3. Kohärenzkonstitution im gesprochenen Deutsch

Hans-Jürgen Eikmeyer / Walther Kindt / Uwe Laubenstein / Sebastian Lisken / Thomas Polzin / Hannes Rieser / Ulrich Schade

Das Projekt "Theoretische Grundlagen und Simulation von Prozessen der Kohärenzkonstitution im gesprochenen Deutsch" hat sich schwerpunktmäßig mit dem Phänomen der "Reparatur" auseinandergesetzt. Reparaturen entstehen in der gesprochenen Sprache unter anderem dann, wenn einem Sprecher ein tatsächlicher oder ein vermeintlicher Fehler unterläuft, den er, sofern er ihn bemerkt, zumeist zu reparieren sucht. Ein typisches Beispiel einer Reparatur findet sich in (3.1).

(3.1) und vorne drauf liegt ein grünes eh n blaues Rechteck
(Forschergruppe Kohärenz 1987, S. 27)

Reparaturen sind erst in neuerer Zeit zu einem allgemein anerkannten Gegenstand linguistischer Untersuchungen geworden. Einen ersten Ansatz für ihre systematische Analyse stellen die Arbeiten von Schegloff, Jefferson und Sacks (1977) und Schegloff (1979) dar. In diesen Arbeiten wird der Begriff "Reparatur" ("repair") eingeführt. Des weiteren wird versucht, den Gegenstandsbereich der Reparaturen einzugrenzen und zu gliedern. Ein wichtiges Ergebnis dieser Arbeiten war die Erkenntnis, daß Reparaturen in ihrer syntaktischen Struktur nicht willkürlich, sondern offenbar nach bestimmten festen Regeln durchgeführt werden. Levelt (1983) konkretisierte diese Erkenntnis, indem er Reparaturen syntaktisch in einen Zusammenhang mit Koordinationskonstruktionen stellte.

Bei einer Klassifikation sind die grundlegenden Typen "Selbstreparatur" und "Fremdreparatur" zu unterscheiden (vgl. Schegloff et al. 1977; Schegloff 1979), je nachdem, ob der Sprecher einer gestörten Äußerung oder dessen Gesprächspartner die Reparatur vornimmt. In diesem Sinne stellt (3.1) eine Selbstreparatur dar, während in Beispiel (3.2) eine Fremdreparatur vorliegt.

(3.2) P: so und dann die nächst kürzeren Klötze
das sind Quadrate
R: n Würfel
P: ja Würfel
(Forschergruppe Kohärenz 1987, S. 38)

Reparaturen dienen nach der im Projekt vertretenen Auffassung, wie sie bereits in den Abschnitten 2.4.3 und 2.4.4 angesprochen wurde, der (Wieder-)Herstellung von Kohärenz bei der Sprachverarbeitung: Selbstreparaturen entstehen, wenn im Produktionssystem ein inkohärenter Zustand vorliegt; Fremdreparaturen, wenn das Rezeptionssystem in einen inkohärenten Zustand gerät.

Das Projekt hat sich bei seinen Untersuchungen auf (satzinterne) Selbstreparaturen beschränkt. Aber auch bei einer Beschränkung auf Selbstreparaturen und deren Beitrag zur Kohärenzherstellung ist das Rezeptionssystem zu berücksichtigen, da eine vom Produktionssystem durchgeführte Selbstreparatur vom Rezeptionssystem verarbeitet werden muß. Erfolgreiche Selbstreparaturen bewirken einerseits im Produktionssystem den Übergang von einem inkohärenten zu einem kohärenten Zustand (vgl. Abschnitt 2.4.3). Im Rezeptionssystem, also bei der Verarbeitung einer Selbstreparatur des Produktionssystems, treten andererseits besondere Verarbeitungsschwierigkeiten auf, da das Produktionssystem durch die Selbstreparatur einen Teil der zuvor produzierten Äußerung (in einer durch Modelle vom Produktionssystem zu explizierender Weise) zurücknimmt. Das Rezeptionssystem muß also einen vorliegenden (möglicherweise schon) kohärenten Zustand verwerfen, um einen anderen, ebenfalls kohärenten Zustand zu erreichen (vgl. Abschnitt 2.4.4).

Da sowohl die Produktion als auch die Rezeption von Selbstreparaturen den Prozeß der Kohärenzherstellung exemplarisch beleuchten, ergeben sich aus den Untersuchungen zu beiden Fällen wichtige Einsichten in diesen Prozeß. Um aber die Abhängigkeit der beiden Teilsysteme und die Funktion von Kohärenz für "Verständigung" (vgl. Abschnitt 2.4.1) zu erkennen, bedarf es der Untersuchung eines Phänomens, wie es durch die Selbstreparatur in idealer Weise vorliegt, aus dem Blickwinkel beider Teilsysteme. Das Projekt untersucht dementsprechend die Produktion von Selbstreparaturen durch das Produktionssystem ebenso wie ihre Verarbeitung durch das Rezeptionssystem.

Die Untersuchungen des Projektes umfassen im einzelnen eine allgemeine Analyse des Phänomens der Reparatur (Abschnitt 3.1), die Modellierung der Produktion von Selbstreparaturen in einem konnektionistischen Produktionsmodell (Abschnitt 3.2), die Modellierung der Produktion einer Unterklasse der Selbstreparaturen, der sogenannten "covert repairs", in einem symbolverarbeitenden Produktionsmodell (Abschnitt 3.3), die Modellierung der Rezeption von Äußerungen mit Reparaturen durch einen C-Parser (Abschnitt 3.4) sowie die Entwicklung eines grammatiktheoretischen Ansatzes, der die Verarbeitung von Reparaturen erlaubt (Abschnitt 3.5). Allen Ansätzen gemeinsam ist die Erkenntnis, daß sowohl Produktions- als auch Rezeptionssystem über eine Komponente bzw. über ein Teilsystem verfügen müssen, die bzw. das Inkohärenzen im System erkennt und gegebenenfalls die Kohärenz durch einen steuernden Eingriff wiederherzustellen versucht. Diese Komponente wird als Metakomponente bezeichnet. Die Funktionsweise eines Systems, das eine Metakomponente integriert, wird exemplarisch anhand der Reparaturanalyse in Abschnitt 3.4 beschrieben. Dabei wird auch aufgezeigt, welche Verbindungen ein solches System zu klassischen Parserformaten hat und in welcher Hinsicht es sich von diesen unterscheidet.

3.1 Reparaturen

Die Analyse des Phänomens der Reparatur beruht auf einer typologischen Untersuchung der im Forschergruppenkorpus (Forschergruppe Kohärenz 1987) vorkommenden Reparaturen. In diesem Abschnitt werden einige Ergebnisse dieser Untersuchung dargestellt; eine ausführliche Fassung liegt mit Kindt und Laubenstein (1990) vor. Im Gegensatz zu Levelt (1983) trennen Kindt und Laubenstein (1990) zwischen einer Reparaturtypologie aufgrund der zu reparierenden Inkohärenztypen und einer Typologie aufgrund der gewählten Reparaturstrategie. Die Unterscheidung von Reparaturen anhand von Inkohärenztypen reflektiert in einem ersten Schritt, in welchem Verarbeitungsstadium der im Produktionssystem durchzuführenden Zuordnungsprozesse das betreffende Inkohärenzproblem auftritt. Bezüglich der Verarbeitungsstadien kann das Problem darin liegen, daß zum gewünschten Zeitpunkt ein zu realisierendes Zuordnungsprodukt noch nicht vorliegt (Suchproblem) oder daß ein vorliegendes bzw. schon realisiertes Zuordnungsprodukt als fraglich bzw. falsch eingeschätzt wird (Korrektheitsproblem). Ein Beispiel für eine Reparatur, mit der ein Suchproblem überwunden wird, ist (3.3); ein Korrektheitsproblem ist dagegen Auslöser für die in (3.1) angeführte Reparatur.

(3.3) die stellen wir auf ehm . auf ihre kurze Seite (sehr leise) ehm wie nennt man das (kurzes Schmunzeln von P und R) auf die Querseite und zwar . hochkant
(Forschergruppe Kohärenz 1987, S. 31)

In einem zweiten Schritt werden Inkohärenzprobleme danach klassifiziert, zwischen welchen Verarbeitungsebenen eine Inkohärenz vorliegt. Dabei sind prinzipiell drei Problemtypen zu unterscheiden. Erstens kann das Problem auftreten, daß die sprachliche Form eines Zuordnungsproduktes als grammatisch inkorrekt eingeschätzt wird (formales Sprachproblem); die Inkohärenz betrifft in diesem Fall syntaktische und phonologische Ebenen. Ein Beispiel für ein solches Problem und seine Reparatur auf der phonolgischen Ebene stellt (3.4) dar.

(3.4) und auf diesem baun eh blauen Klotz
(Forschergruppe Kohärenz 1987, S. 25)

Zweitens sind Probleme möglich, bei denen ein sprachliches Zuordnungsresultat als solches zwar grammatisch korrekt ist, jedoch nicht die Sache bezeichnet, der es zugeordnet wird (Sachproblem); die Inkohärenz betrifft in diesem Fall den Übergang von einer Konzeptebene zu den Ebenen der phonologischen Kodierung. Beispiele für diese Art von Problemen mit ihrer Reparatur sind (3.1) und (3.5).

(3.5) und den linken eh Quatsch den roten stellst du links hin
(Forschergruppe Kohärenz 1987, S. 27)

Drittens schließlich kann sich ein Problem ergeben, wenn sich mit einem korrekten sprachlichen Zuordnungsresultat keine bzw. keine eindeutige Sache verbinden läßt (inhaltliches Sprachproblem). Diese Inkohärenz tritt für das Produktionssystem selbst auf, wenn sich während der Zuordnung sein benutzter Diskursbereich ändert; die Inkohärenz betrifft in diesem Fall die Verbindungen zwischen dem Diskursbereich und den Bereichen der sprachlichen Realisierung. Ein inhaltliches Sprachproblem liegt aber insbesondere auch dann vor, wenn sich mit dem an sich korrekten sprachlichen Zuordnungsresultat keine bzw. keine eindeutige Sache im internen Modell des Produktionssystems vom Rezeptionssystem verbinden läßt. Die Inkohärenz betrifft in diesem Fall die Verbindung des internen Modells mit den Bereichen der sprachlichen Realisierung. Ein Beispiel für die Beseitigung eines inhaltlichen Sachproblems mit einer Reparatur ist (3.6).

(3.6) P: mit der Längsseite stellst du ihn hin
R: ach so, ..
P: also mit der schmalen Längsseite
(Forschergruppe Kohärenz 1987, S. 22)

Inkohärenzprobleme schlagen sich beim Produktionsprozeß gewöhnlich in inkohärenten Äußerungen nieder, die einem Rezeptionssystem die Verarbeitung der Äußerung erschweren und es ihm gegebenenfalls unmöglich machen, die Äußerung so zu interpretieren, daß es ihm möglich ist, die Intention des Produzenten zu erkennen bzw. ihr zu entsprechen. Infolgedessen sucht – wie in Abschnitt 2.4.3 schon angesprochen wurde – das Produktionssystem Inkohärenzen jeglicher Art zu vermeiden bzw. sie zu beseitigen, sofern sie dennoch auftreten und erkannt werden. Sprachliches Mittel für die Beseitigung bzw. Überwindung von Inkohärenzen sind Reparaturen, wobei je nach Typ eines Inkohärenzproblems von dem Produktionssystem eine spezifische Reparaturstrategie ausgewählt wird.

Grundsätzlich kann für die grammatische Realisierung von Reparaturen auf zwei unterschiedliche Strategien zurückgegriffen werden: die Überbrückungsreparatur und die Nachtragsreparatur, die im folgenden kurz vorgestellt werden sollen.

Für die Unterscheidung von Überbrückungsreparaturen gegenüber Nachtragsreparaturen ist es notwendig zu definieren, unter welchen Bedingungen gesagt werden soll, daß in einem Satz eine *formale Äußerungsstörung* vorliegt. Diese soll dann gegeben sein, wenn eine der beiden folgenden Bedingungen erfüllt ist:

(a) Der Satz ist nicht in eine Sequenz von Wörtern zerlegbar, d.h. es kommt wenigstens ein Segment ohne Wortstatus (z.B. ein Wortfragment oder ein Hesitationssignal) vor.

(b) In dem Satz wird eine grammatische Konstruktion abgebrochen.

Die Reparatur einer formalen Äußerungsstörung soll Überbrückungsreparatur heißen,

da durch eine derartige Reparatur wenigstens die durch das zugehörige Störungssegment vorliegende Satzunterbrechung überbrückt wird. Typische Beispiele für Überbrückungsreparaturen sind (3.1), (3.4) und (3.5).

Im Gegensatz zu Überbrückungsreparaturen reparieren Nachtragsreparaturen keine formalen, sondern rein inhaltliche Äußerungsstörungen. Die Reparatur einer solchen inhaltlichen Äußerungsstörung durch einen Nachtrag ist gegeben, wenn wenigstens eine der beiden folgenden Bedingungen erfüllt ist:

(a) Im Nachtrag kommt eine Problemmanifestation, in Form eines Bestätigungs- oder Zurückweisungssignals vor, das nicht in unmittelbarer Nachbarschaft einer formalen Störung liegt.

(b) Der Nachtrag bedeutet eine Stagnation der semantischen Konstruktion des Satzes, d.h. in ihm werden Zuordnungen des vorausgehenden Äußerungsteils wiederholt oder zurückgenommen und dann durch andere Zuordnungen ersetzt.

Da sich eine Typologie von Reparaturen (ebenso wie eine Klassifikation von Versprechern und anderen Objektinkohärenzen) immer an der vorliegenden Oberflächenrealisierung ausrichten muß und nicht an den prozeduralen Problemen festgemacht werden kann, die zu den Reparaturen führen, ist es aufgrund der Oberflächenstruktur einfacher, Reparaturen nach der verwendeten Reparaturstrategie als nach dem Typ des ursächlichen Inkohärenzproblems zu klassifizieren. Im Zusammenhang mit der Kohärenzproblematik sind jedoch die verschiedenen oberflächenorientierten Abgrenzungsmöglichkeiten zwischen den einzelnen Reparaturtypen von geringerer Bedeutung. Eine ausführliche Darstellung und damit auch eine genaue Ausarbeitung der Typologie von Reparaturen, wie sie im Rahmen des Projektes vorgenommen wurde, findet sich jedoch in Kindt und Laubenstein (1990); auch auf eine genaue Abgrenzung dieser Typologie zu der von Levelt (1983) vorgeschlagenen Einteilung wird dort eingegangen.

Die genannte Einteilung der Reparaturen in Überbrückungsreparaturen bzw. Nachtragsreparaturen steht allerdings insofern auch mit der Überwindung inkohärenter Zustände im Produktionsprozeß in einem engen Zusammenhang, als bei Nachtragsreparaturen die im Produktionssystem aufgebaute syntaktische Struktur erhalten bleibt. Das heißt, ein vorliegender stabiler Zustand des syntaktischen Teilsystems des Produktionssystems ändert sich nicht durch die Produktion der Nachtragsreparatur. Dementsprechend eignen sich Nachtragsreparaturen nur für die Behandlung von Sachproblemen und inhaltlichen Sprachproblemen. Umgekehrt muß bei Inkohärenzen, die das syntaktische Teilsystem betreffen, die Überbrückungsreparatur gewählt werden. Natürlich führen aber nicht ausschließlich nur formale Sprachprobleme zu Überbrückungsreparaturen; auch inhaltliche Sprachprobleme und Sachprobleme können durch Überbrückungsreparaturen überwunden werden, wenn auch das syntaktische Teilsystem durch die Inkohärenz betroffen ist und es zur Beseitigung der vorliegen-

den Inkohärenz in einen anderen Zustand überführt werden muß. Ein Beispiel für eine Reparatur eines inhaltlichen Sprachproblems, bei der das syntaktische Teilsystem involviert ist, ist (3.7).

(3.7) `im rechten Winkel mit der Stirns/ mit der vorderen Stirnseite des grünen`
(Forschergruppe Kohärenz 1987, S. 21)

Neben der Unterteilung in Nachtrags- und Überbrückungsreparaturen lassen sich aus der Analyse der Oberflächenrealisierungen noch weitere Informationen darüber gewinnen, auf welche Weise in den Verarbeitungssystemen von Produzent und Rezipient Inkohärenzen beseitigt werden. Als relevante Analyseeinheiten werden in Kindt und Laubenstein (1990) unter anderem der sogenannte Reparaturbezug und der Reparaturversuch definiert. Die für die Inkohärenztypisierung relevante Version dieser beiden Konzepte liefert für (3.7) `mit der Stirns` als Reparaturbezug und `mit der vorderen Stirnseite` als Reparaturversuch. Durch einen Paarvergleich kann man nun als generelles Verfahrensprinzip der Inkohärenzbeseitigung erkennen, daß bestimmte Zuordnungen im Verarbeitungsprozeß wiederholt und andere modifiziert oder eingefügt werden. Insofern ist aus der Oberflächenanalyse jeweils eine bestimmte Bearbeitungsform als Verarbeitungsstrategie für den Übergang in einen kohärenten Zustand zu entnehmen. Beispielsweise kann die bei der Lösung von Suchproblemen zum Erreichen eines stabilen Zustandes erforderliche Zeit dadurch gewonnen werden, daß die Bearbeitungsform einer reinen Zuordnungswiederholung gewählt wird (vgl. hierzu die Modellierung der Produktion von sogenannten "covert repairs" in Abschnitt 3.3).

In den folgenden Abschnitten dieses Kapitels soll aufgezeigt werden, wie Selbstreparaturen und ihre Ursachen in den durch das Projekt entwickelten Modellen vom Produktionssystem bzw. vom Rezeptionssystem behandelt werden; das heißt also, wie die einzelnen Modelle inkohärente Zuständen ausgleichen und überwinden.

3.2 Das konnektionistische Produktionsmodell

3.2.1 Der Aufbau des konnektionistischen Produktionsmodells

Für die kognitive Fähigkeit der Sprachproduktion von Erwachsenen existieren in der Literatur konnektionistische Modelle in der Tradition von Gary Dell (Dell & Reich 1980; Dell 1985, 1986, 1988; Stemberger 1985a, 1985b, 1990; Berg 1986, 1988; MacKay 1987; Schade 1988, 1990a). Die verschiedenen Modelle unterscheiden sich in der Repräsentation syntaktischer Regeln und der damit verbundenen Realisation der Sequentialisierung linguistischer Einheiten wie Wörter, Silben und Phoneme während des Produktionsprozesses. Alle diese Modelle sind lokale konnektionistische Modelle

(nach der Terminologie von Rumelhart & McClelland 1986), da sie jede der genannten linguistischen Einheiten, also jedes Wort, jede Silbe, jedes Phonem aber auch jedes phonologische Merkmal, durch einen Knoten eines konnektionistischen Netzwerks repräsentieren.

Grundsätzlich bestehen konnektionistische Modelle aus einem Netzwerk miteinander verbundener Knoten, die über einen Aktivierungswert verfügen. Übersteigt der Aktivierungwert eines Knotens seinen Schwellwert, so kann der Knoten an seine Nachbarn die Aktivierung weiterleiten. Die Menge der Aktivierung, die ein Knoten weiterleitet, hängt dabei von zwei Faktoren ab: Sie ist zum einen proportional zu der Aktivierung des sendenden Knotens, und sie ist zum anderen proportional zu der Stärke der Verbindung zwischen dem sendenden Knoten und dem Empfänger. Aufgrund der eingehenden Aktivierung verändern sich die Aktivierungswerte der Knoten.

Die Knoten der konnektionistischen Modelle für die Sprachproduktion sind in Ebenen angeordnet. Es gibt beispielsweise eine Ebene mit Wortknoten, eine Ebene mit Silbenknoten, eine Ebene mit Phonemknoten und mehrere Ebenen mit Knoten für phonologische Merkmale, etwa Ebenen für Stimmhaftigkeit, für Artikulationsorte und für Artikulationsarten. In dem hier zugrunde gelegten Modell sind alle Knoten einer Ebene paarweise durch inhibitorische Leitungen verbunden. Inhibitorische Leitungen sind Verbindungen zwischen Knoten, die zu einer Verkleinerung des Aktivierungswertes beim empfangenden Knoten führen, ihn also hemmen. Inhibitorische Verbindungen ergeben sich aus paradigmatischen Relationen und beschreiben Wahlmöglichkeiten für bestimmte strukturelle Positionen: Zu einer bestimmten Zeit kann immer nur ein Wort, nur eine Silbe und nur ein Phonem geäußert werden.

Anders als bei den inhibitorischen Leitungen, die von Dell und MacKay nicht vorgesehen sind, stimmen die Modelle bezüglich der exzitatorischen Leitungen (Verbindungen, über die Aktivierung weitergegeben wird, die zur Vergrößerung des Aktivierungswertes des empfangenden Knotens führt) überein. Exzitatorische Leitungen ergeben sich aus syntagmatischen Relationen zwischen den linguistischen Einheiten und verbinden stets Knoten benachbarter Ebenen: beispielsweise ist ein Silbenknoten mit denjenigen Knoten der Phonemebene verbunden, die die Phoneme der entsprechenden Silbe repräsentieren (vgl. Abbildung 1). Alle exzitatorischen und inhibitorischen Verbindungen sind symmetrisch, so daß bei einer Produktion Feedbackeffekte auftreten (vgl. Dell 1985).

Wird mit der Hilfe eines konnektionistischen Netzwerkes der beschriebenen Art Sprachproduktion modelliert, so wird derjenige Knoten auf einen hohen Aktivierungswert gesetzt, der die zu äußernde linguistische Einheit repräsentiert. Soll beispielsweise lediglich ein Wort geäußert werden, wäre dies der entsprechende Knoten der Wortebene. Soll dagegen ein Sachverhalt mit einem oder mehreren Sätzen ausgedrückt werden, so wird ein entsprechender bzw. mehrere entsprechende Knoten auf einer der semantischen Ebenen voraktiviert.

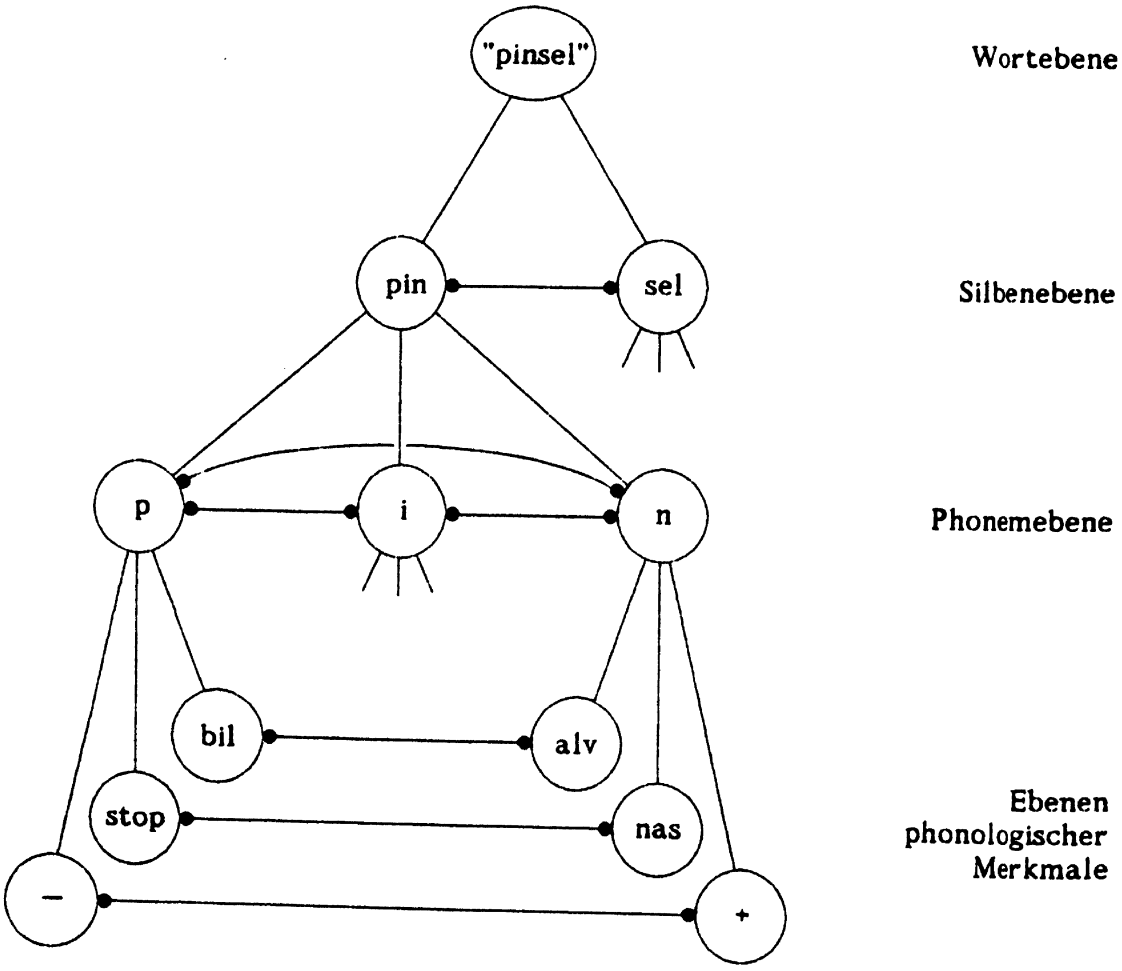


Abb. 1: Ausschnitt aus einem konnektionistischen Netzwerk zur Sprachproduktion

Die empirischen Erkenntnisse, die über den kognitiven Prozeß der Sprachverarbeitung vorliegen, erfordern die sequentielle Realisierung der Phoneme eines Wortes (vgl. z.B. Meyer 1988). In einem konnektionistischen Modell wird daher in bestimmten kurzen zeitlichen Abständen durch eine Selektion ermittelt, welches jeweils der höchst aktivierte Phonemknoten ist bzw. welches die höchst aktivierten Merkmalknoten der betroffenen Ebenen von phonologischen Merkmalen sind. Diese Knoten legen das Produktionsresultat für den betreffenden Zeitpunkt fest und werden anschließend durch eine Selbstinhibition auf den Aktivierungswert "0" zurückgesetzt.

Für die Sequentialisierung der linguistischen Einheiten ist in dem zugrunde gelegten Modell ein spezielles Teilnetz zuständig, in dem syntaktische Regeln durch sogenannte "Kontrollknotenketten" repräsentiert werden (vgl. Schade 1988, 1990a; Eikmeyer & Schade 1991). Eine typische Kontrollknotenkette, mit der etwa die Produktion einer Nominalphrase realisiert werden könnte, besteht aus den Knoten DET, ADJ und N. Sofern eine solche Kette aktiviert wird, hat immer genau einer der genannten Knoten den Aktivierungswert "1" und die anderen den Wert "0". Während der Produktion kann derjenige Knoten, der aktivert ist, Wortknoten seiner syntaktischen Kategorie unterstützen. Nachdem ein entsprechendes Wort produziert wurde, gibt der

aktivierte Knoten der Kette seine Aktivierung an seinen Nachfolger weiter und hemmt sich selbst. Durch dieses Verfahren kann die Sequentialisierung realisiert werden.

3.2.2 Inkohärente Zustände

Die im vorangegangenen Abschnitt beschriebenen Ebenen des konnektionistischen Modells bilden im Sinne von Kapitel 2 die Teilsysteme des Produktionssystems, das durch das Modell abgebildet wird. Entsprechend der in Kapitel 2 vertretenden Auffassung von Kohärenz sind folglich jeweils zwei benachbarte Ebenen (Teilsystem) zum Zeitpunkt T genau dann zueinander in kohärenter Beziehung, wenn die zu diesem Zeitpunkt jeweils am höchsten aktivierten Knoten der Ebenen exzitatorisch miteinander verbunden sind. Des weiteren befindet sich das gesamte System zum Zeitpunkt T genau dann in einem kohärenten Zustand, wenn sich zu diesem Zeitpunkt jede Ebene (jedes Teilsystem) mit allen ihren benachbarten Ebenen (Teilsystemen) in einem kohärenten Zustand befindet.

