wissenschaften in einen anderen Rahmen: nicht die gegenseitige Abgrenzung, sondern der *wechselseitige Austausch von* Ergebnissen und Erfahrungen stehen im Vordergrund. Er verbindet diejenigen Disziplinbereiche mit der CL, die — wenn auch mit unterschiedlicher Ausrichtung und Zielsetzung — diesem Forschungsparadigma verpflichtet sind.

6. Literatur (in Auswahl)

Barr/Feigenbaum/Cohen 1981 · Bátori 1977a · Bátori 1981 a · Bátori 1982 b · Bátori 1986 · Cercone/McCalla 1986 · Christaller/Metzing 1979/1980 · von Hahn 1978 · Hayes 1984 · Hays 1976 · Hess/Brustkern/Lenders 1983 · King 1983 · Krallmann 1968 · Lenders 1980 · Lutz/Schmidt 1982 · Metzing 1982 · Palmer 1984 · Papp/Szepe 1976 · Schnelle 1982 · Straßner 1977 · Ungeheuer 1971 · Wahlster 1982 b · Winograd 1983.

Dieter Krallmann, Essen
(Bundesrepublik Deutschland)

6. Computerlinguistik und die Theorie der formalen Sprachen

1. Einleitung

Gegenstand der Linguistik ist die Beschreibung der natürlichen Sprachen unter verschiedensten Aspekten (synchronen, diachronen u. a.), Aufgabenbereich der Computerlinguistik sind die maschinelle Analyse natürlich-sprachlicher Texte und die Simulation der Sprachanwendung. Die Computerlinguistik stützt sich dabei auf linguistische Beschreibungen, die unter Zugrundelegung mathematischer Modelle formalisiert sind. Die Frage ist nun, welche Rolle die Theorie der formalen Sprachen, die eine sehr allgemeine mathematische Theorie ist, in diesem Bezugsrahmen spielt.

Die Verbindung von Computerlinguistik, Theorie der formalen Sprachen und Gebieten der Linguistik, besonders der Syntax, erklärt sich zum einen durch die Entwicklung dieser Disziplinen, die entscheidend durch Noam Chomsky geprägt wurde. (Siehe unten 2.) Zum anderen ist die Theorie der formalen Sprachen von der Linguistik und der Computerlinguistik als empirischer bzw. angewandter Wissenschaft a priori unabhängig. Daher ist zu fragen, was die Motive dafür sind, daß sie als eine für diese Gebiete so wichtige Theorie angesehen wird, insbesondere inwiefern sie für Fortschritte in der Computerlinguistik wesentlich ist. Auch hierzu in 2. einige allgemeine Gesichtspunkte. In 3. wird die immer wieder gestellte Frage behandelt, welcher formalsprachliche Typus für die Beschreibung natürlicher Sprachen am besten zugrundezulegen ist. Die Antwort(en) hierauf (in 4.) ist (sind) wichtig für die Konstruktion computerlinguistischer Systeme, da sich daraus Konsequenzen für die Lösbarkeit vieler Probleme und die Komplexität von Verfahren ergeben, d. h. Konsequenzen für die praktische Entwicklung und Ausführbarkeit von Programmen.

2. Linguistik, Theorie der formalen Sprachen und Computerlinguistik

2.1. Entwicklung

Die Theorie der formalen Sprachen entstand in enger Verbindung mit demjenigen Zweig der Linguistik, der unter den Namen 'generative Grammatik' bekannt ist. Der Begründer beider ist Noam Chomsky (Chomsky 1956; 1957; 1959).

In der Linguistik setzt Chomsky an Positionen des amerikanischen Strukturalismus an, die er in neuer Weise interpretiert und weiterentwickelt. Dazu gehören die Konstituentenanalyse (Bloomfield 1933; Harris 1951) und der seit Harris (1952) auftretende Begriff der syntaktischen Transformation. Die Theorie der formalen Sprachen, die nun als mathematische Metatheorie für linguistische Beschreibungen entsteht, orientiert sich an Theorien über Zeichensysteme allgemeinster Art, die aus der Logik und Mathematik bekannt sind, — es sei besonders auf die Arbeiten von Thue, Turing, Kleene und Post hin-

gewiesen (vgl. 5.). Ausschlaggebend für die linguistische Modellbildung erweist sich die Übernahme des Rekursivitätsbegriffs (Bar-Hillel 1953).

