefer, F. (Hrsg.), Strukturális magyar nyelvtan 1. (Mondattan) Akadémiai Kiadó, Budapest, 1992

21. Klein, U., (Hrsg.), Stukturen und Verfahren in der maschinellen Sprachverarbeitung. AQ-Verlag, Dudweiler, 1985

22. Koskenniemi, K., A General Computational Model for Word-Form Recognition and Production. In: Proceedings of Coling 84: 22nd ACL, 1984

23. Lakoff, G., Instrumental Adverbs and the Concept of Deep Structure. Foundations of Language, 4, 1968, p. 429

24. Lamb, S. M., An Outline of Stratificational Grammar. Georgetown University Press, Washington, D. C., 1966

25. Luckhard, H. D. - Zimmermann, H. H., Computergestützte und maschinelle Übersetzung (Praktische Anwendungen und angewandte Forschung). AQ-Verlag, Saarbrücken, 1991

26. Mater, E., Rückläufiges Wörterbuch der deutschen Gegenwartssprache. Bibliographisches Institut, Leipzig, 1982

27. Muthmann, G., Rückläufiges deutsches Wörterbuch. M. Niemeyer Verlag, Tübingen, 1991

28. Ortmann, W. D., Wortbildung und Morphemstruktur hochfrequenter deutscher Wortformen, I-II. Goethe Institut, München, 1985

29. Papp, F. (szerk.), A magyar nyelv szóvégmutató szótára. Akadémiai Kiadó, Budapest, 1969

30. Prószéky, G., Számítógépes nyelvészet, SZÁMALK, Budapest, 1989

31. Rohrer, Ch., Maschinelle Übersetzung mit Unifikationsgrammatiken. In: Bátori, I. - Weber, H. J. (Hgg.), Neue Ansätze in Maschineller Sprachübersetzung: Wissensrepräsentation und Textbezug. M. Niemeyer Vlg., Tübingen, 1986, pp. 75-100

32. Rüschoff, B., Fremdsprachenunterricht mit computergestützten Materialien (Didaktische Überlegungen und Beispiele). Max Hueber Verlag, München, 1988 (2. Auflage)

33. Tarvainen, K., Einführung in die Dependenzgrammatik. M. Niemeyer Vlg, Tübingen, 1981

34. Tesniere, L., Esquisse d'une syntaxe structurale. Paris, 1953

35. Tesnière, L., Eléments de syntaxe structurale. Klincksieck, Paris, 1959

36. Uszkoreit, H., Problematische Konstruktionen für kontextfreie Phrasenstrukturgrammatiken. In: Klenk, U. (Hrsg.), Strukturen und Verfahren in der maschinellen Sprachverarbeitung. AQ-Vlg. Dudweiler, 1985, pp. 129-151

37. Uzonyi, P., Gépi szóképzési elemzés. In: Számítógép és nyelvoktatás I., VEAB értesítője (Hrsg.: F. Papp), 1983, pp. 85-86

38. Uzonyi, P., Német nyelvi oktatóprogramok és egy magyar képzett szóalakokat elemző program. In: Számítógép és nyelvoktatás, II., VEAB értesítője (Hrsg.: F. Papp), 1984, pp. 73-78

39. Uzonyi, P., Morfémikai szabálytípusok - Műhelymunkák a nyelvészet és társtudományai köréből, II. sz. (1986), pp. 133-152

40. Uzonyi, P., Combinatorial phrase-generation: a new dimension for CALL. In: Linguistics and Methodology in CALL (ed. by I. Kecskés and F. Papp), Budapest, 1986, pp. 150-153

41. Uzonyi P., Speichern von Sätzen und Texten in Computer-Lernprogrammen. In: Germanistisches Jahrbuch DDR-UVR, Budapest, 1989, pp. 285-293

42. Uzonyi, P., On text-generation in CALL-programs. In: New Tendencies in CALL (ed. by I. Kecskés and L. Agócs), Debrecen, 1989, pp. 92-94

43. Uzonyi, P., Sprachspezifische Übungen im computergestützten Unterricht. In: Unser Thema 7, Lektorat für dt. Spr. u. Lit., Budapest, 1990, pp. 37-40

44. Uzonyi, P., O morfologicheskoj kompetencii. In: Hunyadi-Klaudy-Lengyel-Székely (Hgg.), Könyv Papp Ferencnek, KLTE, Debrecen, 1991, pp. 233-239

45. Uzonyi, P., Számítógépes szótárak és egyéb nyelvi adatbázisok. KLTE, Debrecen, 1991

46. Uzonyi, P., Szótárbővítés interaktív fordítói rendszerben. In: Első Magyar Alkalmazott Nyelvészeti Konferencia, Nyíregyháza, 1991, pp. 672-675

47. Uzonyi, P., Von der rechnergestützten menschlichen Übersetzung zur menschengestützten Rechnerübersetzung. In: Klaudy, K. - Kohn, J. - Molnár, K. - Szalai, L. (Hgg.), Transferre necesse est. Aktuelle Fragen der Übersetzung. BDTF, Szombathely, 1993, pp.
2 249-256

48. Wahlster, W., Natürlichsprachliche Systeme. Eine Einführung in die sprachorientierte KI-Forschung. In: Bibel, W. - Sickmann, J. H. (Hgg.), Künstliche Intelligenz. Springer-Verlag Berlin, 1982

49. Wehrli, E., A Government-binding Parser for French. Institut pour les Etudes Semantiques et Cognitives, Université de Geneve. Working Paper No. 48., 1984

50. Weisweber, W., Ein Dominanz-Chart-Parser für generalisierte Phrasenstrukturgrammatiken. KIT-Report 45, Technische Universität, Berlin, 1987

51. Mc-Gee Wood, M. - Chandler, B. J., Machine translation for Monolinguals. In: Proceedings of the 12th International Conference on Computational Linguistics, Budapest, 1988, pp. 760-763

52. Woods, W. A., Transition Network Grammars for Natural Language Analysis. In: CACM, 13 (10), 1970, pp. 591-606

53. Zaliznjak, A. A., Grammaticheskij slovar russkogo jazyka. Russkij jazyk, Moskau, 1977.

54. Proudian, D. - Pollard, C.J., Parsing Head-Driven Phrase Structure Grammar. ACL Proceedings, 23rd Annual Meeting, 1985, pp. 167-171

55. Paulus, D., Endliche Automaten zur Verbflexion und ein spezielles deutsches Verblexikon. In: Morik, K. (Hrsg.), GWAI-87 - German Workshop on Artificial Intelligence. Proceedings, Springer, Berlin, 1987, pp. 340-344