Die Bedeutung dieser Übertragung des Kohärenzprinzips auf die Verhältnisse des konnektionistischen Produktionsmodells läßt sich an einem einfachen Beispiel verdeutlichen.

Betrachten wir dazu zunächst die Silbenebene und die Wortebene des konnektionistischen Produktionsmodells. Ein Knoten der Silbenebene repräsentiert eine Silbe, ein Knoten der Wortebene ein Wort. Ein Silbenknoten ist genau dann mit einem Wortknoten verbunden, wenn die zugehörige Silbe eine Silbe des fraglichen Wortes ist. Nehmen wir nun an, die beiden Ebenen stehen zu einem Zeitpunkt T in kohärenter Beziehung zueinander. Dann sind die am höchsten aktivierten Knoten beider Ebenen exzitatorisch miteinander verbunden. Das bedeutet, daß die Silbe, die zum Zeitpunkt T zur Produktion vorgesehen ist, die Silbe, deren Knoten also die höchste Aktivierung aufweist, eine Silbe des Wortes ist, dessen Produktion ebenfalls zum Zeitpunkt T ansteht.

Nun verfügt das konnektionistische Produktionsmodell nicht nur über die Ebenen, die die linguistisch relevanten Einheiten enthalten, sondern auch über Ebenen von Kontrollknoten, die die Produktion steuern und die eine Art verallgemeinerter Syntax repräsentieren. Eine dieser Kontrollknotenebenen ist die Ebene der Silbenkontrollknoten, ebenfalls eine Nachbarebene der Silbenebene, deren Aufgabe darin liegt, für die korrekte Produktionsabfolge der Silben eines Wortes zu sorgen. Sollte auf der Ebene der Silbenkontrollknoten etwa der am höchsten aktivierte Knoten derjenige sein, der für die Produktion einer Vorsilbe zuständig ist, so muß im Kohärenzfall der am höchsten aktivierte Silbenknoten eine Vorsilbe repräsentieren, da der genannte Kontrollknoten nur mit solchen Silbenknoten exzitatorisch verbunden ist. Stehen also Silbenebene, Wortebene und die Ebene der Silbenkontrollknoten wechselseitig in kohärenter Beziehung zueinander, so muß die Silbe, die zur Zeit produziert wird,

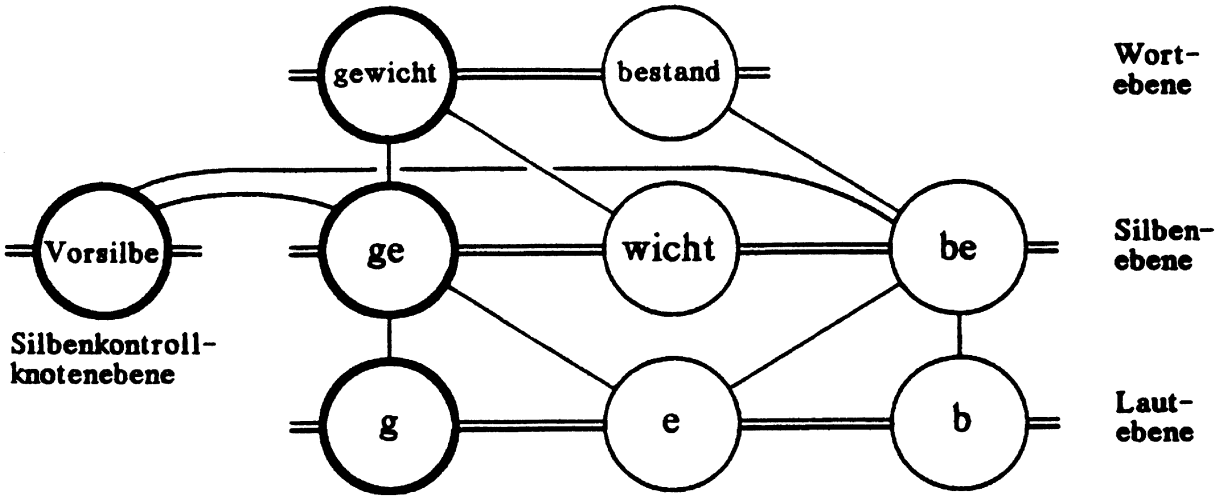
nicht nur irgendeine Silbe des aktuellen Wortes, sondern exakt dessen Vorsilbe sein (vgl. Abbildung 2).

Wie aus dem Beispiel ersichtlich wird, ist das konnektionistische Sprachproduktionsmodell immer dann in einem kohärenten Zustand, wenn der Produktionsprozeß korrekt abläuft. Das heißt, durchläuft der Produktionsprozeß eine Folge von kohärenten Zuständen, so daß eine Prozeßkohärenz vorliegt, so wird auch das Ergebnis der Produktion eine Äußerung sein, die für sich genommen als kohärent bezeichnet werden kann. In dieser Hinsicht verdeutlicht das konnektionistische Produktionsmodell, wie aus Prozeßkohärenz Objektkohärenz folgt (vgl. Kapitel 2).

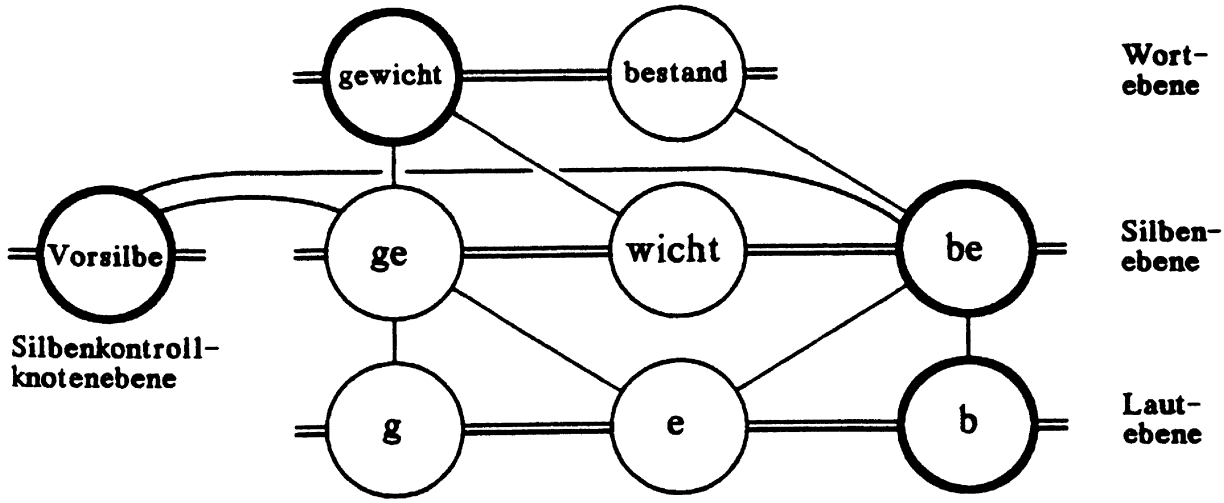
Um die Produktion inkohärenter Äußerungen zu vermeiden, ist es jedoch nicht notwendig, daß sich das System stets in einem kohärenten Zustand befindet. Notwendig für die Gewährleistung einer kohärenten Äußerung ist lediglich, daß es sich in einem kohärenten Zustand befindet, wenn eine Selektion erfolgt. Mit anderen Worten, die Zeitpunkte, zu denen der Produktionsprozeß auf kohärente Zustände des Modells zurückgreifen muß, sind genau die Zeitpunkte, zu denen das Modell tatsächlich die Produktion eines Lautes leistet.

Direkt nach einer Produktion werden alle Knoten stark gehemmt, die linguistische Einheiten repräsentieren, deren Produktion durch den gerade erfolgten Produktionsschritt als abgeschlossen gekennzeichnet werden kann. Nehmen wir beispielsweise an, der letzte Laut einer Vorsilbe wurde soeben produziert. Dann wird sowohl der Lautknoten gehemmt, der den gerade produzierten Laut repräsentiert, als auch der am höchsten aktivierte Silbenknoten. War das Modell während der angenommenen Produktion in einem kohärenten Zustand, so repräsentiert dieser Silbenknoten zum einen eine Vorsilbe und zum anderen eine Silbe des aktuell zu produzierenden Wortes.

Auf der Kontrollebene aktiviert der Silbenkontrollknoten, der für die Produktion der Vorsilbe zuständig war, nach Beendigung der Produktion der Vorsilbe einen Silbenkontrollknoten, der die Produktion der nächsten Silbe steuert, und hemmt sich dann selbst. Damit ist dann beispielsweise ein Silbenkontrollknoten, der für die Produktion einer Stammsilbe zuständig ist, der am höchsten aktivierte Knoten der Ebene der Silbenkontrollknoten. Auf der Ebene der Silbenknoten ist aber möglicherweise die Vorsilbe eines nicht aktuellen Wortes zu diesem Zeitpunkt, also direkt nach erfolgter Selektion, noch am höchsten aktiviert. Damit befindet sich das System zu diesem Zeitpunkt nicht in einem kohärenten Zustand. Diese Inkohärenz muß jedoch erst bis zur Produktion des nächsten Lautes überwunden werden, damit weiterhin die Lautfolge produziert wird, die geplant ist und die als kohärent im Sinne einer Objektkohärenz angesehen werden kann. In aller Regel erfolgt die Überwindung derartiger kurzfristiger Inkohärenzen durch die Ausbreitung von Aktivierung, wie sie für ein konnektionistisches Modell charakteristisch ist. Dadurch, daß auf der jeweiligen hierarchisch höheren Ebene – im Beispiel also die Wortebene – der Knoten, der die aktuell zu produzierende linguistische Einheit dieser Ebene repräsentiert, alle anderen Knoten der Ebene in bezug auf den Aktivierungswert deutlich übertrifft,



Kohärenter Fall



Inkohärenter Fall

Abb. 2: Beispiele für kohärente bzw. inkohärente Zustände im konnektionistischen Produktionsmodell (Der jeweils höchst aktivierte Knoten einer Ebene ist durch die dickere Umrandung hervorgehoben.)

können im Normalfall – also bei nur geringfügigen Störungen – auch auf der hierarchisch niedrigeren Ebene – im Beispiel also auf der Silbenebene – nur die Knoten eine größere Menge Aktivierung ansammeln, die mit dem hoch aktivierten Knoten der höheren Ebene exzitatorisch verbunden sind. Das sind also die Knoten, die linguistische Einheiten repräsentieren, die Teile der gewünschten hierarchisch höheren Einheit sind. Am meisten Aktivierung kann natürlich derjenige Knoten der

niedrigeren Ebene ansammeln, der auch von den Kontrollknoten unterstützt wird, um die korrekte Reihenfolge zu gewährleisten. Mit anderen Worten, der jedem konnektionistischen Modell eigene Mechanismus der Aktivierungsausbreitung gewährleistet im Normalfall das Erreichen von stabilen Zuständen zwischen den durch die Modellebenen gegebenen Teilsysteme des modellierten Produktionssystems im Sinne von Abschnitt 2.3.1.

Befindet sich das Modell zum Zeitpunkt einer Selektion aber (immer noch) in einem inkohärenten Zustand, kann das zu einer inkohärenten Äußerung führen, etwa einer Äußerung, die einen Versprecher enthält. Eine solche Äußerung sollte dementsprechend die Produktion einer Reparatur nach sich ziehen, wobei die Produktion einer Reparatur in zwei Schritte zerfällt. Als erstes muß die Notwendigkeit einer Reparatur überhaupt erkannt werden. Erst dann kann in einem zweiten Schritt ein Reparaturversuch eingeleitet werden.

3.2.3 Die Überwindung der inkohärenten Zustände durch die Produktion von Reparaturen

Zum Erkennen von inkohärenten Zuständen verfügt das konnektionistische Produktionsmodell über eine Monitorkomponente, die zum Zeitpunkt einer Selektion für bestimmte Ebenen überprüft, ob sie sich in kohärenter Beziehung zueinander befinden (vgl. Schade 1990b). Gegebenenfalls, also falls die Monitorkomponente Anzeichen für einen inkohärenten Zustand auffindet, bricht die Komponente den Produktionsprozeß ab, da in einem solchen Fall mit der Produktion einer inkohärenten Äußerung gerechnet werden muß. Nach einer solchen Unterbrechung wird ein Reparaturversuch eingeleitet, mit dem die Prozeßkohärenz (wieder-)hergestellt werden soll.

Sofern die Monitorkomponente eine Inkohärenz feststellt, die zum Abbruch des Produktionsprozesses führt, wird – im Gegensatz zu sogenannten "Editor"-Modellen (vgl. z.B. Baars, Motley & MacKay 1975; Laver 1980; Motley, Baars & Camden 1981, 1982; Motley, Camden & Baars 1979, 1981, 1982 sowie Levelt 1989) – nicht analysiert, wie der Fehler exakt aussieht bzw. aussehen würde, der als Folge des inkohärenten Zustandes des Systems als Inkohärenz in der schon produzierten bzw. zur Produktion anstehenden Äußerung auftritt. Eine solche Analyse kann ein konnektionistisches System nicht leisten. Sie setzt jedoch auch in symbolverarbeitenden Systemen eine unökonomische Vervielfachung von Kenntnissen über den Aufbau von Wörtern, Phrasen, Sätzen usw. voraus, die letztlich nur schwer zu begründen ist (vgl. Berg 1986).

Das konnektionistische Modell greift also für den Reparaturversuch, mit dem der Produktionsprozeß nach dem Abbruch durch die Monitorkomponente wiederum einsetzt, nicht auf Wissen über das exakte Aussehen der möglicherweise schon erfolgten Fehlleistung zurück. Der Ansatzpunkt für einen Reparaturversuch im Fall eines

inkohärenten Zustandes des konnektionistischen Produktionsmodells ergibt sich lediglich aus den Aktivierungsmustern der verschiedenen Kontrollknotenebenen zum Zeitpunkt des Produktionsabbruchs und dem "Wissen" darüber, zwischen welchen Ebenen der inkohärente Zustand bestand, der für die Monitorkomponente den Anlaß bildete, den Produktionsprozeß abzubrechen. Mit Einleitung des Reparaturversuches ist entsprechend im Modell auch keine allgemeine Manipulation an Aktivierungswerten verbunden. Es erfolgt lediglich eine Veränderung in den Aktivierungswerten einiger Kontrollknoten auf der Ebene von Kontrollknoten, die der hierarchisch höheren der beiden Ebenen zugeordnet ist, zwischen denen die Inkohärenz durch die Monitorkomponente bemerkt wurde (vgl. Schade 1990a). Mit dieser (im Vergleich zu den erwähnten "Editor"-Modellen ökonomischen) Strategie, für Reparaturversuche lediglich aufgrund ebenenbezogener Information einzuleiten, entspricht das konnektionistische Produktionsmodell den anhand der empirischen Daten gewonnenen Erkenntnissen über die Typologie von Inkohärenzproblemen und deren Überwindung durch Reparaturen, wie sie in Abschnitt 3.1 ausgeführt wurden.

Im konkreten Fall setzt der Produktionsprozeß normalerweise am Beginn der Produktion derjenigen linguistischen Einheit wieder ein, die der hierarchisch höheren der beiden Ebenen zuzuordnen ist, zwischen denen die Inkohärenz vorlag. Trat beispielsweise eine Inkohärenz zwischen der Silbenebene und der Wortebene auf, so wird das aktuelle Wort von Beginn an erneut produziert, wie dies etwa in Beispiel (3.8) geschieht.

(3.8) worauf . ein blaues Br/ eh Blau/ Bauklötzchen, . als liegendes Rechteck, langgestreckt liegt
(Forschergruppe Kohärenz 1987, S. 26)

Ergibt sich dagegen eine Inkohärenz zwischen der Wortebene und der Konzeptebene, so setzt die Produktion am Beginn der aktuellen Phrase wieder ein, wie es etwa in den Beispielen (3.1) und (3.9) der Fall ist.

(3.9) und da setzt du jeweils rechts und links diese grünen Zylin eh diese roten Zylinder .. neben
(Forschergruppe Kohärenz 1987, S. 29)

Das Verhalten, das das konnektionistische Produktionsmodell zeigt, kann entsprechend der empirisch-simulativen Methode mit empirischen Daten über Reparaturen verglichen werden, wobei jedoch beachtet werden muß, daß aus den empirischen Daten vielfach nicht exakt hervorgeht, welche Art von Fehler ein Sprecher gemacht hat. Das konnektionistische Modell kann die Produktion eines bestimmten Versprechers oft aufgrund verschiedenster Ursachen und Störungen nachbilden, wobei diese unterschiedlichen Ursachen unter Umständen zu unterschiedlichen Reparaturversuchen Anlaß geben.

Entsprechend ergibt sich aus der Strategie, die das konnektionistische Modell für den Beginn eines Reparaturversuchs anwendet, lediglich die Vorhersage für empirische Daten, daß ein Reparaturversuch mehrheitlich mit der Produktion des Beginns derjenigen linguistischen Einheit beginnen sollte, die der anhand der Oberflächenrealisierung als fehlerhaft eingestuften direkt übergeordnet ist. Diese Vorhersage läßt sich statistisch belegen.

Als Ursache für Reparaturen wurden in diesem Abschnitt bisher Inkohärenzen angenommen, die sich direkt aus inkohärenten Beziehungen zwischen zwei benachbarten Ebenen des konnektionistischen Produktionsmodells ergaben. Damit haben die bisherigen Ausführungen über die Produktion von Reparaturen (bzw. deren Ursachen) zwei wichtige Punkte unberücksichtigt gelassen. Dieses ist zum einen die Frage, ob bzw. inwieweit das konnektionistische Produktionsmodell über ein internes Modell vom Rezipienten verfügt und wie Inkohärenzen in bezug auf ein solches internes Modell aussehen und überwunden werden. Zum anderen ist zu diskutieren, wie das Modell den Fall behandelt, bei dem im Verlauf einer Selektion zwei Ebenen nach der in Abschnitt 3.2.2 gegebenen Definition daraufhin überprüpft werden, ob sie zueinander in kohärenter Beziehung stehen, und bei dem zu diesem Zeitpunkt in einer dieser Ebenen (gewöhnlich der hierarchisch niedrigeren) die Aktivierungsverteilung (noch) "unausgeprägt" ist, so daß die Frage nach der kohärenten Beziehung nicht entschieden werden kann. "Unausgeprägt" bedeutet in diesem Zusammenhang, daß keiner der Knoten der entsprechenden Ebene ausreichend Aktivierung angesammelt hat, um eine bestimmte Schwelle (Produktionsschwelle bzw. Selektionsschwelle) zu überschreiten. Im Sinne von Abschnitt 2.3.1 ist damit das durch die betreffende Ebene repräsentierte Teilsystem noch nicht in einem stabilen Zustand, was nach der in Abschnitt 3.1 gegebenen Typologie von Inkohärenzproblemen dem sogenannten "Suchproblem" entspricht. Beim Auftreten von Suchproblemen produziert das konnektionistische Produktionsmodell ähnlich wie das symbolverarbeitende Modell, das in Abschnitt 3.3 vorgestellt wird, sogenannte "covert repairs". Entsprechende Angaben zu den Ursachen und dem Aussehen von "covert repairs" sowie über den Vergleich der beiden Modell findet sich daher ebenfalls in Abschnitt 3.3.

Zu der Frage nach dem internen Modell vom Rezeptionssystem ist zu sagen, daß ein internes Modell in der derzeit vorliegenden Version des konnektionistischen Produktionsmodells nicht enthalten ist. Allerdings ist die Forderung, daß im Falle einer durch die Monitorkomponente festgestellten Inkohärenz ein Reparaturversuch unternommen wird, als wichtige Randbedingung anzusehen. Sie ist nur dann sinnvoll, wenn die produzierte Äußerung durch einen Rezipienten verarbeitet werden soll. Anderenfalls würde es ja genügen, wenn das Produktionssystem über Mechanismen verfügte, mit denen inkohärente Systemzustände überwunden werden könnten. Für die Wiederherstellung eines kohärenten Zustandes innerhalb des Produktionssystems selbst ist die Durchführung einer Reparatur nicht notwendig; doch eine entsprechende Äußerung wie etwa (3.10) – statt (3.1) – kann durch einen Rezipienten nicht korrekt verarbeitet werden.

(3.10) und vorne drauf liegt ein grünes eh das war jetzt die falsche Farbbezeichnung, aber ich weiß wieder, was ich meine . Rechteck

Für die Bearbeitung einer Teilklasse von inhaltlichen Sachproblemen, bei denen nämlich die Inkohärenz im Produktionssystem dadurch auftritt, daß sich mit einer Äußerung keine bzw. keine eindeutige "Sache" innerhalb des internen Modells vom Rezeptionssystem verbinden läßt (vgl. Abschnitt 3.1), ist natürlich die Erweiterung des Produktionsmodells durch ein Teilsystem, daß das interne Modell vom Rezeptionssystem repräsentiert, notwendig. Entsprechende Probleme bzw. deren Reparaturen können also durch das konnektionistische Produktionsmodell in seiner derzeitigen Ausprägung nicht nachgebildet werden.

Die erstrebenswerte Erweiterung des Modells in dieser Hinsicht stellt eine wichtige und interessante Zielrichtung für Projekte dar, die auf dem hier erreichten aufbauen; sie bedarf jedoch einer eingehenden Kenntnis der möglichen Struktur von internen Modellen ebenso wie Hypothesen über deren Veränderung in der Zeit abhängig vom Geäußerten. Zudem müßten sowohl die internen Modelle selbst wie auch ihre Veränderungsabläufe konnektionistisch repräsentiert werden können. Die angeführten Probleme zeigen, wie aufwendig eine entsprechende Erweiterung sein dürfte; sie lassen aber auch die Vermutung zu, daß die Monitorkomponente bei ihrer Prüfung auf Kohärenz, sofern sie das interne Modell tangiert, hinter der inkrementell fortschreitenden Äußerung zurückfällt, da vor einer Überprüfung auf Kohärenz das interne Modell zunächst entsprechend der Äußerung verändert werden muß. Diese einfache Vorüberlegung erlaubt damit die Hypothese, daß Reparaturen, die inhaltliche Sachprobleme überwinden sollen, später als andere einsetzen und auch aus diesem Grund vergleichsweise häufig als Nachtragsreparaturen realisiert werden müssen.

3.3 Das symbolverarbeitende Produktionsmodell

Das diesem Abschnitt zugrundeliegende symbolverarbeitende Modell (vgl. Eikmeyer 1987a, 1989; Schade & Eikmeyer 1991) bildet Produktionsprozesse nach, die zur Produktion von sogenannten "covert repairs" führen. "Covert repairs" bilden eine Unterklasse von Überbrückungsreparaturen (vgl. Abschnitt 3.1) und sind nach Levelt (1983) dadurch gekennzeichnet, daß sie gerade nicht anzeigen, welche Art von Problemen zu ihrer Produktion geführt hat. Zur Klasse der "covert repairs" gehören drei verschiedene Typen, nämlich Pausen, Hesitationen und Wiederholungen (vgl. Levelt 1983; Kindt & Laubenstein 1990). Nach der Interpretation von Siegman (1979) haben "covert repairs" zumeist folgende Funktionen: Sie signalisieren einerseits Suchprobleme des Produzenten (vgl. Abschnitt 3.1); andererseits haben sie in dialogischen Situationen die Funktion als "turn holder"; sie erlauben es also dem Sprecher, trotz seiner (Such-) Probleme am Rederecht festzuhalten.

3.3.1 Der Aufbau des symbolverarbeitenden Produktionsmodells

Das symbolverarbeitende Produktionsmodell geht nicht davon aus, daß bei einer Produktion von "covert repairs" die Symbolverarbeitung direkt gestört ist, daß also fehlerhafte Manipulationen von symbolischen Ausdrücken aufgetreten sind. Das Modell verlagert die Explikation der "covert repairs" vielmehr auf die Ebene der Prozeßstruktur.

Grundlegend für das Modell ist die Vorstellung, daß in einem Produktionssystem, das kognitive Prozesse abbilden soll, eine Anzahl von Prozessen parallel ablaufen (vgl. Bock 1982 zur Begründung einer derartigen Annahme für den hier relevanten Prozeß der Sprachproduktion). Jeder einzelne Prozeß, der etwa in einem Teilsystem des Produktionssystems abläuft, kann intern eine sequentielle Struktur besitzen, er kommuniziert jedoch mit anderen (parallel ablaufenden) Prozessen in anderen Teilsystemen des Gesamtsystems.

Grundsätzlich kann eine Prozeßkommunikation entweder synchron oder asynchron ablaufen. Bei synchroner Kommunikation signalisieren die Prozesse einander zunächst die Bereitschaft zum Senden bzw. Empfangen. Sind beide Prozesse bereit, kann die Kommunikation stattfinden. Nach erfolgter Kommunikation wird dann die Verbindung abgebaut. Bei asynchroner Kommunikation dagegen schickt der sendende Prozeß seine Information ab, ohne sicher zu sein, daß der empfangende Prozeß sie auch erhält. Das vorgeschlagene Produktionsmodell basiert auf asynchroner Kommunikation zwischen den beteiligten Prozessen. In einem System paralleler Prozesse mit asynchroner Kommunikation können in natürlicher Weise Synchronisationsprobleme auftreten; beispielsweise dann, wenn der Sender sendet, aber der empfangende Prozeß nicht empfangsbereit ist, oder wenn ein Prozeß, der auf Information wartet, diese nicht erhält. Wie in Abschnitt 3.3.3 näher erläutert wird, nimmt das Modell derartige Synchronisationsfehler als Anlaß zur Produktion von "covert repairs".

Im Modell wird von vier parallelen Prozessen ausgegangen:
- dem "turn taking"-Prozeß (TT), der die "turn taking"-Regeln von Sacks, Schegloff und Jefferson (1974) realisiert,
- dem Prozeß der syntaktisch-semantischen Planung (SSP),
- dem Prozeß der phonetischen Codierung (PC) und
- dem Artikulationsprozeß (AR).

Die genannten Prozesse sind in der gegebenen Reihenfolge hierarchisch angeordnet. Das heißt, die Ausgabe des "turn taking"-Prozesses bildet die Eingabe für den Prozeß der syntaktisch-semantischen Planung; dessen Ausgabe wird als Eingabe für den Prozeß der phonetischen Codierung benutzt, und die Ausgabe dieses Prozesses dient dem Artikulationsprozeß als Eingabe. Aus dieser Anordnung folgt, daß die genannten Prozesse entsprechend ihrer hierarchischen Anordnung kommunizieren müssen. Dafür wird in allen Fällen eine asynchrone Prozeßkommunikation angenommen.