Neu ist a) das Konzept der generativen Syntax als Beschreibungsmodell für natürlich-sprachliche Satzmengen, welches dann zu dem der Transformationsgrammatik erweitert wird (Chomsky 1957; 1965), und b) die mathematische Theorie der generativen Syntaxen und der von ihnen erzeugten Sprachen, auch Typ-i-Sprachen ($0 \leq i \leq 3$) genannt, die den Kern der Theorie der formalen Sprachen bildet. Die Theorie geht von folgenden grundlegenden Annahmen aus: (a) eine *formale Sprache L* ist eine beliebige Menge von Zeichenketten, konstruiert über einem endlichen Inventar von Grundzeichen. Dabei besteht jede Kette aus endlich vielen (als Spezialfall 0) Vorkommen eindeutig identifizierbarer Grundzeichen. Definitionsgemäß gibt es endliche und unendliche formale Sprachen. Sätze einer natürlichen Sprache können als Zeichenketten betrachtet werden, deren Grundzeichen je nach Beschreibungsebene Phoneme, Grapheme, Morpheme oder Wörter sind. Mengen natürlichsprachlicher Sätze sind unter diesem Aspekt formale Sprachen.

(b) Eine *Syntax G für eine formale Sprache L* legt eine berechenbare Funktion fest, deren Bildmenge genau L ist. Gibt es für gegebenes L ein solches G, so kann L mittels G aufgezählt werden. L ist somit eine aufzählbare Menge. (Es gibt allerdings auch nicht-aufzählbare formale Sprachen, d. h. solche, für die kein G exisitiert, vgl. Hermes 1961, 11 ff.) (c) Als Formalismus für G werden die generativen Syntaxen eingeführt. Diese beruhen auf dem Konzept der Wiederschreib-Regel (rewriting rule), das auf Thue (1914) zurückgeht. Eine *Wiederschreibregel* ist eine Anweisung, eine Zeichenkette z_1 durch eine Zeichenkette z_2 zu ersetzen (= wiederzuschreiben). Eine Kette uz_1v wird dadurch in uz_2v umgewandelt. Eine *generative Syntax G* ist durch zwei disjunkte Zeicheninventare, die *Kategorien-* und die *Endsymbole,* und eine darüber gebildete Menge von Wiederschreib-Regeln festgelegt, durch die beginnend bei einem ausgezeichneten Kategoriensymbol (wie z. B. SATZ) Zeichenketten abgeleitet werden. Die Menge der so erhaltenen Ketten aus Endsymbolen ist *die von G erzeugte Sprache* , notiert L(G). (Für eine Syntax G einer natürlichen Sprache ist L(G) die Menge der Sätze dieser Sprache.) Es werden Typen generativer Syntaxen unterschieden, abhängig von der Form der Regeln: 1) die Gesamtmenge der generativen Syntaxen, auch *Typ-0-Syntaxen* genannt, 2) die *Typ-1-Syntaxen* (mit der Variante der *kontextsensitiven Syntaxen*), bei denen abgeleitete Ketten nicht kürzer als die Ausgangsketten sind, 3) die *kontextfreien* oder *Typ-2-Syntaxen,* bei denen pro Regel genau ein Kategoriensymbol ersetzt wird, 4) die *Typ-3-Syntaxen,* ein Spezialfall der Typ-2-Syntaxen, bei denen die Ersetzung jedesmal nur am Ende (bzw. nur am Anfang) der wiederzuschreibenden Kette erfolgt. Eine *Typ-i-Sprache L* ($0 \leq i \leq 3$) ist eine formale Sprache, für die es eine Typ-i-Syntax G mit $L(G) = L$ gibt. Typ-1-Sprachen nennt man auch *kontextsensitiv,* Typ-2-Sprachen *kontextfrei* .

Die Typisierung der generativen Syntaxen erfolgt in Verbindung mit der Frage, mit welchem Formalismus natürliche Sprachen syntaktisch am besten zu beschreiben seien. Wir kommen in *Kapitel* 3 auf diese Frage zurück. Für die Entwicklung in der Linguistik ist die Annahme rekursiver Regelsysteme mit unendlicher generativer Kapazität von besonderem Gewicht. Probleme für die syntaktische Beschreibung bieten Einbettungs- und Anreihungsstrukturen, deren größtmögliche Komplexität sich schwer abschätzen läßt, wie bei Koordinationen, Nebensatzeinbettung u. a. Generative Syntaxen erlauben es, solche Fälle durch endliche Regelmengen, die unendlich viele Sätze erzeugen, zu erfassen.

Die Definition der Typen führt zur Untersuchung ihrer mathematischen Eigenschaften. Man stellt fest, daß die Sprachen eine Hierarchie bilden derart, daß jede Typ-i-Sprache mit $1 \leq i \leq 3$ eine Typ-(i-1)-Sprache ist, es aber Typ-(i-1)-Sprachen gibt, die nicht vom Typ-i sind (Beweise vgl. die Gesamtdarstellungen von Salomaa 1973 und Maurer 1977).

Um 1960 wird die Theorie der formalen Sprachen für die Informatik interessant, nachdem erstmals eine höhere Programmiersprache formal definiert wurde. Es ist ALGOL-60, festgelegt in der Backus-Naur-Form (Backus 1959). Diese erweist sich als eine Notationsvariante für kontextfreie Syntaxen, wodurch die Verbindung zur generativen Syntax hergestellt wird. Von nun an wird die Theorie der formalen Sprachen grundlegend für die Programmiersprachen, was sie zu einem Teilgebiet der Informatik werden läßt.

Es fällt auf, daß für einige Zeit in Lingui-

stik und Informatik verschiedene Problembereiche behandelt werden. In der Informatik wird a) die Theorie der formalen Sprachen ausgebaut und um eine Vielzahl neuer Syntaxtypen erweitert, b) die Automatentheorie entwickelt und dabei Äquivalenzbeziehungen zwischen Automaten- und Syntaxtypen gefunden, und werden c) Algorithmen für die syntaktische Analyse von Programmen (= Parser) entworfen. Die Linguistik dagegen konzentriert sich über Jahre auf die Transformationsgrammatik (TG). Es werden mathematische Theorien der TG formuliert (Ginsburg/Partee 1969; Peters/Ritchie 1973) und auch Syntaxanalyse mit TGen behandelt (Petrick 1965), doch wird die TG in der Informatik kaum beachtet. TGen sind überaus komplexe Systeme, die für natürliche Sprachen als adäquate Modelle angesehen werden, für Programmiersprachen erweisen sich einfachere Formalismen jedoch als ausreichend.

Ein Umschwung im natürlich-sprachlichen Bereich wird durch die sich entwikkelnde Computerlinguistik herbeigeführt. Ihre einschlägigen Anwendungen wie maschinelle Übersetzung und Frage-Antwort-Systeme erfordern eine Syntaxanalyse natürlich-sprachlicher Sätze (und Texte), wofür Verfahren zu finden sind. Wie in der Informatik spricht man hier von Parsern und benutzt zum Teil dieselben Parse-Methoden wie dort. Die TG erweist sich nun trotz einiger Versuche als wenig brauchbar, was an der Vielfalt ihrer formalen Mittel, deren Zusammenspiel undurchschaubar bleibt, liegt. Eine Alternative bieten die eigens für computerlinguistische Zwecke entwickelten erweiterten Netzwerkgrammatiken (Woods 1970), automatenartige Syntaxen, mit denen sich eine Syntaxanalyse leichter durchführen läßt. Sie werden einer der verbreitetsten Formalismen in der Computerlinguistik.

Als für die Computerlinguistik interessant haben sich die in den letzten Jahren aufgekommenen Unifikationsgrammatiken erwiesen, vornehmlich die generalisierte Phrasenstrukturgrammatik (GPSG, zuletzt Gazdar/Klein/Pullum/et al. 1985) und die lexikalisch-funktionale Grammatik (LFG, Bresnan/Kaplan 1982) (s. Art. 18). Im Gegensatz zur TG, die Oberflächen- und Tiefenstruktur eines Satzes unterscheidet und sie durch Transformation verbindet, beschreibt man hier auf der syntaktischen Ebene nur die Oberflächenstrukturen. Die semantischen bzw. funktionalen Strukturen werden parallel zu den syntaktischen entwickelt. GPSG und LFG sind verschiedene Formalismen, denen auch verschiedene Konzeptionen über natürliche Sprachen zugrundeliegen. Doch gibt es bestimmte Gemeinsamkeiten: man geht davon aus, daß jede natürliche Sprache im wesentlichen mit einer kontextfreien Syntax beschreibbar ist. Da aber eine solche Syntax Tausende von Regeln enthalten müßte, versucht man, durch Erweiterungen des Formalismus zu weniger komplexen Regelsystemen zu kommen. Solche Mittel sind Merkmalsmengen und Operationen darüber sowie Kontrollmechanismen für die Übertragung von Merkmalen von einem Satzteil auf andere. Die Verbindung zur Theorie der formalen Sprachen ergibt sich dadurch, daß diese Formalismen als Syntaxtypen mathematisch festzulegen sind. Erst dann läßt sich zeigen, von welchem Typ die erzeugten Sprachen sind, und können weitere formale Eigenschaften bewiesen werden. Hiermit verbundene Fragestellungen wurden in letzter Zeit behandelt, vgl. Uszkoreit/Peters (1984), Berwick (1984 a). Die Attraktivität von GPSG und LFG für die Computerlinguistik erklärt sich daraus, daß ihr kontextfreier Kern es erlaubt, für die Syntaxanalyse Algorithmen zu verwenden, die aus der Informatik bekannt sind.