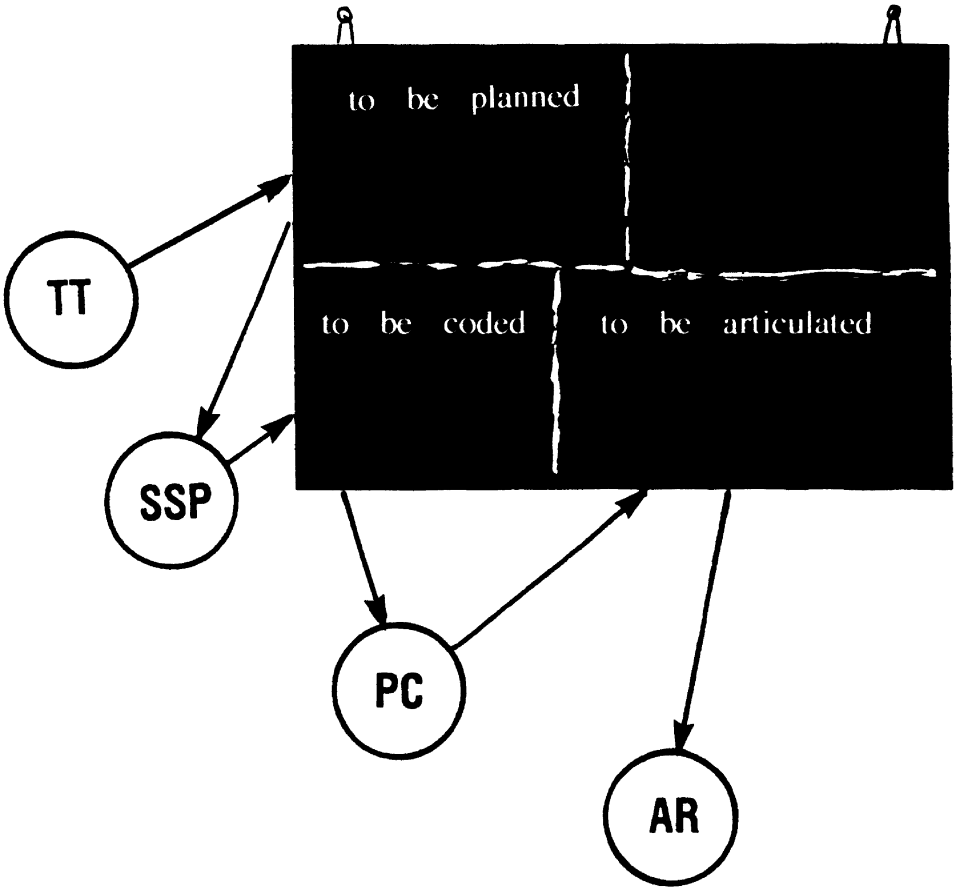


Abb. 3: Aufteilung der Zugriffsregionen des "Puffer"-Prozesses

Will man Prozeßkommunikation simulieren, hat man prinzipiell die Wahl, entweder die synchrone oder die asynchrone Kommunikation als grundlegend zu akzeptieren und die jeweils andere Form mit Hilfe der gewählten zu explizieren. So nimmt z.B. das CSP-Modell (communicating sequential processes; vgl. Hoare 1985) synchrone Kommunikation als Basisverfahren an. Bei dem üblichen Verfahren der Darstellung asynchroner mit synchroner Kommunikation wird zwischen die beiden kommunizierenden Prozesse ein dritter Pufferprozeß geschaltet. Dieser kommuniziert synchron mit je einem der beiden anderen Prozesse, so daß deren Kommunikation untereinander asynchron wird.

In der Simulation des Modells (vgl. Eikmeyer 1987a, 1989) in CheOPS (objekt-orientierter PROLOG-Dialekt, vgl. Eikmeyer 1987b) wird Prozeßkommunikation durch die Technik des Versendens von Nachrichten dargestellt, eine Form der synchronen Kommunikation. Entsprechend des von Hoare vorgeschlagenen Verfahrens dient in der Simulation des symbolverarbeitenden Modells ein sogenanter "Puffer"-Prozeß der Realisierung der zu simulierenden asynchronen Prozeßkommunikation. Die oben aufgeführten Prozesse hinterlassen bei diesem "Puffer"-Prozeß ihre eigenen Resultate (Ausgaben) und holen Resultate anderer Prozesse, die sie als

Eingabe benötigen, hier ab. Somit hat der Pufferprozeß die Funktion einer schwarzen Tafel ("blackboard"). Einzelne Bereiche oder Regionen auf der Blackboard dienen jeweils der Kommunikation zweier hierarchisch aufeinander folgender Prozesse (vgl. Abbildung 3).

3.3.2 Inkohärente Zustände

Die im Zusammenhang mit dem symbolverarbeitenden Modell fokussierten Inkohärenzen sind Suchprobleme (vgl. Abschnitt 3.1). Suchprobleme treten in diesem Modell dann auf, wenn einer der vier zentralen Prozesse im "Puffer"-Prozeß eine Eingabe abruft, ohne daß in dem entsprechenden Zugriffsbereich (von einem anderen Prozeß) bis zu diesem Zeitpunkt eine Nachricht hinterlegt worden wäre, die als Eingabe verwendet werden kann. Suchprobleme stellen sich also in diesem Modell als Synchronisationsprobleme dar.

Korrektheitsprobleme, die im vorangegangenen Abschnitt für das konnektionistische Modell ausführlich diskutiert wurden, werden im symbolverarbeitenden Modell nicht fokussiert. Sie würden nach diesem Modell vorliegen, wenn einer der angeführten vier Prozesse eine Ausgabe an seinen in Hierarchie nachfolgenden Prozeß weitergibt, die dieser nicht verarbeiten kann (vgl. auch Abschnitt 2.4.3). Lediglich ein Korrektheitsproblem soll im Zusammenhang mit der Frage nach der Modellierung des internen Modells in Abschnitt 3.3.3 kurz angesprochen werden.

3.3.3 Die Überwindung der inkohärenten Zustände durch die Produktion von "covert repairs"

Die Überwindung von inkohärenten Zuständen, die auf Suchproblemen beruhen, ist prinzipiell in einfachster Weise durch Warten lösbar. Liegt beispielsweise im symbolverarbeitenden System für einen Prozeß keine Eingabe vor, enthält also die entsprechende Zugriffsregion des "Puffer"-Prozesses keine Nachricht, so kann der Prozeß, dem die Eingabe für seinen weiteren Ablauf fehlt, prinzipiell so lange warten, bis die gewünschte Eingabe erscheint, bis also der in der Hierarchie vorangehende Prozeß eine Ausgabe produziert hat.

Die Überwindung der Inkohärenz durch Abwarten hat jedoch einen Nachteil: Sofern ein Produzent (Sprecher) in seinem Äußerungsfluß innehält und eine längere Zeit schweigt, fühlt sich der Rezipient (Hörer) möglicherweise berufen, das Rederecht zu ergreifen, um selbst die eine oder andere Äußerung zu produzieren. In bestimmten Situationen liegt jedoch der Verlust des Rederechts nicht im Interesse des Produzenten. Dieser steht also in bestimmten Situationen vor dem Problem, das Rederecht nicht zu verlieren, obwohl ein Suchproblem zu überwinden ist. In einer solchen Situation hilft nach Siegman (1979) unter Umständen die Produktion eines "covert

repairs".

Als "covert repairs" gelten nach Levelt (1983) Pausen, Hesitationssignale (vgl. Beispiel (3.11)) und Wiederholungen (vgl. Beispiel (3.12)) bzw. Sequenzen derartiger Äußerungen (vgl. Beispiel (3.13)).

(3.11) einen roten' eh Bauklotz
(Forschergruppe Kohärenz 1987, S. 21)

(3.12) das lin linke Ende
(Forschergruppe Kohärenz 1987, S. 21)

(3.13) wahrscheinlich sind meine Beispiele zu sprunghaft und und und eh ehm zu zu telegraph/
(Talkshow Kinski – Rosenbauer)

Die "Produktion einer Pause" tritt bei der Überwindung eines Suchproplems durch Abwarten auf; die Äußerung von Hesitationssignalen und Wiederholungen gewährleisten besser als eine solche Strategie des Nichtstuns, daß dem Produzenten das Rederecht nicht genommen wird. Nach Beattie (1979) gibt es graduelle Unterschiede zwischen den drei Typen von "covert repairs", die in Kombination mit der These von Siegman (1979) so zu interpretieren sind, daß die Produktion einer Wiederholung effektiver ist als die Produktion eines Hesitationssignals. Diese graduellen Unterschiede sind auch bestimmend für die Regularitäten innerhalb von Sequenzen von "covert repairs". Um den Einsatz der unterschiedlichen "covert repairs" den empirischen Daten entsprechend nachbilden zu können, verfügt das symbolverarbeitende Modell über ein Teilsystem, das eine Einschätzung des jeweiligen Rezipienten über dessen "Verlangen" nach dem Rederecht damit verrechnet, für wie wichtig es der Produzent selbst hält, das Rederecht zu behalten.

Bezogen auf den Faktor "Wunsch nach Rederecht" beinhaltet das symbolverarbeitende Modell also ein internes Modell vom Rezipienten, dessen jeweilige Einträge einen direkten Einfluß auf die Art eines zu äußernden "covert repairs" im Fall eines Suchproblems haben. Die denkbare Inkohärenz, daß eine Fehleinschätzung des Rezipienten durch diese Modellkomponente zur Produktion eines ineffektiven "covert repairs" führt, etwa einer Pause statt eines Hesitationssignals oder sogar einer Wiederholung, so daß dem Produzenten gegen seinen Wunsch das Rederecht verloren geht, kann durch die Modellierung eines Produktionssystems allein natürlich nicht abgebildet werden. Eine denkbare und interessante Erweiterung des hier diskutierten Ansatzes wäre daher die Modellierung eines Gesamtsystems unter dem Fokus des Problemkomplexes "vorliegendes Suchproblem zusammen mit dem Wunsch, das Rederecht zu behalten". Durch die Modellierung eines derartigen Gesamtsystems wäre es dann auch möglich, aus der Perspektive des Außenstehenden die angesprochene Inkohärenz – Verlust des Rederechts aufgrund einer Fehleinschätzung des Rezipienten – als Verletzung der objektiven Kohärenz zu beschreiben.

3.3.4 Vergleich zwischen dem symbolverarbeitenden und dem konnektionistischen Modell

Sowohl das symbolverarbeitende als auch das konnektionistische Produktionsmodell erzeugen "covert repairs" zur Überwindung von Suchproblemen. Dabei läßt sich das konnektionistische Modell relativ einfach um ein Teilnetz, also um ein Teilsystem, ergänzen, daß die Art des zu produzierenden "covert repairs" analog dem symbolverarbeitenden Modell in Abhängigkeit von einer Einschätzung über den Rezipienten bestimmt (Schade & Eikmeyer 1991). Dies liegt in erster Linie daran, daß der abzubildende Anteil des internen Modells sehr geringfügig ist; lediglich eine einzige Eigenschaft des Rezipienten, nämlich sein "Verlangen" nach der Übernahme des Rederechts, muß repräsentiert werden.

Ein geringfügiger, aber für die Anschauung von Kohärenz interessanter und daher wichtiger Unterschied zwischen beiden Modellen liegt dagegen in der Modellierung der Ursache von "covert repairs", dem Suchproblem. Im symbolverarbeitenden Modell ergeben sich Suchprobleme, weil parallel ablaufende Prozesse asynchron kommunizieren (vgl. Abschnitt 3.3.1) und es dabei passieren kann, daß ein in der Hierarchie nachgeordneter Prozeß auf seine Eingabe, die Ausgabe des übergeordneten Prozesses warten muß. Im symbolverarbeitenden Modell fließt also Information stets in eine Richtung der Hierarchie der Teilsysteme; übergeordnete Teilsysteme (Prozesse) steuern die hierarchisch niedrigeren Prozesse (vgl. Abschnitt 2.2.2). Kohärenzherstellung in einem solchen Modell bedeutet bei der Überwindung von Korrektheitsproblemen, also bei der Überwindung von Zuständen, in denen eine Eingabe in ein Teilsystem von diesem nicht akzeptiert wird, entweder eine Änderung im Zustand des Teilsystems, so daß es die Eingabe akzeptiert, oder die Einschaltung eines Metasystems, das das übergeordnete Teilsystem dahingehend beeinflußt, eine für das nachgeordnete Teilsystem akzeptable Eingabe als Ausgabe zu produzieren. Ein direkter Einfluß nachgeordneter Teilsysteme auf ihre Vorgänger ist dagegen nicht vorgesehen.

Im Fall des konnektionistischen Modells entstehen Suchprobleme, wenn sich in einem Teilsystem zum Zeitpunkt einer Selektion (noch) kein stabiler Zustand eingestellt hat. Im Gegensatz zum symbolverarbeitenden System fließt jedoch auch in diesem Fall Information in die hierarchisch niedrigeren Teilsysteme; diese können sogar – da sie ihrerseits auch hierarchisch höher liegende Teilsysteme beeinflussen – zur Stabilisierung beitragen. Entsprechend kann in konnektionistischen Modellen die Kohärenzherstellung zwischen den hierarchisch angeordneten Teilsystemen ohne den Eingriff eines Metasystems gewährleistet werden. Eine Art Metasystem ist jedoch auch in dem konnektionistischen Modell notwendig, um Inkohärenzen, die sich auf die Sequentialisierung der Produktion ausgewirkt haben, zu beseitigen. Das liegt daran, daß die Sequentialisierung mit Selektionen verknüpft sind, die nur durch ein Metasystem rückgängig gemacht bzw. wiederholt werden können.

3.4 Ein inkrementeller Parser in C zur Analyse von simulierten Reparaturen ("repairs")[1]

3.4.1 Empirische und theoretische Voraussetzungen

Der vorliegende Teil verknüpft drei normalerweise weit auseinanderliegende Bereiche,
- die Analyse von syntaktischen Mustern, die in spontaner Rede vorkommen, von sogenannten "repairs" und verwandten Phänomenen,
- das Segmentieren von Wörtern aus unstrukturierten Inputs und
- das Parsing-Paradigma der nicht-deterministischen Parser, d.h. der Parser, die ambige Ausdrücke analysieren können.

Sie betritt sowohl bezüglich der empirischen Daten, die beschrieben werden, als auch in bezug auf die entwickelten Beschreibungsmittel Neuland. Wie bekannt, behandeln normale Syntaxen Konstruktionen der gesprochenen Sprache nicht. Parser-Paradigmen sind üblicherweise ebenfalls nur dazu geeignet, "kanonische" Inputs zu beschreiben. Dagegen versuchen wir, ein Parsing-Paradigma zu entwickeln, das simulierte empirische Daten analysiert.

3.4.1.1 "Repairs" und deren Modellierung in Grammatikparadigmen

Wichtige Anhaltspunkte für die "repair"-Klassifikation haben wir aus den Arbeiten von Schegloff (1979) und Levelt (1983) gewonnen. Unser Ansatz beschränkt sich ausschließlich auf bestimmte Typen von "self-repairs". Zur Orientierung darüber, welche Struktur prototypische "repairs" zeigen, geben wir hier ein Strukturschema aus Levelt (1983, S. 45) leicht verändert[2] wieder.

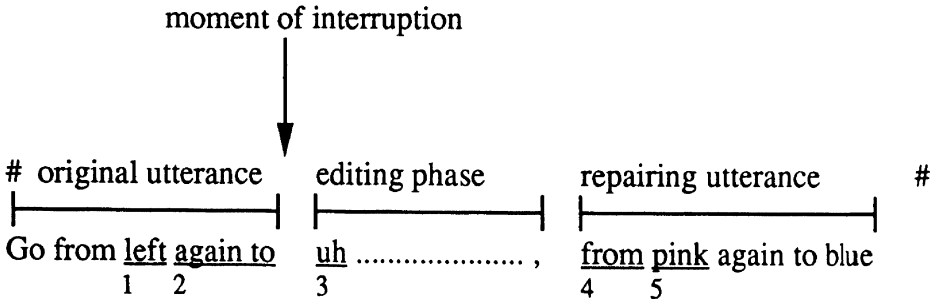
Nach Levelt besteht ein "repair" aus drei Abschnitten. Der erste, die begonnene Äußerung ("original utterance") reicht von der letzten Satzgrenze # bis zum Hinweis, daß ein Äußerungsproblem besteht ("moment of interruption", "editing phase"). Häufig ist der erste Abschnitt daran zu erkennen, daß er eine nicht-wohlgeformte Konstituente enthält, d.h. eine Grammatikregel nicht vollständig realisiert. In der obigen Äußerung fehlt z.B. die NP, die zur Vervollständigung der mit "to" begonnenen Präpositionalphrase nötig wäre. Das partielle Realisieren von Grammatikregeln ist

---

[1] Der vorliegende Parser wurde von S. Lisken und H. Rieser entwickelt. Das "C" in der Überschrift dieses Abschnitts rührt daher, daß der Parser in der Programmiersprache C realisiert wurde. Implementations- und Realisationsgeschichte werden im folgenden vernachlässigt. Vgl. jedoch Lisken und Rieser (1990).

[2] Die Veränderungen bedürften einer längeren Diskussion über die Klassifikation der Komponenten von "repairs", die wir hier aber nicht führen möchten, weil sie für unsere Arbeit nicht wesentlich ist.

bereits ein Hinweis darauf, daß Teile von Reparaturen mit herkömmlichen Syntaxen geparst werden können.



1 = reparandum; 2 = delay (d=3); 3 = editing term; 4 = structural link; 5 = reparans.

An die begonnene Äußerung schließt der zweite Abschnitt an, das Signal dafür, daß ein Äußerungsproblem vorliegt ("editing phase"). Hinweise dieser Art können verschieden realisiert sein, z.B. durch "glottal stop", Pausen oder phonetisches Material. Die Realisierung ist mit Sicherheit sprachspezifisch konventionalisiert. Üblicherweise trifft man in empirischen Daten an dieser Stelle die gesamte Bandbreite der in einer Sprache möglichen Hesitationssignale an. Die beim "editing" verwendeten Äußerungsteile korrespondieren zu keiner normalen Grammatikregel und enthalten oft keine normalen Wörter. Vor dem Hintergrund eines Parsing-Paradigmas stellt sich dieser Abschnitt wie folgt dar: Er kann einerseits als Markierung für eine Unterbrechung des normalen syntaktischen Produktionsprozesses interpretiert, muß aber anderseits durch geeignete Prozeduren segmentiert und isoliert werden.

Der dritte Abschnitt korrespondiert wiederum zu einer grammatischen Regel. Er reicht bis zur nächsten Satzgrenze. Häufig sind der erste und der dritte Abschnitt nach einem analogen grammatischen Muster aufgebaut. Der erste Abschnitt enthält das Reparandum ("left"), der dritte das Reparans ("pink"). Reparandum und Reparans lassen sich nur aus dem Vergleich der Stellen vor und nach dem Anzeigen des Äußerungsproblems feststellen. Liegen im ersten und dritten Abschnitt analoge Grammatikregeln vor, so führt der dritte Abschnitt die im ersten begonnene Grammatikregel zu Ende. So wird im Beispiel oben nach der Formulierung des Reparans die Präpositionalphrase zu Ende produziert. Die strukturelle Analogie zwischen dem ersten und dem dritten Abschnitt wird oft durch die Wiederholung lexikalischen Materials angezeigt (hier "from"). Nicht immer liegt eine Strukturanalogie dieser Art vor, d.h. eine Parsing-Strategie kann sich nicht allein an ihr orientieren. Es muß z.B. auch mit absoluten Neuanfängen gerechnet werden.

Das Beispiel oben zeigt überdies, daß die Äußerung nicht in jedem Fall nach dem Reparandum abbricht (cf. "delay d = 3" [Silben], Anm. der Verf.). In der grammatisch orientierten "repair"-Literatur nimmt die Klassifikation von Typen von "repairs" breiten Raum ein. Für unsere Zwecke ist die Unterscheidung Schegloffs in "prepositioned" bzw. "postpositioned self-repairs" ausreichend. Letztlich bestimmt die

Parser-Architektur, welche Strukturen als wohlgeformt akzeptiert werden. Außerdem sind für einen Parser nur Klassifikationen sinnvoll, die syntaktische Information verwenden. Information anderer Art kann er nicht systematisch verarbeiten.

Nun zum Unterschied zwischen diesen "repair"-Typen:

In "prepositioned repairs" gilt das Anzeigen eines Äußerungsproblems einer Einheit, die noch nicht produziert wurde, es gibt also kein manifestes Reparandum, möglicherweise jedoch ein mentales. In "postpositioned repairs" gilt das Anzeigen eines Äußerungsproblems einem bereits vorhandenen Äußerungsstück. Beide untenstehenden Zeilen geben eine Äußerung wieder, "repair" (A) ist "prepositioned", "repair" (B) "postpositioned":

(A) jetzt kommt eh der grüne
(B) auf das rote [schnell] auf den roten Klotz.

Die von uns verwendeten Daten stammen aus einem Korpus, das im Rahmen der Forschergruppe Kohärenz (cf. Forschergruppe Kohärenz 1987) erhoben wurde. Für die Zwecke der Parserentwicklung haben wir künstliche Beispiele konstruiert, die wesentliche Züge der natürlichen Daten bewahren. Die Idealisierungen betreffen die folgenden Punkte:

- eine natürliche Nicht-Wort-für-Wort-Segmentierung in Äußerungsteile wird als Nicht-Segmentierung approximiert,
- es wird ein "obligatorisches" Hesitationssignal "äh" als "editing term" verwendet,
- die syntaktischen Konstruktionen wurden im Hinblick auf die Erfordernisse der Entwicklung eines allgemeinen nicht-deterministischen Parsers ausgewählt, wobei maßgeblich war, daß Reparaturphänomene bezüglich NP und VP analysiert werden sollten.

Schegloff (1979) verdanken wir nicht nur Information bezüglich vorkommender Typen von "repairs", sondern auch Hinweise darauf, wie eine mögliche Parsing-Strategie aussehen könnte. U.a. im Abschnitt "Syntax or Super-Syntax" führt er aus, daß manche "repair"-Operationen analog zu herkömmlichen Syntaxmustern funktionieren, andere jedoch eher im Sinne einer regulierenden und korrigierenden Metakomponente, welche das Zurückgehen in die normale Syntax ermöglicht. Anhand der Beispiele (A) und (B) kann man sich das wie folgt klar machen: eh in (A) stört die "Syntax-Progression" nicht. Man könnte eh als fakultatives Element jeder normalen Syntaxregel auffassen. In der Tat kann man im empirischen Material solche Hesitationen vor oder nach allen Konstituenten im herkömmlichen Sinn finden. Syntaxregeln dieser Art wären denkbar, deren empirische Interpretation allerdings unklar. Dies würde jedoch für das Parsen keine Probleme aufwerfen, denn der Parser muß nur in der Lage sein, Inputs als wohlgeformt oder nicht-wohlgeformt zu akzeptieren. Dagegen braucht man zur Bestimmung des Zusammenhangs zwischen auf das rote und

auf den roten Klotz einen komplizierten Mechanismus, der folgendes leistet:

a) Realisieren einer Grammatikregel bis zu einer bestimmten Konstituente (in (B) bis zum Erfassen des Nomens),
b) Erkennen, daß ein Abbruch vorliegt, daß also eine Regel nicht zu Ende gebracht wurde,
c) Zurückgehen zu einem nicht als problematisch markierten Teil der Äußerung (hier Beginn der Präpositionalphrase),
d) Wahl einer (i.a. strukturell analogen) Fortsetzung mit neuem Strukturmuster,
e) Markieren der strukturellen Analogie (Präposition auf).[3]

Im wesentlichen wird also von einer Grammatikregel zu einer davon verschiedenen analogen übergegangen. Der Abbruch (in anderen Fällen eine Hesitation) signalisiert, daß ein "Umschalt- oder Regulierungsmechanismus" in Aktion treten wird. Diese Abfolge Regel – Kontrolle/Regulieren – neue Regel kann man sich als von einer Metakomponente gesteuert vorstellen. Sie macht es möglich, "regelhaft" aus einer Syntaxregel "auszusteigen" und eine neue Syntaxregel "anzuwählen", und zwar unter Beachtung gewisser "constraints", wie z.B. dem der strukturellen Analogie.

Überlegt man vor diesem Hintergrund, wie eine Syntax der gesprochenen Sprache aufgebaut werden kann, so erhält man generell und spezifisch für die Analyse von "repairs" zwei konträre Möglichkeiten. Die erste davon ist: Man bemüht sich, die dem Aufbau von "repairs" zugrundeliegenden Regularitäten zu entdecken und Syntaxregeln für "repairs" zu formulieren. Diese Aufgabe kann man wie üblich lösen durch die distributionelle Analyse von "repair"-Konstituenten, die Etablierung von Strukturmustern und schließlich die Entwicklung eines Korpus von Syntaxregeln, welche "repairs" erzeugen oder analysieren. Eine normale Syntax für wohlgeformte Ausdrücke im üblichen Sinne könnte durch Regeln dieser Art ergänzt werden und würde dann z.B. Sätze mit und ohne Reparaturen beschreiben. Mit einem Schlagwort kann man dieses Vorgehen der Erweiterung einer herkömmlichen Grammatik

(I) Grammatisierung von "repair"-Strukturen

nennen. Man stützt sich dabei auf ein neues Konzept von Wohlgeformtheit, Wohlgeformtheit-für-die-gesprochene-Sprache oder ähnlicher. Diese Vorgehensweise wird z.B. auch durch Levelt (1983) nahegelegt, der eine "well-formedness rule" für "repairs" angibt, die wesentlich von der Koordination "und" Gebrauch macht. Dadurch wird suggeriert, daß sich auf einer abstrakten Stufe "und"-Koordination und "non-prepositioned repairs" analog verhielten.

Die dazu konträre Methode ist, das wurde bereits bei der Beschreibung des

---

[3] Was unter "struktureller Analogie" zu verstehen ist, wird weiter unten deutlich, vgl. insbesondere die Ausführungen in den Abschnitten 3.4.2.3 und 3.4.2.2 (Reparatur- und Fragmentanalyse) dazu.

obigen Beispiels herausgearbeitet, eine Metagrammatik zu etablieren, die a) bis e) oben zu leisten in der Lage ist. Ihre Aufgabe besteht darin, den Übergang von nicht-abgeschlossenen Regelfolgen zu anderen (möglicherweise abgeschlossenen) zu steuern. Während in den üblichen Grammatiken jede gewählte Regelfolge zu Ende gebracht werden muß, damit ein wohlgeformter Ausdruck erzeugt/analysiert werden kann, gibt es hier eine neue Möglichkeit des Übergangs von einer Regelfolge zu einer anderen. Das Korpus der normalen Syntaxregeln wird dagegen *nicht* erweitert. Dies sei anhand der folgenden CFPSG (kontextfreien Phrasenstrukturgrammatik) illustriert:

[1] A → B C D
[2] A → B E D
[3] E → F G
[4] B → b
[5] C → c
[6] D → d
[7] F → f
[8] G → g

Im Rahmen einer üblichen syntaktischen Ableitung kann man die Regelfolge <[1],[4],[5],[6]> oder die Folge <[2],[4],[3],[7],[8],[6]> wählen, jedoch nicht <[1],[4],[5],[2],[4],[3],[7]>. Solche Übergänge sollen durch Metaregeln erreicht werden, die zu den normalen Syntaxregeln hinzukommen. Auf diese Art und Weise trägt man dem Tatbestand Rechnung, daß viele Ausdrücke in den empirischen Daten *normalen* Wohlgeformtheitsbedingungen genügen und keinesfalls "abweichen". Man tendiert dazu, Letzteres zu übersehen, wenn man für die Differenz zwischen gesprochener und geschriebener Sprache argumentiert.

Mit einem Stichwort kann man diese Vorgehensweise in Opposition zur ersten

(II) metagrammatische Modellierung von "repair"-Strukturen

nennen. (I) ist natürlich immer im Rahmen von CFPSG erreichbar, wenn auch um den Preis der Einführung von "undurchsichtigen" Konstituenten der Form "repair" und einer Reihe von zusätzlichen Modifikationen, um z.B. Wortfragmente zu erfassen oder mit Rekursionsproblemen fertig zu werden.

(II) ist nach unserer Auffassung nur in prozeduralen Ansätzen modellierbar. Dabei spielen Prozeduren folgender Art, die den empirisch gewonnenen Forderungen a) bis e) oben entsprechen, eine wesentliche Rolle:

- regelkonform verfahren bis zur letzten analysierbaren Konstituente,
- Signal des Wechsels zu einer alternativen Regelfolge erkennen,
- mögliche alternative Regelfolgen berechnen,
- alternative Regelfolge ansteuern,
- in alternativer Regelfolge erneut regelkonform verfahren.

Das Parserformat muß neben normalen Regeln Metaregeln dieser Art enthalten. Die in dieser Arbeit benutzten Metaregeln werden in Abschnitt 3.4.2.2 unter "Reparatur- und Fragmentanalyse" sowie im Abschnitt 3.4.2.3 in den Regeln (S4) und (S5) beschrieben.