2.2. Systematische Zusammenhänge

Die den computerlinguistischen Anwendungen zugrundegelegten linguistischen Beschreibungen müssen mathematisch formalisiert sein. Dies liegt daran, daß ein Computerprogramm ein mathematisches Verfahren ist und daher mathematisch definierte Strukturen als Eingabe erfordert. Die für das Verfahren definierten Eingaben bilden die Klasse der Probleme, die es bearbeitet. So soll ein Parser P für einen gegebenen Satz s bzgl. einer gegebenen Syntax G eine Aufbaustruktur zu finden. Alle Paare (G, s) bilden die Problemklasse von P. Dabei sind formale Einschränkungen zu machen, etwa daß G zu einem bestimmten Syntaxtyp gehören muß, damit P anwendbar wird. So bearbeitet z. B. der bekannte Earley-Parser nur kontextfreie Syntaxen.

Der systematische Zusammenhang zwischen linguistischer Syntax, Theorie der formalen Sprachen und Computerlinguistik stellt sich nun folgendermaßen dar: a) in der Linguistik werden Formalismen für die syntaktische Beschreibung natürlicher Sprachen entwickelt, b) in der Theorie der formalen Sprachen werden die mathematischen Eigen-

schaften dieser Formalismen untersucht, c) in der Computerlinguistik entwirf man ausgehend von den linguistischen Modellen und der Theorie der formalen Sprachen Verfahren für die Sprachanalyse und -synthese

Im Mittelpunkt der Theorie der formalen Sprachen stehen die Typ-i-Sprachen ($0 \leq i \leq 3$) und -syntaxen. Obwohl man in der Linguistik und auch der Informatik meistens keine reinen Typ-i-Syntaxen verwendet, sondern Erweiterungen anfügt oder ganz andere Formalismen wie Automaten benutzt, ist die Hierarchie der Typ-i-Sprachen für theoretische Betrachtungen wichtig, da sie als Bezugssystem dient. Auch wenn man einen anderen formalen Typ vorzieht, möchte man wissen, wie groß die von ihm erfaßte Sprachenklasse ist, wobei man sich gut an der Typ-i-Hierarchie orientieren kann. Sei K eine solche Klasse. Man sucht dann das größte i, für das alle Sprachen in K vom Typ i sind. Die Bedeutung dieser Hierarchie rührt daher, daß sie unter vielen Aspekten erforscht ist und man viele allgemeine Eigenschaften der Sprachen von gegebenem Typ i und wichtige Unterschiede zwischen Sprachen, die verschiedenen Typen i angehören, kennt. Für Verfahren in der Computerlinguistik kommen solche Ergebnisse zum Tragen, wie in 4. ausgeführt wird. Daß eine fundamentale Klasseneinteilung vorliegt, zeigt sich besonders an einer Äquivalenz zwischen Sprachen- und Automatentypen: für jedes i gibt es einen Automatentyp A, für den gilt: jede Typ-i-Sprache wird von einem Automaten des Typs A akzeptiert und jeder solche Automat akzeptiert eine Typ-i-Sprache. In diesem Sinne sind folgende Sprach- und Automatentypen äquivalent: Typ-0-Sprachen und Turingmaschinen, Typ-1-Sprachen und linear beschränkte Automaten, Typ-2-Sprachen und Push-Down-Automaten, Typ-3-Sprachen und endliche Automaten. Die Bedeutung der Typ-i-Hierarchie wird ferner durch Abgeschlossenheitseigenschaften unterstrichen: bestimmte mengentheoretische Operationen über beliebigen Sprachen desselben Typs i ergeben wieder eine Typ-i-Sprache. (Zu sämtlichen Sätzen und Beweisen vgl. Salomaa 1973, Maurer 1977).

3. Natürliche Sprachen und formale Sprachen

Für die Behandlung der Frage, welche Syntaxtypen für natürliche Sprachen zugrunde zu legen sind, gibt es zwei methodische Ansatzpunkte: eine Theorie ist zu verwerfen, a) wenn sie nicht mächtig genug ist, allen Beobachtungsdaten gerecht zu werden, und b) wenn sie zu mächtig ist, so daß sie unmögliche Fälle miterfaßt.