3.4.1.2 Der inkrementelle "repair"-Parser und das Kohärenzproblem bei Reparaturen

Wie hängen Reparaturen, Parser-Paradigmen und das Kohärenzproblem zusammen? Reparaturen stellen Lösungen von prototypischen Kohärenzproblemen dar. Sie sind Mechanismen, die auf regelgeleitete Art mit Schwierigkeiten fertig werden, die bei der Produktion von Äußerungen auftreten. Diese "regelgeleitete Art" umfaßt eine Reihe von Aspekten, die nicht alle durch einen syntaktisch orientierten Parsing-Mechanismus erfaßt werden können, jedoch ist für ihre vollständige Beschreibung ein solcher Mechanismus notwendig. Zur Verdeutlichung dieser Auffassung greifen wir auf folgendes Beispiel zurück:

```
B16: 6-16   P:   und den linken- eh Quatsch den roten stellst du links
                 hin'
     6-17   R:   hochkant hinstellen,
```

Die Äußerung ist in einen Dialog zwischen P und R eingebettet, P will das Adjektiv `linken` zu `roten` reparieren. Voraussetzung dafür ist ein auf Semantik und Syntax bezogenes "monitoring". P zeigt die Absicht dadurch an, daß er die an einer syntaktischen Regel orientierte Produktion unterbricht und eine Inkohärenz verursacht, d.h. eine begonnene Syntaxregel (Produktion einer NP) unterbricht und etwas produziert, was unter keine der bestehenden Syntaxregeln subsumiert werden kann.

Dies geht jedoch nicht etwa regellos vor sich, sondern indem er eine ebene Tonhöhenbewegung produziert (`linken-`), eine Hesitation (`eh`) und dann noch einen (in diesem Kontext) metagrammatischen Ausdruck (`Quatsch`). Diese auch regelhaften Dinge zählen wir nicht zur Objekt-, sondern zur Metagrammatik, d.h. der Grammatik, die es erlaubt, mit Problemen fertig zu werden, die auf der Objektebene auftreten und die (nach "Intervention") ein problemloses Weiterfahren auf der Objektebene ermöglicht.

Als Sprecher des Deutschen weiß R, daß P ein Problem hat. Er kann die von P gegebenen Markierungen interpretieren und das Reparans abwarten. Schon eine einzige der gegebenen Markierungen hätte genügt, um diesen Effekt bewirken zu können, Kooperativität vorausgesetzt. R operiert mit einem "monitoring" bezogen auf die Äußerung und reagiert entsprechend den Regeln eines sozialen Systems ("abwarten, was kommt"). Die Auswirkung der Reparatur betrifft letztlich die semantische Seite. Anstatt auf einen links (von etwas) lokalisierten Block will P auf einen roten Block referieren. Diesen roten Block soll R links (von etwas) hinstellen. Korrekte

Referentialisierung ist in diesem Kontext notwendig für korrektes Handeln, da R nach den Anweisungen von P eine Blockwelt nachbauen soll.

Es sind bei einer Reparatur also mindestens folgende Aspekte involviert:
a) Selbstmonitoring bezüglich Syntax und Semantik
b) Fremdmonitoring bezüglich Syntax und Semantik
c) Produktion von Reparaturschemata
d) Analyse von Reparaturschemata
e) Reparatur als Handlung im sozialen System.

Der "repair"-Parser behandelt die syntaktische Seite von a), b) und d) unter Rückgriff auf Arbeiten von Schegloff und Levelt. Ausgeklammert wurde die Semantik. Infolgedessen wird auch nur die *syntaktische* Kohärenzeigenschaft von Reparaturen erfaßt, die, wie im Abschnitt 3.4.1 theoretisch dargelegt und im Abschnitt 3.4.2.2 an einem Beispiel demonstriert, darin besteht, "regelhaft" aus einer Syntaxregel "auszusteigen", einen Problemindikator zu produzieren und eine neue Syntaxregel "anzuwählen", und zwar unter Beachtung gewisser "constraints", wie z.B. dem der strukturellen Analogie.

3.4.1.3 Segmentierung in natürlichen Daten und deren Approximation

Spontane Daten enthalten Segmentierungen, deren genaue Prinzipien noch nicht bekannt sind. Häufig fallen darin die Pausen nicht mit den herkömmlichen Wortgrenzen zusammen, sondern finden sich im Wortinneren, insbesondere nach Vorsilben. Neben Pausen, Betonung und Tonhöhenbewegung spielt auch die Geschwindigkeit, mit der Silbengruppen gesprochen werden, als Strukturierungsprinzip eine wichtige Rolle. Da wir noch keine Vorstellung davon haben, in welche "metrischen" Blöcke Äußerungen gegliedert sind und welches die zulässigen Variationen sind (bei langsamem, betontem Sprechen kann z.B. die Wortgrenze durchaus respektiert werden) haben wir uns dazu entschlossen, einen nicht-segmentierten graphematischen Input als erste Annäherung an die tatsächlich vorkommenden phonetisch-phonologischen Verhältnisse anzunehmen. Wir gingen dabei von der Überlegung aus, daß ein Apparat, der einen nicht-gegliederten Input analysieren kann, sicherlich allgemein genug ist, um einen Input mit stärker ausgeprägten Strukturen zu verarbeiten. Eine stärker ausgeprägte Struktur bekäme man dann, wenn man irgendeine Segmentierung, z.B. eine nach Silbengruppen, vorgäbe. Diese Entscheidung würde nur zu einem etwas anderen Einlesemechanismus führen, erst einmal bei der Wortebene angekommen,

würde ein solcher Parser genauso arbeiten können, wie der hier beschriebene.[4]

Im Gegensatz zur "repair"-Modellierung, für die es noch keine anerkannten methodologischen Regeln gibt, stellt die Segmentierung von Äußerungen in Wörter ein Gebiet dar, für das zahlreiche Forschungsansätze, vor allem auf dem Gebiet der Phonologie, existieren. Wir orientieren uns hier an einer Liste von experimentell erhärteten Vorgaben, die in Cole and Jakimik (1980) abgedruckt ist. Die beiden Autoren gehen davon aus, daß die Segmentierungsleistung von Sprechern durch die folgenden Annahmen charakterisiert werden kann:

1. Wörter werden durch die Interaktion von phonetischer Information und Wissen segmentiert.
2. Die Wörter in einer Äußerung werden nacheinander segmentiert.
3. Die Segmentierung eines Wortes liefert syntaktische und semantische "constraints", die zur Segmentierung des nachfolgenden Wortes verwendet werden.
4. Die Segmentierung eines Wortes lokalisiert die Laute, welche das folgende Wort beginnen.
5. Die Laute eines Wortanfangs werden benutzt, um Wort-Kandidaten zu ermitteln.
6. Die Laute eines Wortes werden sequentiell verarbeitet.
7. Ein Wort ist bewußt segmentiert, wenn die sequentielle Analyse der Laute alle möglichen Wortkandidaten bis auf einen eliminiert hat.

Bezüglich dieser Liste müssen wir eine Reihe von Einschränkungen machen. Die wichtigste davon ist, wie bereits betont, daß wir "nur" graphematische Inputs verarbeiten können. Außerdem enthält unser Parser keine Semantik-Komponente. Für ein erstes Modell eines "repair"-Parsers wäre eine nicht-triviale Integration der Semantik zu kompliziert gewesen. Hinzu kommt, daß im Rahmen der herkömmlichen Parserformate und Parsing-Strategien wenig über die Interaktion von Syntax und Semantik bekannt ist. Ein vernünftiger Kenntnisstand wäre z.B. dann erreicht, wenn die semantische Information konstruktiv in den Parsingprozeß einbezogen werden könnte, mit anderen Worten, wenn man wüßte, wie semantische Information bei der Lösung bekannter Parsing-Probleme heranzuziehen ist. Dies wäre z.B. dann gegeben, wenn die Kenntnis der beschriebenen Situation und deren "Konstituenten"

- die Segmentierungsaufgabe erleichtern, d.h. den Lexikonzugriff und den Wortfindungsprozeß beschleunigen,
- die Reduktion von Ambiguität syntaktischer Art ermöglichen,
- die Wahl zutreffender anaphorischer Bezüge gestatten,
- Abkürzungsverfahren für die syntaktische Analyse bereitstellen oder

---

[4] Ein ähnliches Projekt wird in Tomita (1987, S. 45) unter dem Stichwort "[parsing of] unsegmented sentences (with no spaces between words, typical in Japanese)" erwähnt.

– die Syntax auf ein Minimum beschränken würde. Dies ist jedoch beim gegenwärtigen Kenntnisstand nicht der Fall.[5] Infolgedessen beschränken wir uns bei der obigen Prämisse 1. auf die Ausnützung der Interaktion von phonetischer (d.h. hier graphematischer) Information und syntaktischem Wissen, ebenso wie unter 3. Konkret stellt sich dies in unserem Ansatz so dar: Das erste Wort wird unter Abgleich mit den Worteinträgen des Lexikons ermittelt. Dessen grammatische Charakteristika (Haupt- und Nebenkategorien) werden festgestellt. Daraufhin werden alle Syntaxregeln gesucht, die dieses Wort parsen. Dabei ist zu beachten, daß der Input mehrere Segmentierungen zulassen und, als Folge davon, mehrere Anfangswörter ergeben kann.

Durch "Weiterschalten" in den passenden Syntaxregeln (falls mehrere vorliegen) wird die Kategorie des nachfolgenden Wortes bestimmt. Diese Information wird für die Segmentierung des nächsten Wortes benutzt. Schlägt die Segmentierung fehl, liegen unter Umständen Reparaturen oder Wortfragmente vor. Mit anderen Worten verwendet der Parser für sämtliche Segmentierungen bis auf die des ersten Wortes eine "kategoriale Hypothese", d.h. eine Vorgabe für das nächste zu segmentierende Wort bezüglich Haupt- und Nebenkategorien. Verschiedene kategoriale Hypothesen ergeben sich durch die Verfolgung verschiedener Parsing-Alternativen im Falle von ambigen Ausdrücken.

Forderung 4. ergibt sich aufgrund der Anlage der Segmentierungsprozedur, 5. und 6. sind durch die Strukturierung des Lexikons in einem "Lexikonbaum" erfüllt (s. im Abschnitt 3.4.2.1). 7. ist für unsere Zwecke in dieser Form zu wenig allgemein. Es bezieht sich nur auf eine mögliche Weise, einen Input zu segmentieren. Wie bereits oben bemerkt, gehen wir dagegen davon aus, daß ein Input auf mehrere Arten segmentiert werden kann, mit anderen Worten, das Problem der Parsing-Alternativen (Nondeterminismus-Problem) stellt sich bereits auf der Ebene der Segmentierung, und für *jede dieser Segmentierungen* muß 7. erfüllt sein. Vgl. dazu auch Abschnitt 3.4.1.4.

3.4.1.4 Natürliche und simulierte Daten

Im Anschluß an den vorhergehenden Abschnitt, der das Verhältnis von idealisierten und natürlichen, im Experiment gewonnenen Daten behandelt, wollen wir zeigen, welche Typen von nicht-segmentierten Inputs der Parser erfolgreich analysiert und zu welchen Belegstellen im Korpus diese Inputs korrespondieren. Der Parser ordnet allen unten aufgeführten Inputs eine Strukturbeschreibung in dem im Beispiel in 3.4.2.2

---

[5] Die Integration der Semantik in Parser beschränkt sich im allgemeinen darauf, daß das Resultat, eine syntaktische Struktur, semantisch interpretiert wird. Dabei kann die Semantik höchstens als Filter verwendet werden. Diese Rolle von Semantik entspricht jedoch nicht der oben dargelegten Auffassung bezüglich ihrer Funktion beim Parsen.

beschriebenen Sinne zu. Unter den benutzten idealisierten Inputs sind jeweils die Belegstellen aus dem Korpus angegeben. Der idealisierte Input *simuliert* Vorkommen von Reparaturen, wie sie die Belegstellen prototypisch zeigen. Man beachte, daß es nicht Ziel des Projekts war, die gesamte Syntax der natürlichen Daten zu modellieren.

Der Kommentar zu den Belegen setzt die eingangs getroffene Unterscheidung in "prepositioned" und "postpositioned repairs" voraus.

I. "prepositioned repairs":

pre_pos.1 : deraehkleinemannschlaeft

```
B1:
1-64    P:   nein er steht ganz der die Säulen stehen praktisch im
1-65         Hintergrund' und der eh . grüne eh ragt nur etwas .
             hinter
1-66         die . durch die beiden eh roten Säulen durch,
1-67    R:   mhm mhm
```

Die relevante Reparatur befindet sich in der NP `der eh . grüne.`

pre_pos.2 : derkleineaehmannschlaeft

```
B2:
1-49    P:   und jetzt is eh oben auf dem . grünen . beziehungsweise
1-50         blauen viereckigen eh Säulenteil is oben noch ein gelbes
1-51         halbrundes . etwas mondförmiges Dach
```

Die relevante Reparatur befindet sich in der NP
`dem . grünen . beziehungsweise blauen viereckigen eh Säulenteil.`

pre_pos.3 : derkleinemannaehschlaeft

Wie B1. Die relevante Reparatur befindet sich zu Beginn der VP
`eh ragt nur etwas . hinter die . durch die beiden eh roten Säulen durch.`

pre_pos.4 : derkleinemannschlaeftruhigaehundtief

```
B4:
2-30    R:   nee, nich so richtig (Schmunzeln) ach so Moment mal ..
             also
2-31         der is nich richtig viereckig,
2-32    P:   nee nee der is eh länglich
```

Die relevante Reparatur steht vor dem Prädikativum der VP in
`nee nee der is eh länglich.`

II. Wortfragmente im Zusammenhang mit "prepositioned repairs":

pre_pos.1f1 : derkleikleinemannschlaeft
pre_pos.1f2 : deraehkleikleinemannschlaeft
pre_pos.2f1 : derkleinemamannschlaeft
pre_pos.2f2 : derkleineaehmamannschlaeft
pre_pos.4f1 : derkleinemannschlaeftruhigunundtief
pre_pos.4f2 : derkleinemannschlaeftruhigaehunundtief

```
B5:
5-248   R:   also . hm . also ich so/ soll jetzt ich leg jetzt einfach
5-249         mal so ne sechs vom Würfel,
5-250   P:   ja

B6:
1-94    P:   und jetzt müßteste . nja gleichzeitig . eh . auf die linke
1-95         Seite auf das lin linke Ende dieses blauen Steins' einen
1-96         roten' . eh Bauklotz setzen' n roten viereckigen Bauklotz
1-97         derselben Länge wie im hinteren Teil die vorkommen' also
1-98         links nen roten' und rechts einen gelben,
```

B5 zeigt ein Wortfragment in der VP `so/ soll jetzt` , B6 in der NP `das lin linke Ende`.

III. "postpositioned repairs"

Häufig sind im empirischen Material Korrekturen im intuitiven Sinn, bei denen ein analoges Strukturmuster gewählt wird. Auf der idealisierten Input-Seite für den Parser haben wir:

post_pos.10 : dieaehderkleinemannschlaeft
post_pos.11 : diekleineaehderkleinemannschlaeft
post_pos.12 : diekleinefrauaehderkleinemannschlaeft
post_pos.13 : diekleineundtapferefrauaehderkleinemannschlaeft

Unter den empirischen Belegen finden sich:

```
B7:
1-109   P:   jaa . wobei die Kanten mit dem unteren ziemlich genau
1-110         abschließen, (3 sec. Pause) also steht nich ver/ dieser
1-111         diese Würfel stehn nich versetzt, auf die rechte Seite
              wie
1-112         gesagt nen grüner, . Würfel,
1-113   R:    links n gelber und rechts n grüner,
1-114   P:    genau, (4 sec. Pause)

B8:
1-64    P:    nein er steht ganz der die Säulen stehen praktisch im
1-65          Hintergrund' und der eh . grüne eh ragt nur etwas .
              hinter
```

```
1-66          die . durch die beiden eh roten Säulen durch,
1-67    R:    mhm mhm

B9:
4-146   P:    (leise) ja sonst wirds schwierig, + ja und dann setzte
              von
4-147         dir aus gesehn rechts diese dieses blaue . Dingen drauf
              und
4-148         | links . das grüne, waste vor/ vorher als . als rot und
              gelb
4-149   R:    |                      mhm
4-150   P:    stehen hast,
4-151   R:    mhm

B10:
7-22    P:    mhm dann stellst du' .. eh . also du hast jetzt dem den
7-23          roten vor den blauen gelegt'
7-24    R:    vor den blauen'
7-25    P:    ja . vor den blauen' .
```

Sie alle zeigen die Korrekturen in NPs, cf. dieser diese Würfel (B7), der die Säulen (B8), diese dieses blaue . Dingen (B9), dem den roten (B10).

Die nachfolgenden idealisierten Inputs variieren die Stellung der Reparatur an Konstituentengrenzen. Im wesentlichen testen sie die Leistungsfähigkeit von (S3) und (S4) aus Abschnitt 3.4.2.3.

post_pos.1 : deraehderkleinemannschlaeft
post_pos.2 : derkleineaehderkleinemannschlaeft
post_pos.3 : derkleinemannaehderkleinemannschlaeft
post_pos.3a : derkleinemannaehaehderkleinemannschlaeft
post_pos.4 : derkleineundaehderkleinemannschlaeft
post_pos.5 : derkleinemannschlaeftruhigaehruhigundtief
post_pos.6 : derkleineundtapfereaehkleinemannschlaeft
post_pos.7 : derkleineaehkleineundtapferemannschlaeft
post_pos.8 : derkleinemannschlaeftaehschlaeftruhigundtief
post_pos.9 : derkleinemannschlaeftruhigundtiefaehschlaeft

```
B11:
6-57    P:    eh und dann der blaue kommt auf den gelben Klotz und der
6-58          | ro/ em gelbe kommt auf den . eh blauen Klotz
6-59    R:    |                                        ja

B12:
5-248   R:    also . hm . also ich so/ soll jetzt ich leg jetzt einfach
5-249         mal so ne sechs vom Würfel,
5-250   P:    ja

B13:
1-12    R:    mhm und die dadraufstehenden de sind der gleichen . der
```

1-13 | gleichen Länge auch der gleichen Ff
1-14 P: | haben dieselbe Länge sind aber viereckig'.
1-15 wobei das linke . n grünes is und das rechte n ro eh n blaues
1-16 is .. die beiden Säulen stehen unten eh ein Stück ausnnander

B15:
5-161 P: gut und jetzt legste auf diese blauen Klötzchen an ihren
5-162 Enden an & die habn ja zwei Enden'
5-163 R: mhm &
5-164 P: rechts und links je ein blau/ ein grünes Klötzchen n
5-165 viereckiges

B16:
6-16 P: und den linken- eh Quatsch den roten stellst du links hin'
6-17 R: hochkant hinstellen,

B17:
2-30 R: nee, nich so richtig (Schmunzeln) ach so Moment mal .. also
2-31 der is nich richtig viereckig,
2-32 P: | nee nee der is eh länglich
2-33 R: | ach so ja jetzt hab ichs jaja mhm' okay
2-34 P: | also und hat hat ne breite Seite und ne schmale Seite, ja'
2-35 R: | mhm ja

B18:
5-90 P: also ich fang nochmal von vorne an (schneller) damit wir mal
5-91 sehen ob du das so genauso hast, + also sechs Klötzchen grün
5-92 die stehen auf ihrer- . Querseite, auf der breiten auf der .
5-93 stehen hochkant,

5-253 R: dann hab ichs aber nich mehr waagerecht vor mir dann hab ich
5-254 se nich . dann hab ich se dann

B19:
1-94 P: und jetzt müßteste . nja gleichzeitig . eh . auf die linke
1-95 Seite auf das lin linke Ende dieses blauen Steins' einen
1-96 roten' . eh Bauklotz setzen' n roten viereckigen Bauklotz
1-97 derselben Länge wie im hinteren Teil die vorkommen' also
1-98 links nen roten' und rechts einen gelben,

B20:
4-28 P: gut, das habn wir jetzt ne' das wär vorn, jetzt von dir aus

4-29 | gesehn hinten' . stellst du neben das grö
4-30 R: | halt warte nochmal (kurzes
4-31 Lachen) ich hab doch dien den grünen' . Klotz hochkant
4-32 gestellt und da oben drauf den blauen . langen
4-33 P: ja hochkant jetzt daß die schmale Seite, . die längere
4-34 | schmale Seite, ... oben is ne' also nich daß du das eh ganz
4-35 R: | ach jetzt
4-36 P: | hochkant stellst sondern nur
4-37 R: | weiß ich glaub ich erst welches du meinst (5 sec. Pause)

B21:
7-39 P: jetzt legst du auf den blauen der is ja ganz . flächig
7-40 R: mhm
7-41 P: einen grünen Stein der dann eben dieses kleine Viereck &
7-42 also das is praktisch n Quadrat, also n grünes Quadrat,
7-43 R: mhm
7-44 P: legst du da- direkt neben den gelben,
7-45 R: mhm .

B22:
5-69 P: ja die mußte nachher so verbinden daß die . die zwei blauen
5-70 langen Klötze die werden nachher draufgelegt
5-71 R: mhm
5-72 P: so daß also so ne Brücke entsteht

B23:
5-221 R: ehm können wir vielleicht zjetz fm mach ich das andere mit
5-222 mit Zentimetern oder so, also ich hab hier n grünen Klotz
5-223 stehn links außen'
5-224 P: ja

Eine Reparatur nach dem ersten Element einer Konstruktion (post_pos.1) zeigt die NP in B22 daß die . die zwei blauen langen Klötze. Bei B22 liegen die Verhältnisse aber erheblich komplizierter als im obigen Input, da auch das Satzmuster geändert wird (Nebensatz mit Konjunktion "daß" zu Hauptsatz). Zurückgehen zum Konstruktionsanfang nach Produktion des ersten Elements finden wir auch in der Präpositionalphrase mit mit Zentimetern oder so von B23.

Wiederholung eines größeren Abschnitts einer NP, parallel zu post_pos.2, hat die NP der gleichen . der gleichen Länge in B13. B13 hat auch eine Reparatur des Anfangs einer VP sind der gleichen . der gleichen Länge auch der gleichen Ff haben dieselbe Länge sind aber viereckig', B12 zeigt eine Reparatur eines Satzanfangs ich so/ soll jetzt ich leg jetzt einfach mal so ne sechs vom Würfel. Korrektur einer VP modelliert post_pos.8, Wiederholung eines Satzanfangs simulieren post_pos.3 und post_pos.3a.

Häufig ist in den Daten die Reparatur eines Adjektivs in einer NP, cf.

n ro eh n blaues (B13),
den linken- eh Quatsch den roten (B16),
einen roten' . eh Bauklotz setzen' n roten viereckigen Bauklotz (B19),
die schmale Seite, . die längere schmale Seite (B20),
n Quadrat, also n grünes Quadrat (B21).

Vorkommen dieses Typs decken wir z.B. durch die etwas komplizierter strukturierten post_pos.4 und post_pos.6 ab. Eine Reparatur im Bereich des Adverbialkomplexes modellieren post_pos.5 und post_pos.9. In den Daten findet sich ein Beispiel dafür in der VP des indirekten Fragesatzes von B18 ob du das so genauso hast.

3.4.2 Funktionsweise des Parsers

In den folgenden Abschnitten beschreiben wir die Funktionsweise des Parsers von zwei verschiedenen Standpunkten aus. Der Abschnitt 3.4.2.2 "Beispiel- und implementationsgebundene Funktionsbeschreibung" stellt die Analyse eines grammatisch ungestörten und eines "gestörten" (reparierten) Inputs in der zeitlichen Reihenfolge des Analysevorgangs vor. Die verwendeten Bearbeitungsschritte werden hier informell, anhand der vom Programm manipulierten Datenobjekte vorgeführt. Diesem Abschnitt vorangestellt ist (als Abschnitt 3.4.2.1) eine Einführung in diese Datenobjekte, welche linguistische Entitäten repräsentieren (Lexikon bzw. Lexikonbaum, Syntax, Parsing-Baum als Repräsentation des üblichen Syntaxbaums) oder Verwaltungsfunktionen des Parsers übernehmen (Parsing-Baum, active rules). Der andere Standpunkt, von dem aus die Arbeitsweise des Parsers beschrieben wird, findet sich in Abschnitt 3.4.2.3 "Abstrakte Funktionsbeschreibung des 'repair'-Parsers". Dort werden Regeln angegeben, die auf Entitäten wie Parsing-Alternativen, Syntaxregeln, Lexikonelementen und dem Input operieren und einen Parsing-Vorgang für die hier behandelten und ähnliche Input-Phänomene beschreiben. Wir zeigen dort auch, wie das Ergebnis des Parsings in einer baumartigen Struktur repräsentiert werden kann. Das Parser-Programm modelliert die in der abstrakten Beschreibung angegebenen Regeln. In der abstrakten Funktionsbeschreibung verzichten wir jedoch bewußt darauf, den Zusammenhang zu Objekten oder Arbeitsschritten des Parsers (z.B. Baumkonstruktion – Manipulationen des Parsing-Baums) anzugeben.

Es ist auch möglich, sich beim Lesen entweder auf die abstrakte Beschreibung (3.4.2.3) oder die (detailliertere) Beispiel- und implementationsgebundene Beschreibung (3.4.2.1, 3.4.2.2) zu beschränken.

3.4.2.1 Verwendete Datenstrukturen

Wörter

Das Lexikon, das der Parser verwendet, befindet sich als sequentielle Liste von Wörtern in einem File. Zu jeder Wortform werden die Hauptkategorie und einige von zehn möglichen Nebenkategorien angegeben.[6] Die in den Haupt- und Nebenkategorien jeweils möglichen Werte werden durch Zahlen kodiert. Die Kategorienangaben werden zu einer Kategorienstruktur innerhalb der Wortstruktur zusammengefaßt. Diese wird mit einer Graphemkette für die Wortform zu einer Wortstruktur zusammengefaßt.

Beispiel für einen Worteintrag (Erläuterungen in Klammern):

schlaeft, Wortart 5	(intransitives Verb)
Person 3	(3. Person)
Numerus 1	(Singular)
regierter Kasus 0	(Verbvalenz 0)
Tempus 1	(Präsens)
Modus 1	(Indikativ)

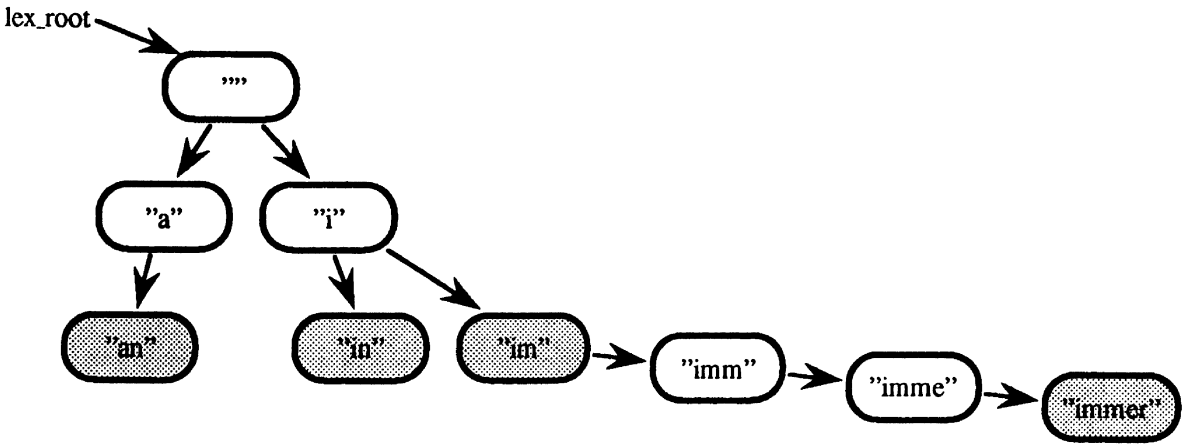
Der Lexikonbaum

Zum effizienteren Durchsuchen des Lexikons und zur Erfüllung der Forderungen 5. und 6. nach Cole und Jakimik (vgl. Abschnitt 3.4.1.3) werden die sequentiell eingelesenen Wörter in eine Baumstruktur überführt. Einem Knoten des Lexikonbaums entspricht eine Graphemsequenz, die Anfang von Wortformen im Lexikon sein kann (anderen Graphemsequenzen entspricht kein Knoten im Lexikonbaum). In jedem Knoten sind nun die Grapheme vermerkt, die (nach Maßgabe der Wortformen im Lexikon) mögliche Fortsetzung der zu diesem Knoten gehörenden Graphemsequenz sind. Zu jedem dieser Grapheme findet sich auch ein Verweis auf einen Knoten der nächsttieferen Ebene, der die entsprechend verlängerte Graphemsequenz repräsentiert. Falls die in einem Knoten repräsentierte Sequenz schon vollständige Form eines Wortes ist, findet sich im Knoten zusätzlich noch ein Verweis auf die Wortinformation, die in einer Kategorienstruktur abgespeichert ist, wie sie oben unter "Wörter" vorgestellt wurde.