Ansatz a): Chomsky (1957) wendet sich gegen Typ-3-Syntaxen (finite state grammars), da in natürlichen Sprachen symmetrische Strukturen der Form $a_1 ... a_n c b_n ... b_1$ ($n \geq 1$) vorkommen (z. B. bei Satzeinbettungen), worin für jedes i ($1 \leq i \leq n$) a_i und b_i voneinander abhängen und zu denen es keine andere Struktur gibt, in der entweder ein a_i oder ein b_i fehlt und sonst alles unverändert bleibt. Unter der Annahme, daß n beliebig groß werden kann, liegt keine Typ-3-Sprache vor, woraus sich die Inadäquatheit einer Typ-3-Syntax ergibt. Die nächste Frage ist, ob man mit Typ-2-Syntaxen auskommt. Sie wurde öfter verneint, doch ist die Lage weniger klar als beim Typ 3. Bekannt ist die Argumentation von Bar-Hillel/Shamir (1961) zu *respectively* und von Postal (1964) zu Nominalisierungen im Mohawk. Hier gibt es Überkreuz-Konstruktionen der Form axbyc mit $x = d_1 ... d_n$ und $y = e_1 ... e_n$ ($n = 1, 2, ...$), worin sich jeweils d_i und e_i aufeinander beziehen. Es wird behauptet, daß, da gleichviel d's und e's vorhanden sein müssen, Englisch und Mohawk nicht kontextfrei sind. Pullum/Gazdar (1982) entkräften diese und weitere Argumente empirisch und teils auch mathematisch. Doch werden neue gefunden. So hält Higginbotham (1984) Englisch wegen Strukturen in *such-that-* Sätzen für nicht kontextfrei, Postal/Langendoen (1984) versuchen dasselbe an Hand von elliptischen Konstruktionen (sluicing clauses) zu zeigen. Doch sind auch diese Argumente empirisch nicht stichhaltig, wie Pullum (1984 a) nachweist. Shieber (1985) behauptet die Nicht-Kontextfreiheit des Schweizerdeutschen an Hand dort auftretender Strukturen in Infinitivkonstruktionen. Zu fragen ist jedoch, inwieweit diese von Schweizerdeutschen als korrekt empfunden werden. Noch ein Ergebnis: nach Culy (1985) soll das Vokabular des in Mali gesprochenen Bambara nicht kontextfrei sein. — Als Fazit kann man festhalten, daß die Argumente gegen die Kontextfreiheit natürlicher Sprachen wenig überzeugend sind.

Ansatz b): Hier geht man von der Menge aller formalen Sprachen aus und fragt, welche Einschränkungen mindestens zu machen sind, damit eine Theorie natürlicher Sprachen unnatürliche Strukturen ausschließt. Die klassische Frage ist: Sind natürliche

Sprachen notwendigerweise entscheidbar oder nur rekursiv aufzählbar oder nicht einmal das? (Die Menge der rekursiv aufzählbaren Sprachen ist gleich der Menge der Typ-0-Sprachen, die entscheidbaren bilden eine echte Teilmenge davon, vgl. Salomaa 1973 und s. 4.) Als Argument für die Entscheidbarkeit wird angeführt, daß ein Mensch normalerweise jede Wortfolge als korrekten oder nicht korrekten, abweichenden oder nicht abweichenden Satz seiner Muttersprache klassifizieren kann, er somit über ein Entscheidungsverfahren dafür verfügt. Überlegungen hierzu finden sich in Putnam (1961). — Peters/Ritchie (1973) beweisen, daß die Menge der Sprachen, die von Transformationsgrammatiken erzeugt werden, die Menge der Typ-0-Sprachen ist. Es gibt somit nicht-entscheidbare darunter. Zugrundegelegt ist dabei das Aspects-Modell (Chomsky 1965). Nimmt man die Entscheidbarkeit natürlicher Sprachen an, sind für ihre Beschreibung allenfalls Transformationsgrammatiken, die formal in geeigneter Weise eingeschränkt sind, adäquat. Peters/Ritchie (1973) finden eine Bedingung, die Transformationsregeln erfüllen müssen, damit die erzeugte Sprache entscheidbar ist, und Peters (1973) zeigt, daß die für natürliche Sprachen bekannten Transformationsregeln diese Bedingungen erfüllen. — Eine andere Argumentation geht davon aus, daß nicht Entscheidbarkeit sondern Erlernbarkeit das ausschlaggebende Kriterium für natürliche Sprachen sei. Diesen Standpunkt nimmt Chomsky (1980 b) ein. Er kritisiert Levelt (1974), der ausgehend von einer Präzisierung des Erlernbarkeitsbegriffs zeigt, daß nur entscheidbare Sprachen erlernbar sind. Chomsky hält Levelts Präzisierung für inadäquat und führt Argumente dafür an, daß erlernbare Sprachen nicht notwendigerweise entscheidbar sind. Was die existierenden natürlichen Sprachen betrifft, so scheinen sie auf Grund von Peters (1973) tatsächlich entscheidbar zu sein.