---

[6] Die Benennung der Nebenkategorien sowie die Frage, welche Nebenkategorien in welcher Reihenfolge für eine gegebene Hauptkategorie erwartet werden, lassen sich in der gegenwärtigen Implementierung nicht von außerhalb des Programms beeinflussen. Das Lexikonformat muß auf die Voreinstellungen des Programms ausgerichtet werden.

Beispiel für einen Lexikonbaum mit den Wörtern "an", "in", "im", "immer":[7]



Dieser Baum wird dann in der Segmentierungsprozedur[8] verwendet: Parallel zum Einlesen von Graphemen aus dem Input und gesteuert von diesen Graphemen geht der Parser im Lexikonbaum "abwärts", bis alle möglichen Segmentierungen gefunden sind. Die "Wanderung" beginnt an der Wurzel des Lexikonbaums, in der die leere Graphemsequenz repräsentiert ist. Dann wird jeweils gefragt, ob das nächste Graphem im Input als Nachfolgebuchstabe im aktuellen Knoten des Lexikonbaums vermerkt ist. Falls ja, wird der entsprechende nächsttiefere Knoten angesteuert und die Frage wiederholt. Bei jedem Schritt wird außerdem gefragt, ob bereits ein Worteintrag erreicht ist. Sofern eine kategoriale Hypothese vorliegt (hierzu s. die Erläuterungen am Ende von Abschnitt 3.4.1.3), werden nur solche Worteinträge betrachtet, die mit dieser Hypothese übereinstimmen. Der Vorgang setzt sich solange fort, wie die vorliegenden Inputgrapheme es zulassen. Auf diese Weise werden alle "alternativen Segmentierungen", die von der aktuellen kategorialen Hypothese zugelassen werden, gefunden (alternative Segmentierungen: zwei Wörter, Form des einen ist gleich dem Anfang der Form des anderen; also Homonyme oder auch Wortpaare wie "im"/"immer"). Jede dieser alternativen Segmentierungen wird anschließend in einer eigenen Parsing-Alternative verfolgt (vgl. hierzu a.a.O., außerdem "active rules" in diesem Abschnitt oder die abstrakte Funktionsbeschreibung, i.e. Abschnitt 3.4.2.3).

---

[7] Die Lexikonbaumknoten sind hier mit den repräsentierten Graphemsequenzen beschriftet. Falls es sich bereits um eine vollständige Wortform handelt, ist der entsprechende Knoten mit einem Raster unterlegt.

[8] In diesem Absatz beschreiben wir den Einsatz des Lexikonbaums für die Segmentierung von Wörtern aus dem Input. Für die Einbettung dieser Prozedur in den Gesamtvorgang des Parsings vgl. Abschnitt 3.4.2.3, Strategien S0) und S1) (Segmentierung ohne bzw. mit kategorialer Hypothese) oder den Ablauf des Beispiel-Parsings in Abschnitt 3.4.2.2.

Die Syntax

Die Syntaxregeln werden dem Parser in einer Form analog zum Wortbeispiel oben mitgeteilt, d.h. sowohl für die Haupt- wie für die Nebenkategorien werden die möglichen Werte für die jeweilige Kategorie in Zahlen codiert. Eine Syntaxregel besteht aus einer Kategorienstruktur für die linke und (n) Kategorienstrukturen für die rechte Regelseite ((n) bezeichnen wir als Regellänge). Es werden keine weiteren Informationen, wie sie etwa für kontextsensitive Grammatiken notwendig wären, abgespeichert.

Die komplette Syntax wird durch eine Liste von Syntaxregeln der beschriebenen Art repräsentiert. Eine Hilfsprozedur im Programm vergleicht zwei Kategorienstrukturen auf Gleichheit. Die beiden Kategorienstrukturen können aus der Syntax oder dem Lexikonbaum stammen, es kann sich auch um die kategoriale Hypothese handeln, die während des Parsings verwendet wird.

Der Parsing-Baum

Das Konzept des Parsing-Baums wurde bereits zu Beginn des Abschnitts 3.4.2 eingeführt. Der Parsing-Baum entspricht einerseits dem in der Linguistik üblichen Syntax-Baum, wie er typischerweise als Endprodukt eines Syntaxparsers zu erwarten wäre, andererseits übernimmt er die Verwaltungsfunktion von "charts" oder "tables" anderer Parser (vgl. Abschnitt 3.4.3) und repräsentiert Parsing-Alternativen. Die Verwaltung des Parsings leistet der Parsing-Baum im wesentlichen dadurch, daß er inkrementell aufgebaut wird. Eine Grundidee der Implementierung ist, daß Probleme wie das Parsing angesehen werden können als die stetige Anpassung einer Struktur an den Programminput. Das Ergebnis eines solchen Prozesses ist dann die fertige Struktur. Im Falle des Parsing-Baums mußte das Konzept des "normalen Syntaxbaums" allerdings etwas "angereichert" werden. Wir beschriften Knoten des Parsing-Baums nicht mit syntaktischen Kategorien, sondern mit Syntaxregeln. Diese Idee ist üblicherweise mit jedem Syntaxbaum verknüpft, denn ein Knoten der Kategorie A in einem solchen Baum, der Knoten der Kategorien B, C und D dominiert, setzt das Vorhandensein einer Regel $A \rightarrow B\ C\ D$ in der zugrundeliegenden Syntax voraus. Ein solcher Knoten ist bei uns mit eben dieser Regel beschriftet. In verkürzender Weise bezeichnen wir A als die Kategorie eines solchen Knotens. Unter einem "A-Knoten" verstehen wir einen Parsing-Baum-Knoten, der mit einer beliebigen A expandierenden Regel beschriftet ist. Es gibt übrigens noch eine Art von Knoten, die nicht mit Regeln, sondern mit Worteinträgen aus dem Lexikon "beschriftet" sind und die wir "Wortknoten" nennen. Diese Knoten werden üblicherweise durch einen Knoten mit "präterminaler" Kategorie und ein daran "angehängtes" Terminal repräsentiert. Dahingegen sind im Parsing-Baum Terminals in das Knotenkonzept integriert, weil wir Wortknoten mit der lexikalischen Regel Wortkategorie → Terminal beschriften; wir bezeichnen die Kategorie des Wortes auch als Kategorie des Wortknotens.

In einem "unfertigen" Parsing-Baum müssen jedoch von den oben genannten Knoten der Kategorien B, C und D nicht alle vorhanden sein. Ist nur der B-Knoten vorhanden und die Analyse des Inputs mit der Regel A → B C D noch nicht verworfen, so wäre die Kategorie C Hypothese für den nächsten Analyseschritt. Um diese Abfolge zu kontrollieren, enthält jeder Parsing-Baum-Knoten einen Zähler für die aktuelle Position innerhalb der rechten Seite der von ihm repräsentierten Regel. Wir bezeichnen diesen Zähler als Index des Knotens.[9] So repräsentiert ein Parsing-Baum-Knoten mit seinen dominierenden Knoten und den Indizes dieser Knoten den Stand einer Parsing-Alternative (innerhalb der Syntax).

Verschiedene Parsing-Alternativen befinden sich jedoch an verschiedenen Stellen in der Syntax. Jede Alternative benötigt deshalb ihren eigenen Arbeitsbereich im Parsing-Baum. Einige Alternativen unterscheiden sich bereits in der Wurzel, d.h in der verwendeten Satzregel oder der aktuellen Position in deren rechter Seite. Andere Alternativen sind erst entstanden, nachdem eine Alternative eine nicht-terminale Kategorie K zu parsen hatte, wofür verschiedene Regeln in Frage kamen. Es entsteht dann eine Parsing-Alternative für jede solche K-Regel; diese Regeln sind aber in gleicher Weise in dominierende Regeln eingebettet. Es gibt also auch in Alternativen einen "Verzweigungspunkt" in der Regelhierarchie, unterhalb dessen sie sich unterscheiden. Diese Verzweigung ist im Parsing-Baum repräsentiert, indem statt eines einzelnen Knotens an jeder Stelle des Parsing-Baums eine Gruppe untereinander verbundener, "alternativer" Knoten stehen kann. Diese Knoten haben stets gleiche Kategorien, aber nicht gleiche Regeln zur Expansion dieser Kategorie. Der Parsing-Baum verzweigt also in zwei "Dimensionen": der syntaktischen (A verzweigt in B, C und D) und der "alternativen" Dimension (z.B. C ist nicht-terminale Kategorie mit mehreren Expansionsmöglichkeiten). (Für weitere Erläuterungen hierzu vgl. "active rules" in diesem Abschnitt und die "Baumkonstruktionskonvention" in Abschnitt 3.4.2.3.) Die Stellung von alternativen Knoten im Parsing-Baum ist jedoch nicht ganz unbeschränkt; um dies zu sehen, müssen wir jedoch erst untersuchen, wie eine Parsing-Alternative in unserem Parser repräsentiert ist.

Jede Parsing-Alternative muß außer einem Bereich im Parsing-Baum (dem oben erwähnten "Arbeitsbereich") den Input abarbeiten. Für jede aktuell arbeitende Alternative gibt es deshalb eine "active rule" genannte Struktur, die im Parsing-Baum den zu bearbeitenden Knoten (den "aktiven Knoten") und im Input das nächste abzuarbeitende Graphem markiert. Im Parsing-Baum ist die Stellung von alternativen Knoten nun dadurch beschränkt, daß nur "inaktive" alternative Knoten an beliebiger Stelle auftauchen; "aktive" Knoten, i.e. jene, die die Verwaltungsfunktion ausüben, be-

---

[9] Wir verwenden den Index eines Parsing-Baum-Knotens auch als Markierung für Wortknoten: Da Wortknoten keine Syntaxregeln durchzählen müssen, kann ihr Index einen zu diesem Zweck reservierten negativen Wert annehmen.

schränken sich auf einen bestimmten Teil des Baumes. Eine genaue Erläuterung dieses Sachverhaltes und des "Aktivitäts"-Begriffes findet sich im folgenden Abschnitt über "active rules".

active rules

Der im vorigen Abschnitt vorgestellte Parsing-Baum ist wie gesagt nicht nur Ergebnis, sondern durch seinen inkrementellen Aufbau auch "Verwalter" des Parsings. Eine Parsing-Alternative versucht ständig, dem Parsing-Baum ein aus dem Input zu segmentierendes Wort hinzuzufügen, wobei durch Regel und Index eines Knotens im Parsing-Baum eine kategoriale Hypothese angegeben wird.[10]
Dieser Knoten ist es auch, der den bei erfolgreicher Segmentierung entstehenden Wortknoten unmittelbar dominieren wird. Wir bezeichnen diesen für einen Parsing-Schritt wichtigen Knoten als "aktiven Knoten"; eine Parsing-Alternative wechselt im Laufe ihrer Arbeit mehrmals den aktiven Knoten und läßt dabei (nicht mehr aktive) Knoten im Parsing-Baum zurück, die später insgesamt die Parsing-Geschichte darstellen.

Jede Parsing-Alternative benötigt also über den Parsing-Baum hinaus einen Pointer auf einen Parsing-Baum-Knoten, der (für diese Alternative) den jeweils aktiven Knoten markiert. Außerdem führt die Verfolgung verschiedener syntaktischer Analysen in der Regel zu verschiedenen Segmentierungen des Inputs; aus diesem Grunde muß jeder Parsing-Alternative ein Zähler für die im Input erreichte Position zugeordnet werden. Der Input ist eine Graphemkette, der erwähnte Input-Zähler gibt die Anzahl bereits analysierter Grapheme an. Wir fassen den Pointer auf den aktiven Parsing-Baum-Knoten und den Input-Zähler in einer Struktur namens "active rule" zusammen. Eine active rule repräsentiert somit eine Parsing-Alternative, solange diese verfolgt wird.

Bei Vorhandensein eines "Graphemzählers" in jeder active rule liegt die Frage nahe, ob nicht auch ein "Syntaxzähler" nötig wäre, ein Zähler also z.B. für die Anzahl geparster syntaktischer Konstituenten und damit für die Position der Parsing-Alternative innerhalb der von ihr verfolgten syntaktischen Analyse. Es ergibt sich jedoch für eine einfache Aufaddierung der geparsten Konstituenten keine technische Notwendigkeit. Was benötigt wird, ist eine Verwaltung der Position innerhalb der rechten Seite der aktuell bearbeiteten Regel, wie sie durch den Index unserer Parsing-Baum-Knoten realisiert wird. Wenn eine Regel beendet ist, muß in die dominierende Regel zurückgekehrt und die Position in der rechten Seite dieser Regel

---

[10] Zur genauen Definition dieses Zusammenhangs und für die anderen Erörterungen in diesem Teilabschnitt verweisen wir auf "Normaler Ablauf" in Abschnitt 3.4.2.2 und die Erläuterung der Prozedur "Weiterschalten" dort bzw. auf die abstrakte Beschreibung in Abschnitt 3.4.2.3.

ermittelt werden. Das folgende Beispiel soll dies verdeutlichen. Wir betrachten folgende Syntax:

s	→	np	vp	
np	→	det	adj_p	n
adj_p	→	adj	conj	adj

Gleichzeitig nehmen wir an, der Input des Parsers sei mit dieser (Teil-)Syntax vereinbar, z.B. "Der kleine und tapfere Mann schläft" (hier in segmentierter Notation und unter Voraussetzung entsprechender Lexikoneinträge). Die diese Syntax bearbeitende Parsing-Alternative sei bis zur Segmentierung von "tapfere" fortgeschritten. Die bearbeitete Regel ist dann die adj_p-Regel, die Position in der rechten Seite ist 3. Wenn die nächste Konstituente aufgesucht wird, muß dazu zur dominierenden np-Regel übergegangen werden, deren Zähler erst den Wert 2 hat. In der s-Regel, die nach Abschluß der np durch Segmentierung von "Mann" aufgesucht werden muß, hat der Zähler für die rechte Regelseite erst den Wert 1. Je nach der "syntaktischen Tiefe" der gerade bearbeiteten Regel muß eine beliebige Anzahl von Zählern für eine einzige Parsing-Alternative geführt werden. Wir lösen dieses Problem, indem wir die Zähler nicht in der active rule, sondern im Parsing-Baum führen: Der Zähler für die adj_p-Regel ist der Index des adj_p-Knotens, entsprechendes gilt für den np- und s-Knoten. Nach Beendigung der adj_p-Regel markiert die active rule den np-Knoten als aktiv und findet dort auch den aktuellen Stand des np-Regel-Zählers wieder. Dadurch ergibt sich auch die Möglichkeit, nach Abschluß des Programms (wenn keine Parsing-Alternativen mehr bearbeitet werden und somit auch keine active rules mehr bestehen) durch den Stand des Zählers in den Parsing-Baum-Knoten nachträglich festzustellen, ob diese Regel erfolgreich geparst wurde (d.h. ob das Parsing in der rechten Regelseite bis ans Ende gekommen ist). Der Index eines "erfolgreichen Knotens" muß gleich der Länge seiner Regel plus eins[11] sein. Insgesamt ergibt sich, daß wir alle zur Repräsentation des Parsing-Vorgangs notwendigen Informationen im Parsing-Baum ablegen und nicht in weiteren Strukturen wie den active rules.

Diese Verlagerung eines Zählers für Parsing-Alternativen in den alle Alternativen repräsentierenden Parsing-Baum kann theoretisch dazu führen, daß verschiedene Parsing-Alternativen im Index desselben Knotens verschieden zählen wollen (je nach ihrem Stand in der Syntax). Um dies zu verhindern, muß eine *Regel für einen konsistenten Parsing-Baum mit aktiven Knoten* eingeführt werden: Ein von einer active ru-

[11] Wir müssen eins addieren, um diese Knoten von solchen zu unterscheiden, die bis zur letzten Konstituente der rechten Regelseite gekommen sind, diese aber nicht erfolgreich abarbeiten konnten. In der Implementierung entfällt diese Addition, da Indizes in C mit Null zu zählen beginnen.

le markierter, also aktiver Knoten dominiert keine weiteren aktiven Knoten.

Die nun folgenden Bemerkungen zur Realisierung dieser Regel gehen sehr ins Detail können überlesen werden.

Nach Abschluß der adj_p-Regel (in unserem Beispiel) muß offensichtlich der dominierende np-Knoten als aktiv markiert werden. Nun muß zur Erhaltung der gerade aufgestellten Regel überprüft werden, ob weitere adj_p-Knoten (als "alternative Knoten" [s. "Der Parsing-Baum"] mit dem Knoten in unserem Beispiel verkettet) unter diesem np-Knoten aktiv sind. Falls dies der Fall ist, wird der dominierende np-Knoten kopiert und der betrachtete adj_p-Knoten unter die Kopie gebracht. Nun ist sichergestellt, daß der gesamte Teilbaum unter dem kopierten np-Knoten nur von dieser Alternative bearbeitet wird. Dabei ergibt sich das Problem, unterhalb eines bestimmten Knotens (dem np-Knoten in unserem Beispiel) nach aktiven Knoten zu suchen. Die Eigenschaft, markiert zu sein, ist im Parsing-Baum nicht kodiert. Eine Folgerung aus der oben aufgestellten Regel ist jedoch, daß aktive Knoten im Parsing-Baum nur letzte dominierte Knoten eines anderen Knotens (oder Alternativen davon) sein können. Wenn sich nicht gerade eine Alternative innerhalb des "Weiterschalt"-Prozesses (s. "Normaler Ablauf" in Abschnitt 3.4.2.2) befindet, ist ein aktiver Knoten auch immer eine "rechte untere Zweigspitze" im Parsing-Baum, d.h. ein aktiver Knoten dominiert entweder keine weiteren Knoten oder der letzte dominierte Knoten ist ein Wortknoten.

Diese beiden Feststellungen machen die Suche nach aktiven Knoten wesentlich leichter. Sie führen auch insgesamt zu einer genaueren Beschreibung der Verwaltungsfunktion des Parsing-Baums. Der aktuelle Parsing-Zustand, das sind die aktiven Knoten und deren Einbettungen in dominierende Regeln, findet sich in den Knoten, die von der Wurzel des Parsing-Baums aus durch Verzweigung zu den jeweils letzten dominierten Knoten sowie durch Verzweigung in "Alternativen-Richtung" aufgesucht werden können. Alle weiter "links" liegenden Knoten stellen die Parsing-Vergangenheit dar.

3.4.2.2 Beispiel- und implementationsgebundene Funktionsbeschreibung

Für die nachfolgenden Erörterungen, die das Verhältnis unseres Parsers zu herkömmlichen Parsingstrategien betreffen, wollen wir mit der folgenden Beispielgrammatik (G) arbeiten:

(G) [1] s → np vp

[2] np → det adj_p n
[3] np → det adj n

[4]	adj_p	→	adj	conj	adj
[5]	vp	→	v		
[6]	vp	→	v	adv	
[7]	vp	→	v	adv_p	
[8]	adv_p	→	adv	conj	adv
[9]	adj	→	kleine		
[10]	adj	→	tapfere		
[11]	adv	→	ruhig		
[12]	adv	→	tief		
[13]	conj	→	und		
[14]	det	→	der		
[15]	n	→	mann		
[16]	v	→	schläft		

In Abschnitt 3.4.2.1 unter "Der Parsing-Baum" wurde gesagt, daß wir unsere Implementierung des Parsingproblems im wesentlichen als schrittweise Anpassung des Parsing-Baums an den Input ansehen. Die Schritte in dieser Anpassung, die wir oft die "normale Parsingprozedur" nennen, sollen hier zunächst beschrieben werden. Zu Anfang des Parsings, wo vor Aufbau des Parsing-Baums das erste Wort aus dem Input segmentiert wird, und bei Behandlung von Reparaturphänomenen greift diese "normale Prozedur" nicht; die dann angewendeten Verfahren zielen jedoch darauf ab, nach kleinen Veränderungen möglichst bald die "normale" Prozedur wieder weiterarbeiten zu lassen.

Der Input sei zunächst "derkleineundtapferemannschlaeft".

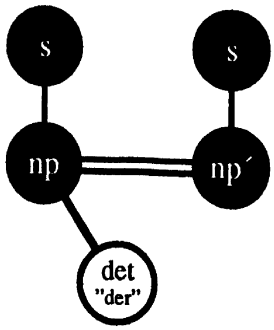
Aufbau des Parsing-Baums zu Beginn des Parsings

Der erste Zeitpunkt, zu dem die normale Parsing-Prozedur nicht arbeitet, ergibt sich aus dem Konzept der normalen Prozedur. Da der normale Programmablauf als Anpassung eines konsistenten Parsing-Baums an den Input verstanden wird, muß in einem anderen Programmteil ein solcher konsistenter Baum erst einmal aufgebaut werden. Dieser Teil entspricht dem z.B. in Abschnitt 3.4.1.3 erwähnten Segmentieren des ersten Wortes ohne kategoriale Hypothese. Es wird also jedes Wort aus dem Lexikon gefunden, dessen Wortform mit dem Anfang des Inputs übereinstimmt. Jedes gefundene Wort wird in einem Parsing-Baum-Knoten repräsentiert. Diese Wortknoten

werden nun mit ihrem "syntaktischen Überbau" versehen, d.h. zu den Kategorien der Knoten werden die Regeln gesucht, die diese Kategorie als "left corner"[12] haben, für jede gefundene Regel wird ein Knoten erzeugt, der den vorigen Knoten direkt dominiert. Diese Prozedur wird auch auf die neuen Knoten angewandt, es wird also schrittweise "überbaut", bis der Vorgang an der Satzebene angelangt ist. Falls der Vorgang nicht zu einer Satzregel kommt, ist das "Überbauen" von einer Kategorie ausgegangen, die nach der verwendeten Grammatik nicht am Anfang eines Satzes stehen kann; die entsprechenden Knoten werden eliminiert. So werden nur Segmentierungen eines ersten Wortes weiter verfolgt, die auch zu einer Satzanalyse führen können.

Demnach untersucht das Programm, welche der Syntaxregeln aus G in der Lage sind, das Wort "der" zu parsen. Dies ist nach [2] und [3] möglich. [2] und [3] sind ihrerseits beide in [1] eingebettet, weil [1] die Kategorie np expandiert.

Folglich erhält man für "der" zwei alternative Einbettungen in Syntaxregeln, die in der folgenden Baumstruktur repräsentiert werden ('np' steht für die np-Regel [2], 'np´' für die Regel [3]):



Es wird zunächst also eine "bottom-up"-Strategie eingeschlagen.

Die Alternative, d.h. die nicht-deterministische Komponente, kommt in G aufgrund der np-Regeln [2] und [3] zustande. Das Aufsuchen der anwendbaren PSG-Regeln erfolgt nach dem "left corner"-Prinzip. "det" ist "left corner" in beiden np-Regeln, und np ist "left corner" in der s-Regel. Über die "left corner" erhält man die unter den anwendbaren Regeln bestehende Dominanz- oder Einbettungshierarchie. Alle bekannten Parser-Formate enthalten analoge Mechanismen, cf. z.B. Kay (1980), Tomita (1987).

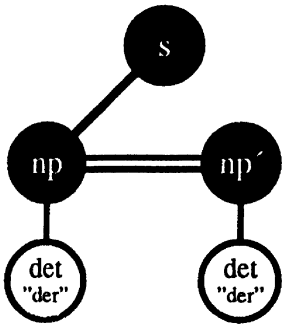
Bei diesem "Überbauen" entsteht ein Baum, in dem mehrere Knoten einen einzelnen Knoten dominieren, also ein nach oben verzweigender Baum. Dieser reflektiert die Strategie des Parsing-Beginns, die ja deutlich bottom-up (nach oben gerichtet) ist.

---

[12] Als "left corner" wird in der Parsing-Literatur die linke (also erste) Konstituente der rechten Seite einer Syntaxregel bezeichnet.

Dieser Baum muß für die weiteren Abläufe in einen konsistenten (top-down-)Parsing-Baum umgewandelt werden. Die Baum-Umwandlung besorgen zwei Prozeduren; die erste teilt den Baum in seine "Pfade" auf, die zweite vereinigt diese Pfade zu einem Baum, indem jeweils auf einer Ebene liegende Knoten verglichen werden: Falls sie mit der gleichen Regel beschriftet sind, werden sie zu einem Knoten vereinigt, die Nachfolger beider Knoten werden unter dem einen Knoten zusammengefaßt (über die Alternativen-Verbindung). Schließlich wird für jeden Regelknoten, der am unteren Ende des Baums steht, eine active rule (Parsing-Alternative) etabliert, die ihre Arbeit mit dem Prozeß "Weiterschalten" (s. unter "Normaler Ablauf") beginnt.

Der Parsing-Baum hat dann folgende Form, wobei doppelte Kanten die Verbindung alternativer Knoten bezeichnen (vgl. "Der Parsing-Baum" unter Abschnitt 3.4.2.1; zum resultierenden Parsing-Baum vgl. auch die Baumkonstruktionskonvention im Abschnitt 3.4.2.3). Es werden zwei Parsing-Alternativen etabliert; die von ihnen markierten und somit aktiven Knoten (vgl. "active rules" unter Abschnitt 3.4.2.1) sind 'np' respektive 'np´'. Wir bezeichnen diese Alternativen im folgenden mit A1 (markiert z.Z. 'np') bzw. A2 (z.Z. 'np´') und werden ihre weitere Entwicklung nun darstellen.



Normaler Ablauf

Wir können nun nach dem oben Explizierten von einem "normalen" Zustand ausgehen, in dem ein konsistenter Parsing-Baum bereits existiert und eine Parsing-Alternative ein Wort mit kategorialer Hypothese aus dem Input zu segmentieren versucht. Die kategoriale Hypothese wird dabei berechnet, indem auf den durch die entsprechende active rule markierten Knoten zugegriffen wird. Sie ist die i-te Kategorie in der rechten Seite der von diesem Knoten angegebenen Regel, i ist der Index dieses Knotens. Nun wird die Segmentierungsprozedur, deren Ablauf unter "Der Lexikonbaum" in Abschnitt 3.4.2.1 angedeutet ist, aufgerufen. Die aktuelle Parsing-Alternative "überlebt" diese Prozedur, falls mindestens eine passende Segmentierung eines Wortes aus dem Input gefunden wird; falls es mehrere solcher Segmentierungen gibt,

"vermehrt" sich die Alternative (es entsteht je eine für jede gefundene Segmentierung), und wir können eine beliebige der so entstandenen Alternativen für die weiteren Erläuterungen betrachten. Diese Alternative erzeugt einen Wortknoten mit dem soeben segmentierten Wort und etabliert diesen als (neuen) letzten dominierten Knoten des von der active rule markierten Knotens. Es ist dabei sichergestellt, daß eine andere durch "Vermehrung" erzeugte Alternative nicht denselben Knoten markiert: Die Segmentierungsprozedur kopiert den markierten Knoten im Rahmen des "Vermehrens" und verbindet die Kopien mit diesem durch "Alternativen-Verkettung" (s. Abschnitt 3.4.2.1: "Der Parsing-Baum" bzgl. "Alternativen-Verkettung", "active rules" bzgl. eines vergleichbaren Kopiervorgangs zur Erhaltung eines konsistenten Parsing-Baums).