Die gesamte Diskussion, die heute andauert, fordert zu einer kritischen Anmerkung dazu heraus, wie sie geführt wird. So fallen Aussagen der Art auf, daß natürliche Sprachen von einem bestimmten Typ i sind, daß sie unendliche Satzmengen sind u. a. Das Problem ist, wie man hier mathematische Begriffe auf einen nicht mathematischen Objektbereich anwendet. Man hat häufig den Eindruck, daß der zu erforschende Gegenstand, die Sprache, und das Beschreibungsmodell nicht klar unterschieden werden. Die Frage ist immer, welches Modell die Beobachtungsdaten am besten erfaßt. Dabei sind Idealisierungen aus Gründen der Beschreibungsökonomie und der Abgrenzung erforderlich. Die verwendeten mathematischen Begriffe beinhalten solche Idealisierungen. Eine solche liegt vor, wenn für natürliche Sprachen Syntaxen mit unendlicher generativer Kapazität angenommen werden: rekursive Regeln werden eingeführt, um die Beschreibung einfach zu halten (Chomsky 1956). Ohne dieses Mittel würden Syntaxen für natürliche Sprachen unüberschaubar komplex, es würde sogar unmöglich, sie zu schreiben, da man die maximale Länge natürlich-sprachlicher Sätze nicht kennt. Allen Charakterisierungen natürlicher Sprachen als nicht vom Typ 3 liegt dieses Unendlichkeitspostulat zugrunde (alle endlichen Sprachen sind vom Typ 3). Es sind aber noch weitere Idealisierungen erforderlich, da a) natürliche Sprachen nicht homogen sind sondern aus einer Vielzahl von Subsystemen, Dialekten, bestehen und b) der Formenreichtum auch innerhalb der Subsysteme so groß ist, daß Theorien leicht durch Gegenbeispiele widerlegt werden können. Eindrucksvoll zeigt dies die Diskussion um die Nicht-Kontextfreiheit natürlicher Sprachen. Um eine Menge von Sätzen bzgl. der Typ-i-Hierarchie oder anderer mathematischer Typen klassifizieren zu können, muß sie so durch Abgrenzungskriterien festgelegt sein, daß man eindeutig bestimmen kann, was dazu gehört und was nicht. So ist z. B. zu fragen, was auf Satzebene beschrieben werden soll und wo bereits satzübergreifende Textstrukturen vorliegen, etwa im Hinblick auf koordinative Verknüpfungen.

Zu erwähnen ist noch, daß auch die Frage, ob nicht doch Typ-3-Syntaxen adäquate Modelle für natürliche Sprachen sind, diskutiert wird. Solche Modelle werden u. a. in der Neurolinguistik benutzt. Einen Überblick gibt Kornai (1985).

Die bisher referierte Diskussion betrachtet Sprachen nur als Mengen von Ketten. In der Linguistik und Computerlinguistik interessiert man sich aber besonders für die Sätzen unterliegenden hierarchischen Strukturen, die z. B. in der Form von Bäumen oder Charts dargestellt werden. Sie bilden die Basis für weitere Untersuchungen, z. B. semantischer Art. Daher bevorzugt man Syntaxen, die den Sätzen geeignete Strukturen zuordnen. So kann es sein, daß man, auch wenn man weiß, daß eine zu beschreibende Satz-

menge theoretisch von einer kontextfreien Syntax erzeugt wird, lieber einen anderen Formalismus, etwa einen mit Transformationsregeln, benutzt, wenn er sich dem Beschreibungsgegenstand als angemessener erweist.

4. Lösbarkeit und Komplexität

Für die maschinelle Sprachanalyse und -synthese werden Verfahren benötigt. Was für welche möglich und praktikabel sind, hängt vom zugrundegelegten Sprachbeschreibungsformalismus ab. Aus der Theorie der formalen Sprachen sind hierfür Theoreme über die Lösbarkeit von Problemen und aus der Algorithmentheorie Theoreme über die Komplexität von Verfahren maßgebend.

(a) Lösbarkeit: Für eine Reihe von Problemen wurde bewiesen, daß sie im mathematischen Sinn unlösbar sind, d. h. es kein mathematisches Verfahren zu ihrer Lösung gibt (zum Begriff des *mathematischen Verfahrens* oder *Algorithmus* vgl. Hermes 1961). Statt *mathematisch lösbar/unlösbar* wird hinfort einfach *lösbar/unlösbar* gesagt, da hier nur Algorithmen betrachtet werden. Für eine Klasse K von formalen Sprachen gilt ein Problem P als lösbar, genau dann wenn P für jedes Element von K lösbar ist, andernfalls ist P für K unlösbar (d. h. ist P für K unlösbar, dann ist P für wenigstens 1 Element von K unlösbar, kann aber für andere Elemente von K lösbar sein). — Einer der grundlegenden Sätze ist der über die Unlösbarkeit des Element-Problems für Typ-0-Sprachen: es gibt Paare (L, k) mit L einer Typ-0-Sprache und k einer Zeichenkette über dem Vokabular von L, für die es unlösbar ist, ob k Element von L ist (vgl. Salomaa 1973, Maurer 1977). — Mit dem Begriff der Lösbarkeit von Problemen tritt der der Entscheidbarkeit von Mengen auf. Sei M Teilmenge einer Menge N. M heißt *entscheidbar relativ zu N* , genau dann wenn für jedes Element von N lösbar ist, ob es aus M ist oder nicht. Aus der Unlösbarkeit des Element-Problems für Typ-0-Sprachen folgt, daß es Typ-0-Sprachen gibt, die relativ zur Menge aller über ihrem Vokabular bildbaren Zeichenketten nicht entscheidbar sind.