Der nun folgende Schritt im Ablauf einer Parsing-Alternative kann mit dem Begriff "Weiterschalten" umschrieben werden. Aufgabe dieses Schrittes ist es, die nächste kategoriale Hypothese für diese Alternative zu finden, was Veränderungen im Parsing-Baum zur Folge hat. Der Vorgang kann aufgeteilt werden in die Teilschritte "Index erhöhen und (evtl.) aufwärtswandern" und "(evtl.) nach unten erweitern", welche nachfolgend beschrieben werden.

Der einfachste Fall des "Weiterschaltens" ist die Erhöhung des Index im aktiven Knoten (dem Knoten, den die active rule markiert) um 1. Dadurch wird in dessen rechter Regelseite die nächste Konstituente anvisiert. Nun muß jedoch überprüft werden, ob diese Konstituente existiert, d.h. ob nicht die soeben geparste Konstituente die letzte in ihrer Regel war. In diesem Fall wäre das Ende der rechten Regelseite erreicht, das Parsing mit der aktuellen Regel also insgesamt abgeschlossen. Dies ist jedoch nach der Segmentierung von "der" in beiden Alternativen nicht der Fall. Das optionale "Aufwärtswandern" im ersten Teilschritt wird deshalb weiter unten besprochen.

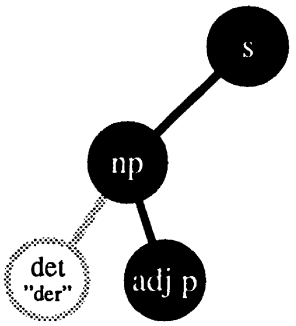
Wir haben nun eine nächste Konstituente in der rechten Seite dieser Regel gefunden, diese Konstituente wird zur kategorialen Hypothese. Diese Hypothese kann jedoch nur dann direkt für den nächsten Segmentierungsversuch verwendet werden, wenn sie präterminal ist. Im Fall einer komplexen "kategorialen Hypothese" müssen die Syntaxregeln gefunden werden, die diese Kategorie expandieren. Für eine solche Regel wird dann ein Knoten unter dem markierten Knoten (als dessen letzter dominierter) erzeugt, und die Markierung der active rule zeigt auf diesen Knoten. Kategoriale Hypothese ist die "left corner" der expandierenden Regel, der Index des Knotens ist entsprechend 1. Falls mehrere Expansionen in der Syntax vorhanden sind, wird für jede expandierende Regel ein Knoten erzeugt, diese Knoten stehen als "Alternativengruppe" an gleicher Stelle wie der vorhin erwähnte einzelne Knoten, und durch "Vermehrung" der Parsing-Alternative zeigt je eine active rule auf diese Knoten. Nun muß (für jede Expansion) die "left corner" als neue kategoriale Hypothese wieder auf Komplexität überprüft und dann möglicherweise nochmals, wie gerade beschrieben, "nach unten erweitert" werden. Auch das "Nach-unten-Erweitern" setzt sich also in

beliebig vielen Stufen fort. Der Vorgang bricht jedoch ab, wenn für die betrachtete Parsing-Alternative (und die durch "Vermehrung" erzeugten) eine präterminale Hypothese gefunden ist. Der nächste Segmentierungsversuch kann beginnen.

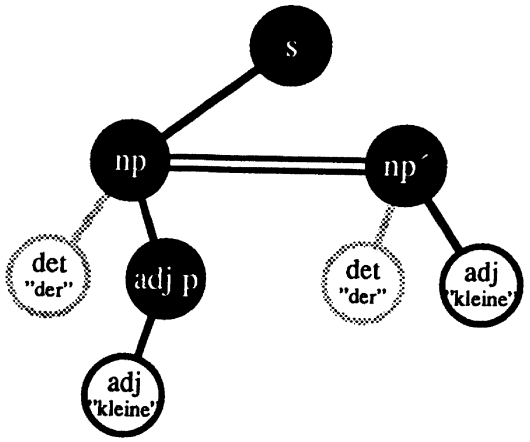
Der Parser geht also die aktuellen Regeln der Alternativen durch und liest die nächste Kategorie ein, die gleichzeitig eine Hypothese über die Kategorie der nachfolgenden Konstituente darstellt. In A2 ist diese Kategorie präterminal (vgl. Regel [3] in (G)), deshalb kann mit ihr ein neues Wort segmentiert werden.

In A1 haben wir dagegen eine nicht-terminale Kategorie gegeben (vgl. (G), [2]). Sie wird durch den Parser so lange expandiert, bis eine präterminale "left corner" vorhanden ist. Hier wird also eine "top-down"-Strategie verfolgt. Insgesamt gesehen, haben wir hier somit eine Kombination aus beiden Strategien gegeben.

A2 liefert eine Segmentierung von "kleine". A1 sieht nach Erweiterung wie folgt aus:[13]



Die neue kategoriale Hypothese für A1 ist 'adj'. Der nächste Segmentierungsversuch ist erfolgreich, denn "kleine" im nachfolgenden Input ist mit 'adj' kompatibel. Der Parsing-Baum wird nun um das Terminal "kleine" erweitert. Analoges geschieht in der Alternative 'np´', deren 'adj'-Kategorie nicht expandiert werden mußte. Somit ergibt sich:



[13] Der hier beschriebene Vorgang entspricht der Anwendung der Regeln (S2) und (S3) aus der abstrakten Funktionsbeschreibung (Abschnitt 3.4.2.3).

A1 liefert also ebenfalls eine Segmentierung von "kleine". A2 bricht nach "kleine" ab, da die Kategorien-Hypothese 'n' und der Inputrest "undtapfere..." aufgrund des Lexikons nicht kompatibel sind. Der erreichte Parsing-Zustand wird jedoch im Parsing-Baum festgehalten. Dabei fällt dem Parsing-Baum eine Rolle zu, wie sie in "chart-Parsern" von der chart ausgeübt wird. Aus empirischen Gründen ist geboten, daß auch ein nur partiell abgeschlossener Parsingzustand erhalten bleibt und nicht vollständig aus der Agenda entfernt wird.[14] Es kann nämlich der Fall eintreten, daß genau diese Regel durch eine Reparatur aktiviert wird. Dies geschähe dann, wenn wir z.B. einen Input der Form

"derkleineundaehkleinemannschlaeft"

zu verarbeiten hätten. Hier greift die Reparatur auf eine Alternative zurück, die bereits verworfen worden war. Verworfene Alternativen sind nicht für immer verloren. Der Parser kennt neben seiner "Parsing-Tabelle", i.e. allen aktiven Knoten, auch den erreichten Parsing-Zustand, i.e. die während der "Parsing-Geschichte" etablierten Expansionen von Kategorien.

Zunächst wird jedoch nur die übrig gebliebene Parsing-Alternative A1 weiter verfolgt. Im Laufe der 'adj_p'-Regel reduziert sich die Prozedur "Weiterschalten" auf den Teilschritt "Index erhöhen", d.h. ein "nach unten erweitern" ist für die beiden verbleibenden Konstituenten 'conj' und 'adj' nicht erforderlich. Mit diesen kategorialen Hypothesen können aus dem noch nicht analysierten Input die Konstituenten "und" sowie "tapfere" segmentiert werden. Es wurde jedoch bereits zu Beginn der Vorstellung der Prozedur "Index erhöhen" erwähnt, daß überprüft werden muß, ob in der aktuellen Regel das Ende der rechten Seite erreicht worden ist. In diesem Fall sagen wir, der Knoten, welcher den markierten Knoten dominiert, habe gerade eine – allerdings komplexe – Konstituente erfolgreich geparst. Wir stellen die Situation anhand dreier Regeln aus (G) dar:[15]

s	→	np	vp	
np	→	det	adj_p	n
adj_p	→	adj	conj	adj

Nach Segmentierung des zweiten 'adj' innerhalb der 'adj_p'-Regel gibt der Index des 'adj_p'-Knotens keine Konstituente der rechten Regelseite mehr an. Der Zähler des 'np'-Knotens referiert jedoch in der rechten Seite der np-Regel auf die 'adj_p', die jetzt insgesamt segmentiert ist. Die soeben besprochene Indexerhöhung mit anschlie-

---

[14] Zum Thema "charts" und "Agenda" cf. Kay (1980) und Johnson-Laird (1983), S. 296–355.

[15] Anhand des gleichen Auszugs aus (G) wurde der Vorgang des "Aufwärtswanderns" bereits unter "active rules" in Abschnitt 3.4.2.1 angesprochen.

ßendem Test auf Regelende ist also auf den dominierenden Knoten anzuwenden, der Prozeß setzt sich möglicherweise in mehreren Stufen aufwärts fort. Welcher Knoten auf diese Weise bearbeitet wird, gibt wieder die active rule an, deren Markierungspointer bei Feststellung eines Regelendes "aufwärts wandert". Es wird dabei sichergestellt, daß sich diese active rule die dominierenden Knoten nicht mit einer anderen active rule "teilt", wozu nötigenfalls eine Kopie dieses Knotens erzeugt werden muß (genauere Erläuterungen zu diesem Prozeß finden sich unter "active rules" inm Abschnitt 3.4.2.1).

Wenn nun die sich nach oben fortsetzende Prozedur "Index erhöhen und (evtl.) aufwärtswandern" bis zu der obersten Baumebene (der Satzebene) kommt und auch dort das Regelende erreicht ist, kann keine weitere kategoriale Hypothese der betrachteten Alternative gefunden werden: Die Syntax ist "durchgearbeitet". Falls die Parsing-Alternative zu diesem Zeitpunkt auch den gesamten Input abgearbeitet hat, ist der Input durch diese Alternative erfolgreich analysiert, das Analyseergebnis findet sich im Parsing-Baum (der jedoch erst nach Beendigung sämtlicher Alternativen ausgegeben wird). Wenn der Input noch nicht beendet ist, wird die Parsing-Alternative auf den Beginn ihrer "Aufwärtswanderung" zurückversetzt und eine Reparaturanalyse versucht. So können Äußerungen wie

"Der kleine Mann schläft ruhig und tief – äh schläft"

korrekt analysiert werden.

Im vorliegenden Beispiel sind wir jedoch noch nicht auf der Satzebene angelangt. Nach Segmentieren von "kleineundtapfere" ist erst die 'adj_p' abgeschlossen. Das Resultat wird im Parsing-Baum abgebildet. Der Parser stellt als nächstes fest, daß er das Ende einer CFPSG-Regel erreicht hat und geht zur "dominierenden" Regel von A1 über. Die aktuelle kategoriale Hypothese ist nun, daß das nächste Wort vom Typ 'n' sein wird. Danach ist in A1 die np-Regel abgeschlossen, und der Parser geht zur vp-Regel über. Auch für die VP werden aufgrund der Regeln [5], [6] und [7] Alternativen etabliert.

Reparaturanalyse

Bislang haben wir gesehen, daß der Parser zwar mit herkömmlichen Strategien arbeitet, sich jedoch in manchen Details von "klassischen" Parsern unterscheidet: Am auffälligsten ist, daß partiell erfolgreiche Alternativen "konserviert" werden und daß eine Datenstruktur, der Parsing-Baum, sowohl die Funktion einer "Parsingtabelle" als auch die einer Agenda erfüllt.

Die charakteristischsten Unterschiede zu herkömmlichen Parsern liegen darin, daß neue Klassen nicht-deterministischer Strukturen erfaßt werden können. Um dies zu sehen, betrachten wir verschiedene mögliche Reparaturansätze zur NP "derkleine-

mann":

(R1) "derkleineaehundtapferemann",
(R2) "derkleineaehgrossemann",
(R3) "derkleineaehderkleinemann",
(R4) "derkleineundtapfereaehkleinemann" [16]
(R5) "derkleineaehdiekleinefrau"

(R1) bis (R4) sind nach dem Leveltschen Schema einschlägige "repairs", bestehend aus "original utterance", "editing phase" und "repairing utterance", die besprochenen Idealisierungen zugestanden (s. das Strukturschema aus 3.4.1.1). Würde der Parser ausschließlich sofunktionieren, wie oben beschrieben, dann bekäme man nur partielle Analysen als Resultate.

Was geschieht nun außerhalb der bisher beschriebenen "normalen" Parsing-Prozedur im Programm? Der wichtigste Fall einer nicht-normalen Arbeitsweise ist die Behandlung einer Reparatur. Die Erkennung von Reparaturen und Fragmenten bildet die erwähnte "metagrammatische Komponente" des Parsers nach Schegloff (Abschnitt 3.4.1.1). Erkannt werden Reparaturen dadurch, daß ein spezielles Reparatursignal – z.Z. die feste Zeichenkette "aeh" – im Input vorkommt und die Segmentierung eines Wortes fehlschlägt. Die betreffende Parsing-Alternative versucht bei fehlgeschlagener Segmentierung, das Reparatursignal einzulesen; falls dies gelingt, wird eine Struktur vom Typ eines Worteintrags im Lexikon (s. unter "Wörter" und "Die Syntax" in Abschnitt 3.4.2.1) erzeugt, die als Hauptkategorie eine spezielle Zahl (Hesitationskategorie) und als Wortform das Reparatursignal erhält. Ein "Wortknoten" (vgl. "Der Parsing-Baum" in Abschnitt 3.4.2.1), der auf diese Hesitationsstruktur verweist, wird im Parsing-Baum an der Stelle eingetragen, an der bei erfolgreicher Segmentierung ein "echter" Wortknoten entstanden wäre. Der Index des aktiven Knotens (der den Hesitationsknoten direkt dominiert) wird jedoch nicht erhöht, da keine Konstituente aus der rechten Regelseite geparst wurde. Hier wird der Unterschied zwischen Grammatisierung von "repair"-Strukturen und der hier verfolgten metagrammatischen Lösung deutlich: Eine Grammatisierung müßte die Hesitationskategorie als Teil der rechten Regelseite auffassen (vgl. Abschnitt 3.4.1.1). Nun beginnt die eigentliche Reparaturprozedur, die nichts anderes tut, als neue Parsing-Alternativen aus der gerade arbeitenden und ähnlichen, bereits verworfenen Alternativen zu erzeugen. Wir beschreiben in Abschnitt 3.4.2.3 unter (S4) die genauen Regeln, nach denen diese neuen Alternativen erzeugt werden, in abstrakter Form. Die

---

[16] Für R4 findet man nur selten Beispiele im empirischen Material. Es wurde hier als Testfall für die Entwicklung eines möglichst allgemeinen Parserformats aufgenommen. Eine Reparatur dieser Art verlangt nämlich, daß der Parser für die Segmentierung von "kleine" auf die Regel [3] der Grammatik (G) zurückgreift, die er bereits im Zuge der Segmentierung von "kleine und tapfere" als *erfolglos* markiert hatte.

angewandte Strategie läßt sich in 4 Teile gliedern:

1. Keine Veränderung. Die aktuelle Alternative versucht, unter Überlesen des Reparatursignals weiterzuarbeiten. Diese Teilstrategie erfaßt Inputs wie

"deraehkleinemannschlaeft"

2. Zurückgehen in dieser und den dominierenden Regeln. Es werden für die aktuelle und alle dominierenden Regeln $(i-1)$ Alternativen erzeugt, wobei i der Stand in der jeweiligen rechten Regelseite, also des Index der jeweiligen Regelknotens ist. Die neuen Alternativen bearbeiten die gleiche Regel mit dem Inputrest ab "aeh", jedoch mit den verringerten Indizes $(i-1)$, $(i-2)$, ..., 1. Möglicherweise entstehen so komplexe syntaktische Hypothesen, die mit der Prozedur "nach unten erweitern" (s. unter "Normaler Ablauf" in diesem Abschnitt) behandelt werden.
Hiermit können Inputs analysiert werden wie

"derkleinemannschlaeftruhigundaehtiefundfest" oder
"derkleinemannschlaeftruhigundaehschlaeft"

3. Aktivieren bereits verworfener Alternativen. Es werden alle Knoten aufgesucht, die mit dem aktiven Knoten oder dessen dominierenden direkt als alternative Knoten verbunden sind. Sofern diese nicht mehr aktiv sind – es wird hier die Prozedur verwendet, die unter "active rules" in Abschnitt 3.4.2.1 im Rahmen der Regel über Parsing-Bäume mit aktiven Knoten beschrieben wurde –, werden die Knoten mit der Strategie 2 behandelt, d.h. es wird in den gefundenen Regeln zurückgegangen und für jeden "Rückkehrpunkt" eine Parsing-Alternative erzeugt, bearbeitet wird jeweils der Input nach dem Reparatursignal.
Input hierfür ist z.B.

"derkleineundaehkleinemannschlaeft" (beachte, daß die Segmentierung "derkleinemann" schon vor "aeh" verworfen werden muß)

4. Erzeugen struktureller Alternativen. Es werden alle Syntaxregeln aufgesucht, die der Regel, welche die aktuell bearbeitete Regel unmittelbar dominiert, "ähnlich" sehen. Ähnlich in diesem Sinne sei eine Regel, wenn sie in ihrer rechten Regelseite eine Kategorie enthält, die mit der Hauptkategorie der aktuellen Regel übereinstimmt. Solche Regeln dominieren also eine Regel, die der aktuellen Regel ähnlich (im Sinne von gleicher Hauptkategorie) ist. Unter der Kategorie einer Regel verstehen wir dabei die von dieser Regel expandierte Kategorie, also die linke Regelseite. Die gefundenen Regeln werden nun zu einem neuen Teil-Parsing-Baum verarbeitet und zum bisherigen Parsing-Baum hinzugefügt (Aufbau eines Parsing-Baums: s. entsprechender Teil von Abschnitt 3.4.2.1)
Ein Beispiel hierfür ist

"derkleinemannaehdiekleinefrauschlaeft" .

Für "prepositioned-repairs" wie (R1) reicht es aus, wenn das Hesitationssignal überlesen und die nächste Kategorie verwendet wird. Wir wenden demnach Teilstrategie 1. an.

Um (R2) zu bekommen, wird eine Parsing-Alternative etabliert, die in der aktuellen Regel eine Kategorie zurückgeht. Es muß ja erneut ein Objekt der Kategorie 'adj' segmentiert werden. In der aktuellen Regel hat der Parser diese Konstituente sozusagen bereits "aufgebraucht". Für (R3) ist sogar ein Zurückgehen an den Regelanfang nötig. In beiden Fällen wirkt also Teilstrategie 2.

(R4) verlangt die Aktivierung einer "aufgegebenen" Alternative (Teilstrategie 3).

Dagegen muß für (R5) ein struktureller Neuansatz bereitgestellt werden. Dieser zieht alle Schritte nach sich, die auch beim Einlesen des ersten Wortes durchzuführen sind, z.B. die Suche nach adäquaten "left corners", dominierenden Regeln, Aufbau des Parsing-Baums usw. (Teilstrategie 4.).

Jede dieser Neuberechnungen einer Parsingalternative hält sich an das *Prinzip der strukturellen Analogie zwischen der ursprünglichen und der reparierenden Äußerung*, macht diese jedoch auf der abstrakten Ebene der Kategorien fest: Wenigstens die Hauptkategorien müssen übereinstimmen.[17] Auch neue Satzanfänge werden von diesem Prinzip der strukturellen Analogie nicht ausgenommen. Eine Äußerung

"derkleineaehvermutlichkommtmax"

würde vom Parser nicht als Reparatur akzeptiert, möglich wäre dagegen

"derkleineaehernakommt" .

Daraus ist ersichtlich, daß durch die zulässigen Parsing-Strategien auch ein Wohlgeformtheitsprinzip festgelegt wird. Das empirische Approximationspotential dieses Wohlgeformtheitsprinzips steht natürlich bei jeder Simulation zur Diskussion. Der Vorteil der Implementierung besteht an diesem Punkt darin, daß man sich auf eine algorithmische Version des Wohlgeformtheitsprinzips festgelegt hat, deren Nachteile offen zutage treten und infolgedessen verhandelbar sind.

Als Beispiel betrachten wir nun den Input

(R2') "derkleineaehgrossemanmann".

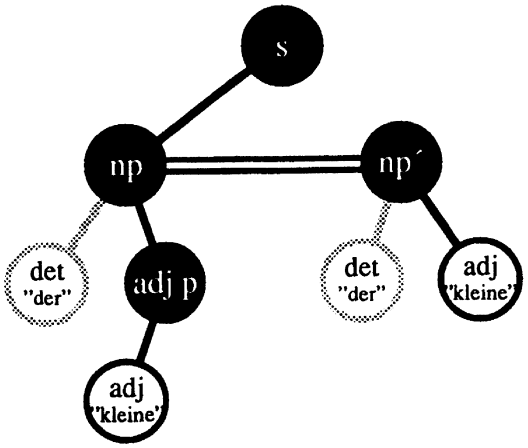
(R2') enthält eine Reparatur "kleine aeh große" und ein Fragment "man"[18]

---

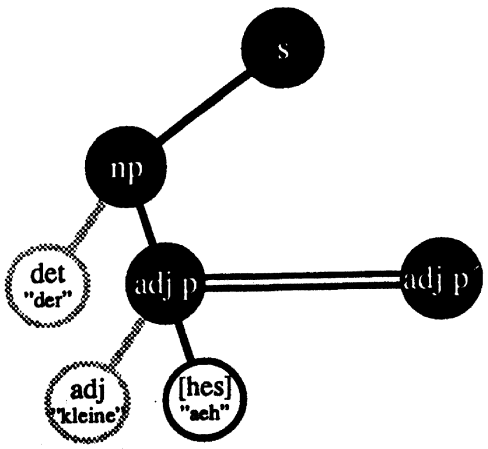
[17] Dies ist eine Modellierung, die dem Leveltschen Konzept des "structural link" entspricht (vgl. Schema und Erläuterung in 3.4.1.1).

[18] in eventueller Eintrag für das Indefinitpronomen "man" im Lexikon stört hier nicht, da mit der angegebenen Syntax keine entsprechende kategoriale Hypothese erzeugt wird.

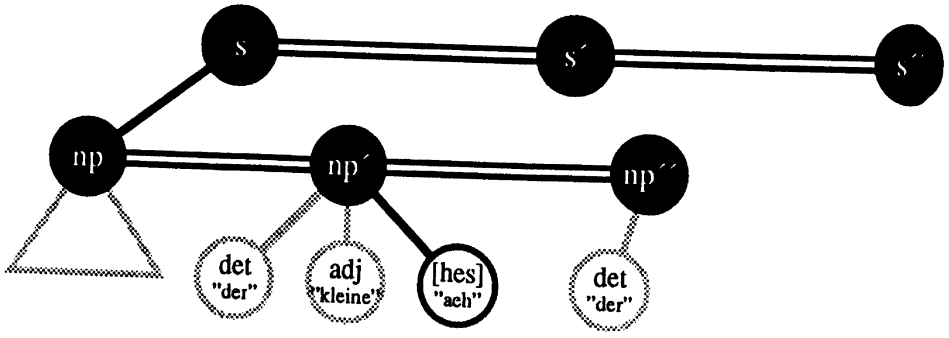
Erinnern wir uns an die Beispielgrammatik (G) und die unter "Normaler Ablauf" (vgl. dort, in diesem Abschnitt) gemachten Bemerkungen. Demnach wird der Input (R2') bis "derkleine" mit 2 Alternativen geparst. Wir bezeichnen diese Alternativen wieder mit A1 und A2. A1 parst mit der 'adj_p'-Regel [4], eingebettet in die 'np'-Regel [2] und die 's'-Regel [1]. A2 parst mit der 'np'-Regel [3], ebenfalls eingebettet in die 's'-Regel [1]. Der Parsing-Baum gibt den Zustand der Alternativen wieder:



Entsprechend "Index erhöhen" im normalen Parsingablauf (vgl. "Normaler Ablauf" oben) ist 'conj' unter 'adj_p' neue kategoriale Hypothese. Sie schlägt fehl, da es in (G) kein Wort "aeh" gibt. Nun werden die oben angeführten Reparaturstrategien 1. bis 4. aktiviert, wodurch eine größere Anzahl Parsing-Alternativen entsteht. (3. "Aktivieren bereits verworfener Alternativen" erzeugt jedoch keine Parsing-Alternativen, da bis zu "derkleine" keine Alternativen zu verwerfen waren). Erfolgreich ist nur die zweite Strategie. In A1 wird durch "Zurückgehen in dieser Regel" für den zur Zeit aktiven 'adj_p'-Knoten eine Alternative erzeugt, welche die im folgenden Bild gezeigten Knoten 'adj_p´' bearbeitet:



Bei A2 schlägt in der aktiven Regel 'np´' die kategoriale Hypothese 'n' ebenfalls fehl. Da auch diese Alternative das Hesitationssignal "aeh" vorfindet, werden erneut die verschiedenen Reparaturstrategien angewandt. Daraus resultiert folgende Veränderung des Parsing-Baums bezüglich 'np´':[19]



Damit ist die Segmentierung der Hesitation abgeschlossen, und es sind alle Neuansätze für die Reparatur vorgesehen, die oben besprochen wurden, z.B.:

1. Weitergehen in der aktuellen Regel nach Überlesen der Hesitation: Ist abgedeckt, da die aktuelle Regel erhalten bleibt.

2a. Reparatur des letzten Terminals, i.e. "kleine" . Dies wird durch das Zurückgehen in der 'adj_p'-Regel bzw. in der 'np'-Regel erreicht und mit Hilfe der Alternativen 'adj_p´' bzw. 'np´´' durchgeführt. Bezüglich des Inputs ist dies der einschlägige Reparaturansatz.

2b. Ein neuer NP- bzw. Satzanfang ist aufgrund von 's´' und 's´´' möglich.

Diese Reparaturstrategien sind Teil der metagrammatischen Komponente des Parsers, welche dem in 3.4.1.1 erwähnten Vorschlag von Schegloff entspricht und die Ergebnisse Levelts voll berücksichtigt. Sie behandeln jene nicht-deterministischen Strukturen, die durch Reparaturen zustandekommen. Das Non-Determinismus-Problem ist in den verschiedenen strukturellen Ansatzmöglichkeiten begründet, wie sie für "repairs" bestehen. Von den Parser-Paradigmen her gesehen sind diese Strategien auch zentral, da eine Neuberechnung der anwendbaren Syntaxregeln vorgenommen wird.

Der Unterschied zur Situation beim Einleseprozeß besteht darin, daß nicht das erste segmentierte Wort den Aufbau der neuen "Parsingtabelle" (hier: der Teilbäume 'adj_p´', 'np´', 's´' und 's´´' im Parsing-Baum) bedingt, sondern die Hesitation "aeh" und die aktuelle Regel, deren kategoriale Hypothese fehlgeschlagen ist.

Nach Erzeugen dieser Alternativen kehrt das Programm in die normale Par-

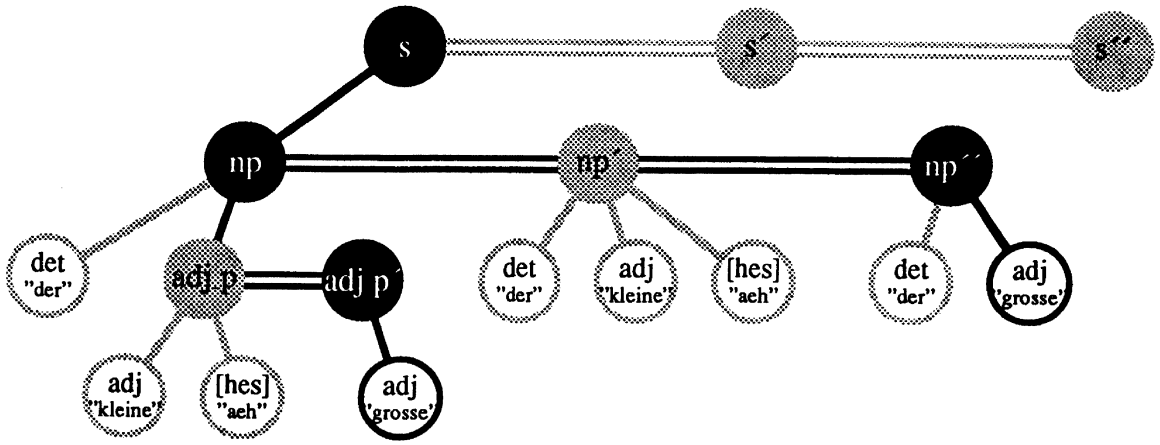
---

[19] Der mit 'np' bezeichnete Knoten ist der im letzten Bild bezüglich Reparaturen veränderte. 's´' stammt noch aus der Behandlung der 'adj_p' und wurde oben nicht aufgeführt.

sing-Prozedur zurück, die neuen Alternativen werden zunächst einem Segmentierungsversuch unterworfen. Es wird hier deutlich, daß die Erzeugung sehr vieler Parsing-Alternativen, von denen die meisten bald wieder verworfen werden, einer genaueren Analyse des Reparatursegments im Input mit anschließender genauerer Vorhersage der nächsten grammatischen Kategorie ("lookahead" u.ä. Mechanismen) vorgezogen wird. Diese Möglichkeit würde zu weniger Parsing-Alternativen, aber einer komplizierteren Prozedur außerhalb des normalen Ablaufs führen.

Im Beispiel setzt sich die Parsing-Prozedur wie folgt fort: Beim nächsten Segmentierungsversuch ist bezüglich 'adj_p'' die kategoriale Hypothese 'adj' erfolgreich, bezüglich 'np'' ebenfalls. Dagegen sind die beiden Satzanfänge 's'' und 's'', die erst noch über "nach unten erweitern" (s. unter "Normaler Ablauf" in diesem Abschnitt) expandiert werden müssen, nicht erfolgreich. Sie liefern nämlich als kategoriale Hypothese 'det' und 'det' ist mit dem Inputsegment "grosse" nicht kompatibel; daß diese Satzanfänge jedoch nötig sein können, zeigten (R4) und (R5) oben.

Als neuen Zustand des Parsing-Baums erhalten wir somit:



's'' und 's'' sowie deren Extensionen werden nicht total eliminiert, sie könnten nochmals als strukturelle Alternativen für eventuell folgende Reparaturen wichtig werden.

Fragmentanalyse

Fragmente wie "man" in

"derkleineaehgrossemanmann"

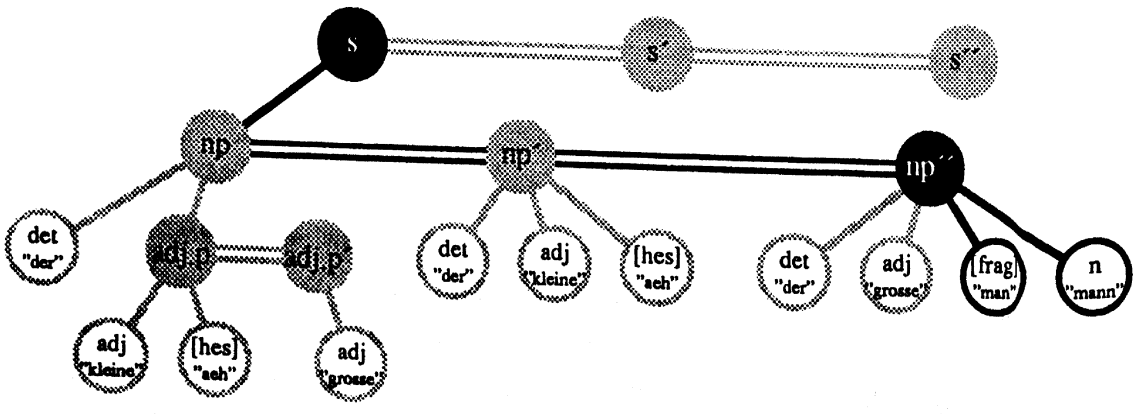
werden so behandelt, daß versucht wird, das Wortfragment zu isolieren und danach weiterzuparsen. Neue Parsingalternativen entstehen dadurch insofern, als der nächste Parsingschritt, von dessen Erfolg die Feststellung des Fragmentes abhängt, mehrere Segmentierungen des Inputs nach dem Fragment liefern kann. Vor allem das Erkennen von Wortfragmenten macht eine Lexikonkonzeption notwendig, die von den normalerweise verwendeten Wortlisten abweicht.

Der Parser versucht ein Wortfragment zu erkennen, falls bei einer mißlungenen Segmentierung kein Reparatursignal im Input entdeckt wird. In abstrakter Form findet sich die angewandte Strategie in Abschnitt 3.4.2.3 als Strategie (S5). Aus programmtechnischer Sicht wird ein Teil-Lexikon aus denjenigen Wörtern angefertigt, die nach der kategorialen Hypothese hätten segmentiert werden können. Diese Wörter sind "Kandidaten" für Fragmente. Es wird versucht, ein Segment im Input zu finden, das Anfang einer Wortform im Teil-Lexikon ist, dieses Segment zu überlesen und nach dem Segment eine erfolgreiche Wortsegmentierung durchzuführen. Gelingt dies, so wird das überlesene Segment in einem Parsing-Baum-Knoten vor dem soeben segmentierten Wort eingetragen. Dazu wird, ähnlich wie für das Reparatursignal, eine Pseudo-Wort-Struktur erzeugt, die als Wortform das Fragment und als Hauptkategorie eine spezielle Fragmentkategorie erhält. Anschließend setzt die Alternative – bzw. die Alternativen, falls die Segmentierung nach dem Fragment zu mehreren Ergebnissen führte oder mehrere (verschieden lange) Fragmente gefunden wurden – ihre Arbeit mit dem Prozeß "Weiterschalten" fort, der zu Beginn dieses Abschnitts beschrieben wurde.

Im Beispiel schlägt die nächste kategoriale Hypothese unter 'adj_p´' und 'conj' bezüglich "manmann" fehl. Eine erneute Anwendung der Reparaturstrategien ist nicht möglich, weil keine Hesitation vorliegt. Infolgedessen versucht das System, die gerade beschriebene Fragmentstrategie anzuwenden. Aufgrund des Lexikons in (G) wird ein Fragment "man" isoliert. Die Segmentierung "man" ist jedoch mit der kategorialen Hypothese 'conj' nicht kompatibel, infolgedessen wird die 'adj_p' bearbeitende Parsing-Alternative eliminiert.

Die Hypothese 'n' unter 'np´´' ist zunächst aus analogen Gründen ebenfalls nicht erfolgreich. Wiederum führt eine Fragmentanalyse weiter. Diesmal ist jedoch, nach graphemweisem Überlesen von "man", der Rest des Inputs mit der kategorialen Hypothese 'n' kompatibel.

Hier nun der aktuelle Zustand des Parsing-Baums:



Der Parser kann nun in der einzig verbliebenen aktuellen Alternative "weiterschalten" und, da er das Ende der 'np'-Regel bezüglich 'np´´' erreicht hat, mit der kategorialen Hypothese 'vp' weiterparsen.

3.4.2.3 Abstrakte Funktionsbeschreibung

Wir beschreiben nun den Parsing-Prozeß anhand einer Reihe von allgemeinen Strategien, deren Implementierung nicht von vornherein bestimmt ist. Dazu definieren wir allgemein eine "Parsing-Alternative" und beschreiben anhand von Regeln ("Strategien"), welche Parsing-Alternativen vom Parsing-Prozeß verfolgt werden.

Zu Beginn des Parsings müssen bestimmte Parsing-Alternativen erzeugt werden, wozu die erste anzugebende Strategie dient. Im Verlauf des Parsings entwickeln sich einzelne Alternativen weiter; sie erzeugen gelegentlich mehrere Parsing-Alternativen, andere werden verworfen. Diese Änderungen werden durch weitere Strategien definiert, die von einer gegebenen Parsing-Alternative ausgehen und meist an Stelle dieser Alternative eine oder mehrere Alternativen etablieren. Mit Hilfe dieser Strategien kann formal eine Abbildung einer Menge von betrachteten Parsing-Alternativen auf eine neue definiert werden. In den Strategien wird eine Parsing-Alternative auf eine Menge neuer Parsing-Alternativen abgebildet, die Vereinigung aller Bildmengen ist die neue Menge der betrachteten Parsing-Alternativen. Gelegentlich wird eine Alternative auch "eliminiert"; das heißt, daß sie auf die leere Menge abzubilden ist.

Einleitende Bemerkungen und Definitionen

Für die folgenden Erörterungen sind zunächst einige Redeweisen einzuführen. Wir betrachten eine kontextfreie Phrasenstrukturgrammatik G. Sämtliche terminalen Symbole aus G sollen in Regeln der Form Kategorie → Symbol zu finden sein. Wir nennen diese Regeln "lexikalische Regeln", die Kategorien auf der linken Seite dieser Regeln "lexikalische Kategorien". Sei L die Teilmenge von G, die alle lexikalischen Regeln enthält (das "Lexikon"). Die Elemente von L bezeichnen wir auch als "Wörter"; die linke Seite einer lexikalischen Regel nennen wir demnach "Wortkategorie", die rechte Seite "Wortform". Wortformen sind Ketten von Graphemen. Der Parsing-Prozeß bearbeitet außerdem eine weitere Graphemkette, den "Input". Wir werden gelegentlich zu einer Graphemkette auch Anfangssegmente oder Teilketten "ab l" oder "von n bis m" betrachten (l, n, m sind natürliche Zahlen), was wir mittels einer Numerierung der Grapheme in der Kette, beginnend mit 1, definieren können. Die Länge einer Graphemkette ist die Anzahl ihrer Graphempositionen.

Die Menge der nicht-lexikalischen Regeln aus G heiße S (die "Syntax"). Ein Element R aus S besteht aus einer (nicht-lexikalischen) Kategorie K auf der linken Seite und einer endlichen Anzahl (beliebiger) Kategorien auf der rechten Seite. Wir

bezeichnen ein solches R als "K-Regel" bzw. K als die "Kategorie von R" und sagen, eine Regel R´ dominiere die Regel R (bzw. dominiere R an Stelle i), wenn sich in der rechten Seite von R´ irgendwo (bzw. als i-te Kategorie) die Kategorie K befindet. Die Länge einer Regel ist die Anzahl der Kategorien ihrer rechten Seite.

Eine Kategorie von spezieller Bedeutung ist 'S', die Satzkategorie. 'S' ist das sog. Anfangssymbol in G, also die Kategorie vollständiger, wohlgeformter Ausdrücke (nach Maßgabe der Grammatik G).

Der Begriff "natürliche Zahl" schließt in diesem Abschnitt die Null *nicht* ein.

Wir können nun die für diese Beschreibung zentralen Konzepte der "Parsing-Alternative" und der "kategorialen Hypothese" definieren.

Definition:

Eine Parsing-Alternative ist ein Paar $(p, \langle (R_1, i_1), (R_2, i_2), \ldots, (R_n, i_n) \rangle)$ aus einer natürlichen Zahl p und einer Folge von Paaren aus je einer Regel R_k und einer natürlichen Zahl i_k, für die gilt:

- p ist nicht größer als die Länge des Inputs,
- R_k ist Element aus der Syntax S, i_k ist nicht größer als die Länge der Regel R_i für alle k von 1 bis n,
- R_1 ist eine 'S'-Regel,
- R_k dominiert R_{k+1} an der Stelle i_k für alle k von 1 bis $(n-1)$.

(Die Zahl p bezeichnet die Position im Input, die Zahlen i_k die Positionen in den rechten Seiten der jeweiligen Regeln, bis zu denen die Analyse fortgeschritten ist.)

Beachte, daß für die Regeln R_k keine weiteren Bedingungen als die oben genannten gelten. Insbesondere bedeutet die Indizierung von 1 bis n nicht, daß die R_k auch in dieser Reihenfolge in der Syntax S stehen müssen – tatsächlich haben wir S als Menge, also ohne feste Reihenfolge definiert –, und S umfaßt in den meisten Fällen auch mehr Regeln, als eine Parsing-Alternative aufnehmen kann.

Jede Parsing-Alternative liefert uns eine Hypothese über die Kategorie des nächsten einzulesenden Wortes, da wir zu jeder Regel eine Zahl – die aktuelle Position in der jeweiligen Regel – mitführen. Diese läßt sich mit der für unsere Parsing-Alternative verwendeten Struktur leicht definieren.

Definition:

Als "kategoriale Hypothese" einer Parsing-Alternative der Form $(p, \langle (R_1, i_1), \ldots, (R_n, i_n) \rangle)$ bezeichnen wir die Kategorie, die in der rechten Seite von R_n an der Stelle i_n steht.

Wir sagen, eine Regel – sei sie syntaktisch oder lexikalisch, im letzteren Fall also ein Wort – "entspricht" einer kategorialen Hypothese, wenn die Kategorie der Regel (also die Kategorie auf der linken Seite der Regel) mit der Hypothese übereinstimmt.

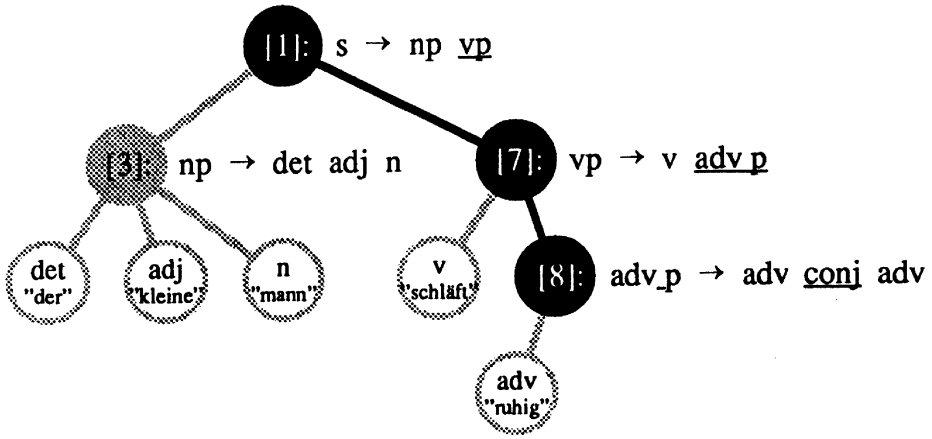
Betrachten wir beispielsweise die Grammatik (G) aus Abschnitt 3.4.2.2 und den Input (hier der Einfachheit wegen in segmentierter Schreibweise):

"Der kleine Mann schläft ruhig und tief" .

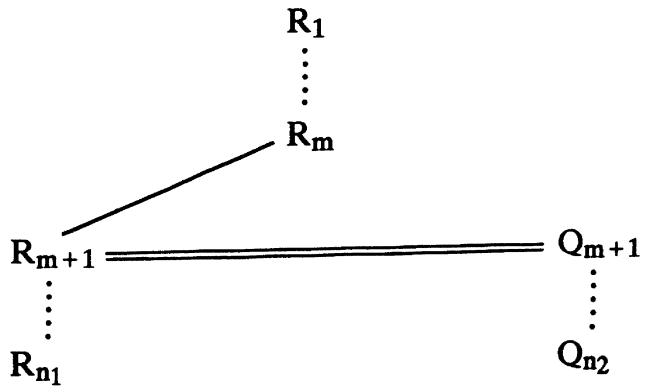
Durch die Analyse dieses Inputs würde zu dem Zeitpunkt vor der Analyse des vorletzten Wortes "und" die folgende Parsing-Alternative bestehen:

(25, ⟨ ([1], 2), ([7], 2), ([8], 2) ⟩)

In einem Baum wäre diese Alternative wie folgt darstellbar:[20]



"Baumkonstruktionskonvention" (vgl. untenstehendes Bild):



[20] Wir stellen auch die Vorgeschichte des Parsings dar, sie ist in dem Knoten mit der Regel [3] und den Wortknoten für die bereits segmentierten Wörter repräsentiert. Diese gehören nicht zu der als Parsing-Alternative definierten Struktur. In den zur Parsing-Alternative gehörenden Regeln [1], [7] und [8] ist eine Konstituente unterstrichen. Wir geben dadurch den zu diesen Regeln gehörenden Positionszähler an (im Beispiel ist der Zähler in allen Regeln – zufällig – gleich 2).

Wir halten es für wünschenswert, zwei verschiedene Parsing-Alternativen der Form
$(p_1, \langle (R_1, i_1), \ldots, (R_m, i_m), \ldots, (R_{n_1}, i_{n_1}) \rangle)$ und
$(p_2, \langle (Q_1, i_1), \ldots, (Q_m, i_m), \ldots, (Q_{n_2}, i_{n_2}) \rangle)$,
die in den ersten (m) Regel-Zähler-Paaren übereinstimmen, in einem Baum so zu darzustellen, daß – für k von 1 bis m – die Regeln R_k und Q_k (die ja gleich sind) in einem einzigen Knoten repräsentiert sind. Im Knoten für R_m (= Q_m) wird dann eine "Alternativen-Verzweigung" angebracht, so daß dieser Knoten zwei Teilbäume dominiert, deren Wurzeln mit den Regeln R_{m+1} und Q_{m+1} etikettiert sind. Diese Verzweigung, zusammen mit der oben gezeigten "syntaktischen" Verzweigung, erzeugt komplizierte Strukturen. Unser Parsing-Baum, den wir in Abschnitt 3.4.2.2 in einigen Beispielen abgebildet haben, hat diese beiden Verzweigungsrichtungen.[21]

Beschreibung des Parsing-Vorgangs

Wir geben nun zunächst eine Strategie an, die uns zu Beginn des Parsings eine Menge von "betrachteten" oder "etablierten" Parsing-Alternativen erzeugt. Bedingung ist, daß diese Parsing-Alternativen mit einem ersten Wort kompatibel sein müssen. Wir wollen jedes Wort, das als erstes Wort aus dem Input segmentiert werden kann, durch eine Parsing-Alternative berücksichtigen.[22]

(S0) Beginn des Parsings, Bottom-up-Strategie:

Für alle Wörter w aus L, deren Wortform mit einem Anfangssegment des Inputs übereinstimmt, setze l_w gleich der Länge der Wortform von w und etabliere alle Parsing-Alternativen $A_{w, R_1, \ldots, R_n} = (l_w, \langle (R_1, i_1), \ldots, (R_n, i_n) \rangle)$, für die gilt:
– $i_k = 1$ für alle k von 1 bis n,
– R_n dominiert die lexikalische Regel w an Stelle i_n (also an 1. Stelle).

Sprechweise: Wir sagen, die Alternativen $A_{w, \ldots}$ haben gerade w abgearbeitet.

Wende auf alle so erzeugten Parsing-Alternativen (S2) (s.u.) an, um zu einer neuen kategorialen Hypothese zu kommen. (Die Parsing-Alternativen $A_{w, \ldots}$ wurden gerade so konstruiert, daß w ihrer kategorialen Hypothese entspricht.)

---

[21] Wir versuchen, auch die Baumkonstruktionskonvention möglichst gut zu realisieren. Aus technischen Gründen (z.B. um die in Abschnitt 3.4.2.2 unter "active rules" genannte "Regel für einen konsistenten Parsing-Baum mit aktiven Knoten" aufrechtzuerhalten) müssen wir jedoch gelegentlich einige gleiche Regel-Zähler-Paare in zwei Knoten repräsentieren.

[22] Zu beachten ist jedoch auch eine syntaktische Restriktion: Das erste Wort muß Anfang eines nach G wohlgeformten Satzes sein. Diese Restriktion ist schon in der allgemeinen Definition einer Parsing-Alternative berücksichtigt, da die erste Regel in der Regel-Zähler-Folge eine 'S'-Regel sein muß. Wenn wir Parsing-Alternativen erzeugen wollen, dann nur solche, die der vollen Definition und damit auch dieser Einschränkung genügen.

Nachdem durch (S0) eine Menge von etablierten Parsing-Alternativen entstanden ist, versucht der Parsing-Vorgang durch wiederholte Anwendung der folgenden Regel, diese Alternativen weiterzuentwickeln. Dazu müssen im Lexikon die Wörter gesucht werden, die der kategorialen Hypothese der jeweiligen Parsing-Alternative entsprechen.

(S1) Treffer (engl. "match"):

Sei eine Parsing-Alternative $A = (p, \langle (R_1, i_1), \ldots, (R_n, i_n) \rangle)$ gegeben. Dann ermittle alle Wörter w aus L, deren Form mit einem Segment des Inputs ab p übereinstimmt und deren Kategorie gleich der kategorialen Hypothese von A ist.

Gibt es solche w aus L, so etabliere an Stelle von A für jedes w eine Parsing-Alternative A_w mit (l_w sei Länge der Wortform von w)
$A_w = (p+l_w, \langle (R_1, i_1), \ldots, (R_n, i_n) \rangle)$.
Analog zur in (S0) verwendeten Sprechweise hat A_w gerade w abgearbeitet. Wir wenden wieder (S2) auf alle diese A_w an.

Gibt es kein solches w aus L, versuche, die "repair"-Strategie (S4) auf A anzuwenden. Ist dies nicht möglich (vgl. die Eingangsbedingung der "repair"-Strategie), versuche, die Fragment-Strategie (S5) anzuwenden. Wenn auch dies nicht möglich ist (weil durch (S5) keine Parsing-Alternativen etabliert werden), dann eliminiere A.

Nachdem im Verlauf von (S1) passende Wörter im Lexikon gefunden worden sind – jedenfalls für einen Teil der Parsing-Alternativen –, ist deren kategoriale Hypothese erfüllt. Innerhalb von (S1) wird deshalb durch Anwendung der folgenden Regel eine neue kategoriale Hypothese, die des nächsten zu segmentierenden Wortes, ermittelt. Auch nach der Erzeugung von Parsing-Alternativen in (S0) mit Hilfe eines Anfangswortes muß bereits diese Regel angewendet werden.

(S2) nächste kategoriale Hypothese, Übergang zwischen Parsing-Zuständen:

Gegeben sei eine Parsing-Alternative $A = (p, \langle (R_1, i_1), \ldots, (R_n, i_n) \rangle)$. (In den Strategien (S0) und (S1) wurde die Sprechweise eingeführt, daß Parsing-Alternativen eine Regel "gerade abgearbeitet" haben. Dies ist der Ansatzpunkt dieser Strategie: Wird (S2) für eine Alternative A ausgeführt, so hat A gerade eine – lexikalische oder syntaktische – Regel abgearbeitet, die der kategorialen Hypothese von A entsprach.)
Betrachte dann an Stelle von A die neue Parsing-Alternative
$A' = (p, \langle (R_1, i_1), \ldots, (R_n, i_n+1) \rangle)$ und wende (S3) auf A' an.

Es muß jedoch sichergestellt werden, daß i_n nicht schon gleich der Länge von R_n war; sonst erhalten wir an der Position i_n+1 keine kategoriale Hypothese mehr, oder anders ausgedrückt, entspräche A' gar nicht mehr der Definition einer

Parsing-Alternative (vgl. die Einschränkung für die Zahlen i_k in der Definition).

Ist also schon i_n gleich der Länge von R_n, so sagen wir, die Regel R_n sei von A gerade insgesamt abgearbeitet. Dann betrachte an Stelle von A die (gekürzte) Parsing-Alternative $A^* = (p, \langle (R_1, i_1), \ldots, (R_{n-1}, i_{n-1}) \rangle)$ und wende (S2) nochmals auf A* an. (Beachte, daß R_n, die aus A "weggekürzte" Regel, der kategorialen Hypothese von A* entspricht.)

Es ist jedoch auch dann ein Problemfall möglich: Durch die Kürzung von A, insbesondere wenn sie durch rekursive Anwendungen dieser Strategie wiederholt wird, könnte eine "leere Parsing-Alternative" $(p, \langle \rangle)$ entstehen, wenn nämlich nicht nur i_n, sondern alle i_k gleich der Länge der Regeln R_k sind (für k von 1 bis n). In diesem Fall sind sämtliche Regeln von A abgearbeitet, und es ist zu prüfen, ob auch der Input zu Ende analysiert ist.

Ist also nicht nur i_n gleich der Länge von R_n, sondern i_k gleich der Länge von R_k für alle k von 1 bis n, so gilt folgende Fallunterscheidung:

- Ist p gleich der Länge des Inputs, so nennen wir A erfolgreich,
- sonst versuchen wir (S4) auf A anzuwenden.

Unabhängig davon ist A – bzw. die in (S4) unter 1. erzeugte Alternative A', die sich in der Regel-Zähler-Folge nicht von A unterscheidet – zu eliminieren.

Die folgende Strategie ist ein wichtiger Teilschritt bei der Ermittlung einer neuen kategorialen Hypothese. Weil sie nicht nur innerhalb von (S2), sondern auch innerhalb der "Reparaturstrategie" (S4) gebraucht wird, ist sie hier gesondert aufgeführt (auch aus Gründen der Übersichtlichkeit). Die Regel behandelt Parsing-Alternativen, deren (neu ermittelte) kategoriale Hypothese nicht-lexikalisch ist. In diesem Fall kann in (S1) noch kein passendes Lexikonelement gefunden werden; statt dessen sind Syntaxregeln aufzusuchen, welche die momentane kategoriale Hypothese expandieren.

(S3) Expansion der syntaktischen kategorialen Hypothese, Top-down-Strategie:

Wenn eine Parsing-Alternative $A = (p, \langle (R_1, i_1), \ldots, (R_n, i_n) \rangle)$ eine lexikalische Kategorie als kategoriale Hypothese hat, behalte A unverändert bei.

Sonst betrachte an Stelle von A alle Parsing-Alternativen der Form
$A_{Q_1, \ldots, Q_m} = (p, \langle (R_1, i_1), \ldots, (R_n, i_n), (Q_1, 1), (Q_2, 1), \ldots, (Q_m, 1) \rangle)$,
deren kategoriale Hypothese lexikalisch ist.

(Beachte, daß aufgrund der allgemeinen Definition von Parsing-Alternativen sicherzustellen ist, daß R_n an Stelle i_n Q_1 dominiert, ebenso Q_1 an 1. Stelle Q_2 usw., und daß aufgrund der Definition der kategorialen Hypothese Q_m als erste Konstituente auf der rechten Regelseite eine lexikalische Kategorie haben muß.)

Wir verlangen nicht, daß m, die Anzahl der zu A hinzugefügten Regeln, für alle durch diesen Vorgang erzeugten Alternativen gleich sei.

Im Gegensatz zur Behandlung von lexikalischen kategorialen Hypothesen in (S1)

müssen wir hier nicht sicherstellen, daß die gewünschte Erweiterung der Regelfolge bis hin zu einer lexikalischen Hypothese tatsächlich existiert. Wir erwarten nämlich von jeder wohlgeformten Grammatik, daß sich solche Regelerweiterungen finden lassen. Bei Implementierung von (S3) muß jedoch auf die Möglichkeit einer Linksrekursion in G geachtet werden.

Mit den bisher betrachteten Regeln ist bereits ein Parser für Inputs, die üblichen grammatischen Wohlgeformtheitskriterien entsprechen, definiert. Durch die Angabe der folgenden zwei Strategien erweitern wir das Konzept der Wohlgeformtheit, um zu einer Simulation der von uns behandelten Reparaturphänomene zu gelangen (zu den Ergebnissen der Simulation vgl. Abschnitt 3.4.1.4).