Sätze wie der genannte stecken theoretische Grenzen der computerlinguistischen Arbeit ab. So wird man es vermeiden, einen Syntaxtyp. dessen Sprachenmenge die aller Typ-0-Sprachen ist, zu wählen, da es kein generelles Analyseverfahren gibt, das für eine beliebige Syntax G dieses Typs und einen beliebigen Satz s entscheidet, ob s von G erzeugt wird. — Für die Typ-1-Sprachen ist das Element-Problem lösbar, und für jede Typ-1-Syntax kann prinzipiell entschieden werden, ob sie einen vorgelegten Satz erzeugt. Es gibt aber auch Typ-0-Sprachen, die nicht vom Typ 1 und trotzdem entscheidbar sind. Doch ist wenig über ihre mathematischen Eigenschaften bekannt, so daß sich kaum mit ihnen operieren läßt. Die Einschränkung auf Typ 1 erscheint für praktische Anwendungen mindestens erforderlich.

Eine Übersicht über weitere unlösbare Probleme in der Theorie der formalen Sprachen gibt Salomaa (1973). Viele davon lassen sich auf das Element-Problem für Typ-0-Sprachen zurückführen. Für die kontextfreien Syntaxen und Sprachen sind besonders folgende zwei unlösbare Probleme wichtig: (1) Das Äquivalenzproblem: Es gibt Typ-i-Syntaxen G_1 und G_2 mit $i \geq 2$, für die es unlösbar ist, ob sie dieselbe Sprache erzeugen. (2) Das Mehrdeutigkeitsproblem: a) Es gibt kontextfreie Syntaxen G, für die es unlösbar ist, ob sie ambig sind, d. h. ob es in der von G erzeugten Sprache Ketten gibt, denen durch G verschiedene Strukturbäume zugeordnet sind. b) Es gibt kontextfreie Sprachen, für die es unlösbar ist, ob sie ambig sind. Dabei heißt eine kontextfreie Sprache *ambig,* wenn jede kontextfreie Syntax, die sie erzeugt, ambig ist. (b) Komplexität: Hier geht es um die Frage, wie schwierig es ist, ein gegebenes Problem zu lösen. Wichtige Komplexitätsmaße sind Zeit- und Platzbedarf für den Ablauf eines Algorithmus zur Lösung eines einschlägigen Problems. Diese werden im Folgenden betrachtet. Der *Zeitbedarf* ist durch die Anzahl der erforderlichen Rechenschritte, der *Platzbedarf* durch die der benötigten Plätze auf einem Speicherband gegeben. Eine Präzisierung der hier auftretenden Konzepte wie *Rechenschritt* u. a. erfolgt in der Theorie der Turingmaschinen (vgl. Hermes 1961, Paul 1978).

Zeit- und Platzbedarf wurden für Parser untersucht. Ein Parser P besteht aus einem Erkennungsalgorithmus, der für eine beliebige Syntax G gegebenen Typs und eine beliebige Zeichenkette k entscheidet, ob k von G erzeugt wird, und aus einem weiteren Algorithmus, der für erkanntes k eine Aufbaustruktur findet. Zeit- und Platzbedarf für P bzgl. G geben an, wieviele Rechenschritte bzw. Speicherplätze zur Lösung dieser Probleme benötigt werden. Es sind Funktionen, deren Parameter die Länge von k (= die An-

zahl der Zeichenvorkommen in k) ist. Sei diese n, der Zeitbedarf t und der Platzbedarf p. Bei der Festlegung von t und p geht es darum, das Minimum an Bedarf zu finden, so daß t(n) und p(n) für jedes k der Länge n ausreichen. Da P auf jede Syntax eines gegebenen Typs anwendbar ist, interessiert man sich für gemeinsame Eigenschaften aller t bzw. p, die durch P für diese Syntaxen gegeben sind. Auf diese Weise erhält man den *Zeit-* und *Platzbedarf für P* in Form eines Schemas für Funktionen.

Die wichtige Frage ist hier, wie sich eine Bedarfsfunktion f mit wachsendem n entwikkelt. Man sagt „f(n) ist 0(g(n))“, wenn es eine natürliche Zahl c und eine Funktion g gibt, so daß für alle natürlichen Zahlen $n \geq 0$ gilt: $f(n) \leq c \cdot g(n)$. Sagt man z. B. daß ein Algorithmus A einen Zeitbedarf von $0(n^2)$ hat, so meint man damit, daß es ein c gibt, so daß A für eine beliebige Eingabe der Länge n nicht mehr als $c \cdot n^2$ Rechenschritte benötigt. Für praktische Anwendungen wichtig ist, ob g ein Polynom sein kann, d. h. ob es eine natürliche Zahl k gibt, für die $f(n) \leq 0(n^k)$ gilt. Weiter möchte man wissen, ob sogar lineares g möglich ist (d. h. mit k = 1). Gibt es kein Polynom g, wie bei exponentiell wachsendem Bedarf ($g(n) = k^{0(n)}$ mit $k \geq 2$), ist praktisches Rechnen nur noch mit kleinem n ausführbar, da sonst bald die vorhandenen zeitlichen oder räumlichen Möglichkeiten nicht mehr ausreichen. Aber auch bei polynomiellem Bedarf wird die praktische Ausführbarkeit schnell eine Grenze erreichen, wenn k nicht klein ist. Man wird daher Algorithmen mit linearem g zu finden versuchen.

Kehren wir zu Parsern zurück. Das Problem bei natürlichen Sprachen ist, daß bzgl. der hierfür interessanten Syntaxtypen Zeit- und Platzbedarf von Erkennungsalgorithmen zumeist unerträglich hoch werden. Die Komplexität hängt direkt vom Algorithmus ab und indirekt vom Syntaxtyp, da dieser die möglichen Algorithmen bestimmt. So hat man für die kontextfreien Syntaxen, der am besten untersuchten Klasse, folgende Ergebnisse (zu den Algorithmen siehe noch Art. 32): 1) Sowohl für die Top-Down- als auch die Bottom-Up-Analyse mit Backtrakking ist ein Platzbedarf von 0(n) und ein Zeitbedarf von $k^{0(n)}$ ($k \geq 2$) anzusetzen. 2) Die tabellarischen Methoden von Cocke-Younger-Kasami und Earley erfordern einen Platzbedarf von $0(n^2)$ und einen Zeitbedarf von $0(n^3)$, mit nicht-ambigen Syntaxen benötigt der Earley-Algorithmus eine Zeit von $0(n^2)$ (vgl. Aho/Ulmann 1972, I, 281 ff.). Man muß aber berücksichtigen, daß jeder dieser Algorithmen auf alle Syntaxen einer großen Teilmenge der kontextfreien anwendbar ist (beim Earley-Algorithmus alle kontextfreien) und die Bedarfsangaben für die „ungünstigsten“ darunter gelten. Zeit- und Platzbedarf vieler speziellerer Syntaxen sind nur linear.

Eine Aufgabe der Computerlinguistik ist es, für natürliche Sprachen hinreichend adäquate Syntaxen zu finden, für die es Erkennungsalgorithmen mit geringem Zeit- und Platzbedarf gibt. Nur lineare Zeit benötigen die LL(k)- (left-to-right leftmost derivation) und LR(k)-Syntaxen (left-to-right rightmost derivation) (Aho/Ullman 1972—73), die spezielle kontextfreie sind. Inwieweit sie für linguistische Beschreibungen ausreichen, ist eine offene Frage. — Der innerhalb der GPSG entwickelte ID/LP-Formalismus (immediate dominance and linear precedence) ermöglicht es, für spezielle kontextfreie Syntaxen eine abgekürzte Notation zu verwenden (Gazdar/Klein/Pullum/et al. 1985). Shieber (1983 a) wandelt den Earley-Algorithmus so ab, daß er auf im ID/LP-Format geschriebene Syntaxen direkt anwendbar ist. Der neue Algorithmus erweist sich bei vielen ID/LP-Syntaxen als erheblich effektiver als Earleys angewendet auf die korrespondierenden kontexfreien Syntaxen. Dies trifft besonders auch für linguistisch interessante Fälle zu. Doch ist Shiebers Algorithmus nicht immer so günstig. Eingehend untersucht ist dies in Barton (1985).

Zu Zeit- und Platzbedarf nicht-kontextfreier Parser vgl. Aho/Ullman (1972—73, I, 456 ff.), zur Komplexität von Transformationsgrammatiken Berwick (1984 a).

Auch die Komplexität von Erzeugungsprozessen wurde untersucht. Es geht hier um den Zeitbedarf für Ableitungen. Ist G eine generative Syntax, so ist er gegeben durch die Anzahl der Ableitungsschritte, die mit G erforderlich sind, um eine beliebige durch G erzeugbare Kette k der Länge n zu erhalten. Erste Ergebnisse hierzu finden sich in Book (1971), einen Überblick gibt Salomaa (1973).

5. Literatur (in Auswahl)

Aho/Ullmann 1972—73 · Bresnan/Kaplan 1982 · Chomsky 1957 · Chomsky 1959 · Chomsky 1965 · Gazdar/Klein/Pullum et al. 1985 · Hermes 1961 · Maurer 1977 · Paul 1978 · Salomaa 1973.

Ursula Klenk, Göttingen
(Bundesrepublik Deutschland)