(S4) Reparaturstrategie, Übergang zwischen Regelfolgen:

Gegeben sei eine Parsing-Alternative $A = (p, \langle (R_1, i_1), \ldots, (R_n, i_n) \rangle)$, wobei ein Versuch, (S1) (die grammatische Segmentierung) auf A anzuwenden, erfolglos war. Falls nun im Input als Segment ab p das Hesitationssignal "aeh" zu finden ist, setze h gleich der Länge von "aeh", $p' = p+h$, und etabliere an Stelle von A

1. die Alternative $A' = (p', \langle (R_1, i_1), \ldots, (R_n, i_n) \rangle)$,
   (vgl. "keine Veränderung" in Abschnitt 3.4.2.2 unter "Reparaturanalyse")

2. für alle k von 1 bis $(i_n - 1)$ die Alternative
   $A_k = (p', \langle (R_1, i_1), \ldots, (R_n, k) \rangle)$

   sowie für alle j von 1 bis $(n-1)$ und alle k_j von 1 bis $(i_j - 1)$ die Alternativen
   $A_{k_j}^{(j)} = (p', \langle (R_1, i_1), \ldots, (R_k, k_j) \rangle)$

   (vgl. a.a.O., "Zurückgehen in dieser und den dominierenden Regeln"),

3. alle Alternativen B, die ursprünglich durch Anwendung von (S0) erzeugt, jedoch während der wiederholten Anwendung von (S1) eliminiert wurden, bevor sie den Input bis zum p-ten Graphem analysieren konnten. Wir betrachten B direkt vor der Eliminierung (die innerhalb von (S1) oder (S2) stattfand) und wenden (S4) 1. und 2. auf B an.
   (vgl. a.a.O., "Aktivierung bereits verworfener Alternativen"),

4. "strukturelle Alternativen": Sei K die Kategorie von R_n. Für jede Regel R aus S, in deren rechter Seite an Stelle i_R eine Kategorie K' steht, welche mit K in der Hauptkategorie übereinstimmt, etabliere alle zulässigen Parsing-Alternativen der Form
   $A_{Q_1, \ldots, Q_m, R} = (p', \langle (Q_1, 1), \ldots, (Q_m, 1), (R, i_R) \rangle)$
   und wende (S3) auf diese an.

Beachte, daß analog zu (S0) impliziert wird, daß Q_1'S'-Regel ist und jede Regel die nachfolgende dominiert. Die kategoriale Hypothese vor Anwendung von (S3) ist die nicht-lexikalische Kategorie K'.

(S5) Fragmentstrategie:

Sei eine Parsing-Alternative $A = (p, \langle (R_1, i_1), \ldots, (R_n, i_n) \rangle)$ gegeben. Etabliere an Stelle von A alle Parsing-Alternativen
$A_f = (p+f, \langle (R_1, i_1), \ldots, (R_n, i_n) \rangle)$ (f sei natürliche Zahl, also mindestens 1), für die gilt:

- das Input-Teilsegment von p (einschließlich) bis p+f (ausschließlich) stimmt überein mit den ersten f Graphemen der Form eines Wortes, welches die kategoriale Hypothese von A als Wortkategorie hat.

Auf diese A_f ist (S1) anzuwenden.

Eigentlich ist eine weitere Bedingung für A_f, daß ein neuer Segmentierungsversuch (mit der alten kategorialen Hypothese) erfolgreich ist. Diesen Versuch unternimmt aber gerade die Strategie (S1), welche alle A_f "durchlaufen" müssen. Dadurch werden die in (S5) erzeugten Parsing-Alternativen, die diese "Segmentierungsbedingung" nicht erfüllen, wieder eliminiert. Eine weitere, sich natürlicherweise ergebende Bedingung ist, daß p+f höchstens gleich der Länge des Inputs sein kann, damit noch mindestens ein Graphem für den Segmentierungsversuch zur Verfügung steht.

Noch einmal kurz der Ablauf des Parsing-Prozesses:

- (S0) liefert eine Anfangsmenge von ("betrachteten") Parsing-Alternativen.
- Diese Menge betrachteter Parsing-Alternativen ändert sich durch wiederholte Anwendung von (S1) auf ihre Elemente. Querverweise in (S0) und (S1) führen zur Anwendung der anderen angegebenen Strategien.

3.4.3 Einbettung des "repair"-Parsers in die Parser-Topographie

Um einschätzen zu können, wie sich der vorliegende Parser zu anderen Paradigmen verhält, ist es vielleicht nützlich, ihn mit bekannten Ansätzen, Early (1969), Kay (1980) und Tomita (1987) zu vergleichen. Dies soll in Form einer Tabelle geschehen. Sie enthält Einträge für Merkmale, welche die Leistungsfähigkeit von Parsern charakterisieren. Eine Spalte "GR" ist für die dem Parser zugrundeliegende Grammatik vorgesehen. "L" bezeichnet die analysierte Sprache, der dabei wichtigste Punkt ist die Analyse ambiger Ausdrücke und solcher, die Störungen enthalten. "dist" bedeutet, daß sowohl Ambiguität als auch Störung vorliegt.

Unter dem Label "Input" wird angezeigt, ob dieser in seg[mentierter] oder nicht-

seg[mentierter] Form gegeben werden muß. "Inp.-R." erlaubt eine Spezifikation in "online" und "stat[isch]". "online" bedeutet, daß der Parsingprozeß mit dem Eintippen des ersten Graphems beginnt, also bevor das erste Wort noch vollständig vorliegt. Dagegen meint "statisch", daß ein insgesamt vorliegender Input analysiert wird. "P.-Str." bezeichnet die verwendeten Parsing-Strategien, unterschieden werden hier "top-d[own]", "bot[tom]-up" und "mixed" für beides. "P.-St." enthält Information über das Instrument, das den Parsingprozeß jeweils steuert und "P.-H." gibt Aufschluß über die gewählte Repräsentation der Parsing-Geschichte.

	GR	**L**	**Input**	**Inp.-R.**	**P.-Str.**	**P.-St.**	**P.-H.**
Early	CFPSG	ambige CFL	seg	stat	top-d.	chart*	chart*
Kay	CFPSG	ambige CFL	seg	stat	mixed	chart** table*	config
Tomita	CFPSG	ambige CFL	seg	online	bot.-up	table**	stack
Lisken/ Rieser	CFPSG	ambige CFL	nicht-seg	stat	mixed	aktive Regeln	Parsing -Baum

Folgende Erläuterungen zur Tabelle seien noch gegeben: "CFPSG" bedeutet (wie in 3.4.1.1) "kontextfreie Phrasenstrukturgrammatik" und "CFL" "kontextfreie Sprache".

Eine chart* nach Early läßt sich am besten anhand der folgenden PSG erläutern:

(G') [1] S → NP VP
[2] NP → Max
[3] VP → V
[4] V → schläft

Gegeben (G') und der Input "Max schläft", erzeugt ein Early-Parser nach den Regeln (E1), (E2), (E3) eine chart*:[23]

[23] Ein Punkt '.' auf einer rechten Regelseite zeigt an, daß die Konstituenten links davon bereits geparst und diejenigen rechts davon noch zu parsen sind. Eine Ziffer neben einem Ausdruck gibt an, in welcher Spalte dieser eingeführt wurde.
' S → NP . VP, 0 ' bedeutet demnach, daß der betrachtete Input von der NP geparst und der Ausdruck in Spalte 0 eingeführt wurde.

(E1) Treffer: Falls 'a' das aktuelle Inputsegment ist und die letzte Spalte einen Ausdruck der Form ' A → k . a m ' enthält mit 'k' oder 'm' möglicherweise 0, dann schreibe in die aktuelle Spalte einen Ausdruck der Form ' A → k a . m '.

(E2) Agenda: Enthält die aktuelle Spalte einen Ausdruck der Form ' A → a . m ', der anzeigt, daß eine Regel vollständig abgearbeitet ist und die letzte Spalte eine Regel der Form ' C → k . A m ', dann füge in der aktuellen Spalte die Information, daß die Konstituente 'A' geparst ist, in der Form ' C → k A . m ' hinzu.

(E3) Top-down-Hypothesen: Enthält eine Spalte einen Eintrag, der anzeigt, daß ein nicht-terminales Element abgearbeitet werden muß, so füge alle Expansionen dieses Elements in der Spalte hinzu.

(E4) Starten und Beenden der Parsing-Prozedur:
Zu Beginn werden die Spalten 0 und 1 bis n etabliert (n = Anzahl der Elemente im Input). Spalte 0 enthält alle top-down-Hypothesen aus G', d.h. alle S-Regeln sowie alle Regeln, welche deren jeweils erste Konstituente auf der linken Seite expandieren. Alle diese Regeln werden auf der rechten Seite vor der ersten Konstituente mit einem '.' markiert. Jedem Wort des Inputs ist, beginnend mit 0, eine Spalte zugeordnet.

Ein Input-String ist erfolgreich geparst, wenn die Spalte n für das letzte Wort des Inputs eine S-Regel aus Spalte 0 enthält mit '.' nach dem letzten Symbol.

(E1) bis (E4) haben trotz der unterschiedlichen Parsing-Strategien (cf. Tabelle) enge Entsprechungen zu den von uns entwickelten Instrumentarien. (E1) entspricht dem Einfügen eines terminalen Elements in den Parsing-Baum, (E2) dem Anzeigen, daß zur nächsten Konstituente übergegangen werden muß, (E3) der Expansion von nicht-terminalen Kategorien im Parsing-Baum. Dagegen starten wir den Parsing-Prozeß mit einer Wortsegmentierung ohne kategoriale Hypothese.

Zu table*: Kay sieht die Verwendung folgender Alternativen vor: Einfache "reachability tables", die für jedes Nonterminal A eine Menge R(A) angeben. R(A) enthält diejenigen terminalen und nicht-terminalen Symbole, die in einer von A dominierten Kette erstes Element sein können. Dies entspricht unserer Redeweise von "left corners". Ein etwas komplizierterer Mechanismus sind Kaysche "rule-selection tables". Eine derartige Tabelle wird als m × n-Array realisiert, wobei m die Anzahl der weder präterminalen noch terminalen Symbole ist und n die Anzahl aller nicht-terminalen Symbole. Ein Feld S(A,B) einer Tabelle S enthält die Liste der Grammatikregeln, in denen A dominierendes und B dominiertes Symbol ist, d.h. alle Regeln der Form A → B C ... mit B aus R(A). Dies korrespondiert im Falle der bottom-up-Strate-

gie zu unserem oben skizzierten Konzept des Suchens aller anwendbaren PSG-Regeln, die im Parsing-Baum entsprechend ihrer Dominanzhierarchie angeordnet werden.

0	1	2
Max	**schläft**	
Hypothesen: S → . NP VP, 0 NP → . Max, 0	Treffer: NP → Max ., 0 Agenda: S → NP . VP, 0 Hypothesen: VP → . V, 1 V → . schläft, 1	Treffer: V → schläft ., 1 Agenda: VP → V ., 1 S → NP VP ., 0

"charts**" enthalten zusätzlich Angaben über abgeschlossene und nicht-abgeschlossene syntaktische Strukturen, Positionsinformation u.a.m. Sie dienen dazu, den Parsing-Prozeß zu optimieren. Durch ihren Einsatz kann die wiederholte Analyse bereits geparster Ausdrücke vermieden werden. "reachability tables", "rule selection tables" und "charts" werden von Kay zur Formulierung von Parsing-Prozeduren verwendet. "configs", i.e. "configuration tables", beschreiben den Parsing-Prozeß. Sie enthalten die aufeinanderfolgenden Parsing-Zustände und die Angabe der Übergangsregel von einem Parsingzustand zu seinem Nachfolger. Information dieser Art ist bei uns im Parsing-Baum enthalten. Der Parsing-Zustand muß bei uns an der Regel, der gerade bearbeiteten Konstituente und den der aktuellen Regel übergeordneten Regeln abgelesen werden.

Tomitas "table**"-Konzeption unterscheidet sich etwas von derjenigen Kays, da er einen bottom-up-shift-reduce-Parser zugrundelegt, während Kay Parser-Algorithmen durch allgemeine Schemata spezifiziert. Die table** schreibt für jedes grammatische Symbol vor, in Abhängigkeit von welchem Parserzustand welche Aktion ("shift", "reduce", Ansteuern des nächsten Parserzustandes) ausgeführt werden muß. Mit jeder Aktion ist eine Anweisung für den Aufbau eines stack, genannt "graph-structured stack" verbunden. Aufgrund des "graph-structured stack" kann der "shift-reduce-"-Parser alternative Parsing-Prozesse verfolgen, ohne daß ein Input-Segment zwei-

mal auf dieselbe Art und Weise analysiert würde. Der Tomita-Parser kann dem Typ der "chart"-Parser zugeordnet werden. Die Entwicklungslinie, die zum Konzept der "graph-structured stacks" führt, hat ihre Ursache in der Behandlung von Non-Determinismus-Problemen und wird von Tomita wie folgt beschrieben: 1. Das System verwaltet eine Anzahl von Stacks gleichzeitig; 2. Stacks werden als "trees (or a forest)" repräsentiert; 3. eine Verallgemeinerung von 2. ist der "graph-structured stack". Diese Idee ist ähnlich derjenigen, die wir mit den Alternativen im Parsing-Baum verfolgen. Die Redeweise von "stacks" ist in diesem Zusammenhang eher metaphorisch. Auch bei uns liegt eigentlich ein Graph vor, wie die Beschreibung des Parsing-Baums in Abschnitt 3.4.2.1 zeigt. Wir befleißigen uns nur der etwas bequemeren Redeweise von Bäumen oder "baumähnlichen" Strukturen.

3.5 Der innergrammatische Ansatz

3.5.1 Grundlagen einer innergrammatischen Modellierung von Reparaturen

Der im Projekt entwickelte grammatiktheoretische Ansatz geht von der These aus, daß die zur Beseitigung von Inkohärenzen satzintern durchgeführten Reparaturen nur Konstruktionsverfahren und Verarbeitungsstrategien verwenden, die dem Sprachverarbeitungssystem ohnehin zur Verfügung stehen. Bezogen auf die grammatische Verarbeitung bedeutet dies, daß der Produktion und Rezeption satzinterner Reparaturen bekannte syntaktische Konstruktionen zugrunde liegen müssen. Zur Präzisierung des schon von Levelt (1983) aufgezeigten Zusammenhangs zwischen Reparaturen und Koordinationskonstruktionen kann einerseits die in Abschnitt 3.1 erwähnte Unterscheidung von *Nachtrags-* und *Überbrückungsreparaturen* (Kindt & Laubenstein 1990) und andererseits eine Typologie für Koordinationskonstruktionen (Günther et al. 1991) herangezogen werden. Dabei zeigt sich, daß Nachtragsreparaturen in den meisten Fällen eine zur *Phrasenkoordination* analoge Konstruktion benutzen und Überbrückungsreparaturen eine noch genauer zu analysierende Variante der *linksausklammernden Koordination*. Die so präzisierte Korrespondenz zwischen Reparaturen und Koordinationskonstruktionen läßt sich mit folgenden Beispielpaaren veranschaulichen.

(3.14) P: und DEN LINKEN- NEIN DEN ROTEN stellst du links hin
(vgl. Beispiel 3.5)

(3.14') P: und DEN LINKEN ODER DEN ROTEN stellst du links hin

(3.15) P: gut und jetzt legste auf diese blauen Klötzchen [...] rechts und links je EIN BLAU/ EIN GRÜNES KLÖTZCHEN
(Forschergruppe Kohärenz 1987, S. 34)

(3.15') P: gut und jetzt legste auf diese blauen Klötzchen [...] rechts und links je EIN BLAUES ODER EIN GRÜNES KLÖTZCHEN

Im Gegensatz zu Nachtragsreparaturen sind bei Überbrückungsreparaturen allerdings selbst lokale syntaktische Inkohärenzen zu bearbeiten. Daher stellt sich im Vergleich zum Modellierungsansatz von Abschnitt 3.4 die Frage, inwiefern Überbrückungsreparaturen durch einen innergrammatischen Ansatz zu erfassen sind, d.h. auf welche Weise syntaktische Inkohärenzen im Rahmen der üblichen grammatischen Verarbeitung beseitigt werden können. Für eine Beantwortung dieser Frage ist ein Vergleich von Überbrückungsreparaturen mit dem Phänomen sogenannter Garden-Path-Sätze hilfreich.

(3.16) Alkohol entfernt mit einem Lappen hinterläßt keine Flecken

Für eine Modellierung der Rezeption von Garden-Path-Sätzen ergibt sich das Problem, daß aufgrund der inkrementellen Verarbeitung zu einem bestimmten Zeitpunkt syntaktische Kategorisierungen und Strukturierungen vorgenommen werden, die später aufgrund neuer Informationen über die zu verarbeitende Äußerung zu revidieren sind. Die zugrundeliegende Strategie einer Äußerungsverarbeitung trotz unvollständiger Information führt also auch bei der grammatischen Verarbeitung zu dem für Inferenzsysteme bekannten Phänomen der *Nichtmonotonie*. Genau derselbe Effekt tritt bei der inkrementellen syntaktischen Verarbeitung von Überbrückungsreparaturen auf.

(3.17) R: also dann weiß ich versteh ich den Unterbau nich
(Forschergruppe Kohärenz 1987, S. 28)

Weder der Produzent noch der Rezipient wissen bei einer inkrementellen Verarbeitung von (3.17) zwangsläufig im voraus, daß die Sequenz weiß ich zu einem Konstruktionsabbruch führt und daher – retrospektiv betrachtet – nicht den üblichen syntaktischen Status besitzt, d.h. nicht in üblicher Weise mit dem Äußerungsbeginn verknüpft werden darf. Prospektiv beurteilt ist das Auftreten eines Konstruktionsabbruchs aber nicht antizipierbar. Deshalb ist es zweckmäßig, die betreffende Sequenz zunächst in gewohnter Weise in eine Konstruktion mit dem Äußerungsbeginn einzubinden. Diese Vorgehensweise macht allerdings eine spätere syntaktische Reorganisation erforderlich. Insofern liegt ebenfalls ein Nichtmonotoniephänomen vor.

Der wesentliche Ansatzpunkt zur Wahl einer innergrammatischen Modellierung von Überbrückungsreparaturen ist also dadurch gegeben, daß Rezipienten auch im Rahmen grammatischer Verarbeitung über die Fähigkeiten einer nichtmonotonen Informationsableitung verfügen. Diese Fähigkeit bleibt unberücksichtigt, wenn man generell eine Parsingstrategie ansetzt, an Stellen mit mehreren syntaktischen Analysemöglichkeiten zunächst alle Alternativen parallel zu verfolgen und/oder eine Analyseentscheidung hinauszuschieben, bis ausreichende Informationen für eine eindeutige Auswahl vorliegen. Auch in Sätzen des folgenden Typs wird im Gegensatz zur Vorgehensweise bei einer Aufschubstrategie nicht mit der syntaktischen Desambiguierung gewartet, bis aufgrund von Kontextinformationen Eindeutigkeit hinsichtlich der Subjekt-

Objekt-Kategorisierung hergestellt ist.

(3.18) Die Studentin beobachtet das Kind

Somit muß auch unabhängig vom Phänomen der Reparaturen modelliert werden, daß Rezipienten im Fall unterschiedlicher syntaktischer Analysemöglichkeiten aufgrund bestimmter Präferenzen Auswahlentscheidungen treffen, daß sie die eventuelle Inadäquatheit solcher Entscheidungen aber später erkennen und durch einen Reorganisationsprozeß korrigieren können. Zugleich darf angenommen werden, daß die Verarbeitung von Überbrückungsreparaturen nach denselben Prinzipen verfährt und daß die betreffenden Verarbeitungs- und Kontrollaufgaben von einem gemeinsamen Teilsystem der grammatischen Verarbeitung wahrgenommen werden.

3.5.2 Inkohärenz in Zuordnungssystemen

Die Entwicklung einer Reparaturtheorie bedarf einer wesentlich expliziteren verständigungstheoretischen Fundierung als bei Schegloff (1979) oder Levelt (1983), weil Reparaturen nur einen Spezialfall der Behandlung von Verständigungsproblemen bilden. Im Sinne von Kindt und Weingarten (1984) sind Verständigungsprobleme als
Zuordungskoordinationsprobleme aufzufassen. Deshalb wird in Kindt und Laubenstein (1990) ein zuordnungstheoretischer Rahmen für die Analyse von Reparaturen entwikkelt. In diesem Rahmen können alle, also auch die grammatischen Prozesse eines Verarbeitungssystems als Zuordnungsschritte interpretiert werden. Inkohärenz ist dann explizierbar als 'nicht erwartungsgemäße' Zuordnung. Genauer befindet sich ein Verarbeitungssystem, das bestimmte Zuordnungsaufgaben durchführt, genau dann im Zustand der Inkohärenz, wenn mindestens für eine der Zuordnungsaufgaben noch keine stabile Zuordnung erreicht ist. Ausgehend von dieser Charakterisierung kann man – wie schon in Abschnitt 3.1 skizziert – verschiedene Inkohärenztypen unterscheiden. Im Prinzip sind aber alle diese Typen auf eine durch Zuordnungsinkonsistenz bedingte Instabilität zurückzuführen. Beispielsweise entsteht bei Suchproblemen dadurch eine Inkonsistenz, daß einerseits zu dem betreffenden Zeitpunkt das Resultat der Zuordnungsaufgabe als leer kategorisiert wird, andererseits aber gleichzeitig eine nichtleere Kategorisierung erwartet wird (vgl. Beispiel 3.3).

Durch die lokale Betrachtung einzelner Zuordnungen darf nicht der Eindruck entstehen, als bestünde die Funktion eines Verarbeitungssystems ausschließlich in der Durchführung isolierter Zuordnungsaufgaben. Vielmehr sind Zuordnungsprozesse bei der Sprachverarbeitung immer in vielfältiger Weise über wechselseitige Erwartungen miteinander gekoppelt. Daher wird Inkohärenz oft dadurch verursacht, daß die Lösung einer Zuordnungsaufagbe durch zusätzliche Anforderungen 'benachbarter' Zuordnungsschritte erschwert wird. Umgekehrt ist im Sinne der Darstellung von

Abschnitt 2.3.1 Kohärenz dann erreichbar, wenn alle miteinander verknüpften Zuordnungsprozesse 'harmonisieren'. Dies impliziert aber auch die positive Möglichkeit, daß lokale Inkonsistenzen durch den Einfluß stabiler benachbarter Zuordnungen 'überspielt' werden, wie dies z.B. beim Überlesen von Druckfehlern der Fall ist.

3.5.3 Modellentwicklung

Aus der in den Abschnitten 3.5.1 und 3.5.2 dargestellten Zielsetzung ergibt sich die Aufgabe, eine für die Modellierung von Reparaturen geeignete Grammatikkonzeption und einen zugehörigen, zuordnungstheoretisch formulierten Grammatikausschnitt zu entwickeln. Hierfür wurde auf den für Ellipsenkonstruktionen entwickelten Syntaxansatz von Kindt (1985) zurückgegriffen, der in derzeit laufenden Forschungsarbeiten zur "Mehrdimensionalen Schaltgrammatik" fortgeschrieben wird (Kindt 1991a, 1991b). Diese Grammatikkonzeption postuliert grundsätzlich die Notwendigkeit einer Betrachtung zweidimensionaler Verknüpfungsstrukturen (Argument- und Wertverknüpfung; Kindt 1985) und konkretisiert dies durch Aussagen über jeweils unterschiedliche Strukturen für die einzelnen Typen von Koordinationskonstruktionen (vgl. auch Günther et al. 1991). In analoger Weise werden dann erstens Strukturhypothesen für Reparaturen aufgestellt und Aussagen darüber gemacht, welche grammatischen Konstruktionen der Beseitigung von Inkohärenzen zugrunde liegen. Zweitens ist insbesondere für Überbrückungsreparaturen im Detail zu rekonstruieren, welche Zuordnungsprozesse zum Auftreten von Inkonsistenzen führen und aufgrund welcher Prinzipien und zugehöriger Zuordnungsschritte solche Inkonsistenzen beseitigt werden. Zu beiden Programmpunkten sollen nachfolgend einige Ergebnisse skizziert werden.

Wie schon in Abschnitt 3.1 dargestellt wurde, eignen sich Nachtragsreparaturen nur zur Behandlung rein inhaltlicher Äußerungsstörungen; folglich liegt bei ihnen auf syntaktischer Ebene keine Inkohärenz vor. Trotzdem kann man fragen, nach welchem Prinzip die zu diesem Reparaturtyp gehörige Konstruktion Einfluß auf zugrundeliegende inhaltsbezogene Zuordnungen nimmt. Diesbezüglich wird bei Nachtragsreparaturen ähnlich wie bei Phrasenkoordinationen als Verknüpfungsstruktur eine *Parallelverknüpfung* der beiden Konstruktionsteile Reparaturbezug und Reparaturversuch (Abschnitt 3.1) angenommen, so daß sich die zugehörige Interpretation der Gesamtkonstruktion als Komposition aus den Interpretationen der beiden Konstruktionsteile ergibt. Für ein Beispiel wie (3.14) bedeutet das, daß `den linken- NEIN den roten` wie `NICHT den linken SONDERN den roten` zu interpretieren ist, daß diese Interpretation aber sukzessiv in einem nichtmonotonen Informationsprozeß konstruiert wird. Auf diese Weise ergibt sich die gewünschte Zuordnungsmodifikation.

Bei Überbrückungsreparaturen liegt demgegenüber eine lokale formale Äußerungsstörung von der Art vor, daß bei der inkrementellen Verarbeitung ein neuer Äußerungsteil nicht in die bisher erwartete Konstruktionsfortsetzung paßt. Beispielsweise ist das Auftreten des Hesitationssignals `eh` im allgemeinen inkonsistent zu der Erwar-

tung, daß als nächstes Äußerungselement ein Wort folgt. Eine solche Inkonsistenz kann nur durch eine Konstruktionsreorganisation aufgelöst werden, die z.B. dem Hesitationssignal einen anderen syntaktischen Status ermöglicht. Als geeignete syntaktische Konstruktion kommt hierfür die *Parenthese* in Frage, die an bestimmten Positionen innerhalb von Sätzen eingeschoben werden kann, ohne daß dadurch syntaktische Abhängigkeitsbeziehungen etabliert werden. Zugleich unterliegt diese Konstruktion intern nicht den üblichen Wohlgeformtheitsbedingungen. Die Möglichkeiten einer Parenthesebildung sind aber neben den positionellen Restriktionen auch noch durch die Forderung eingeschränkt, daß die Äußerungsfortsetzung nach der Parenthese mit dem Äußerungsbeginn vor der Parenthese rückverknüpft werden muß. Eine solche Rückverknüpfung ist aber nur dann möglich, wenn die zu überbrückende lineare Distanz klein, d.h. wenn die Parenthese kurz ist oder wenn andernfalls ein geeignetes zusätzliches Überbrückungsverfahren angewendet wird. Ein analoges Überbrükkungsproblem besteht auch für linksausklammernde Koordinationskonstruktionen. Von dieser Konstruktion kann für Überbrückungsreparaturen das Verfahren einer *Mehrfachverknüpfung* (Günther et al. 1991) unter der Bedingung der *funktionalen Äquivalenz* übernommen werden; eine Verknüpfung zwischen Reparaturbezug und Reparaturversuch als Pendant zu der Verknüpfung, wie sie durch die Konjunktion zwischen den Konjunktionsteilen einer Koordination vermittelt wird, findet aber nicht statt. Insgesamt gesehen wird die einer Überbrückungsreparatur zugrundeliegende lokale Inkohärenz also dadurch aufgelöst, daß der zugehörige gestörte Äußerungsteil als Parenthese syntaktisch isoliert und von den Erwartungen der Zuordnungsprozesse des ungestörten Äußerungsbeginns 'abgekoppelt' wird. Dies bedeutet aber nicht, daß die Parenthese grundsätzlich keinen Beitrag zur Kohärenz der Gesamtäußerung leisten könnte und somit in jedem Fall funktionslos wäre. Ein diesbezüglich interessantes Gegenbeispiel bildet (3.19):

(3.19) R: mhm (4 sec. Pause) aber mir is noch nich ganz klar wie ich also DIE BEIDEN ÄUSSEREN ELEMENTE . eh wie dicht DIE an dem mittleren Element . dran sind
(Forschergruppe Kohärenz 1987, S. 33)

Hier wird eine semantischen Erwartungen entsprechende spezifische Interpretation von die im Reparaturversuch nur durch Herstellung von Koreferenz mit die beiden äußeren Elemente im Reparaturbezug ermöglicht, und somit bliebe die Äußerung ohne die Informationen des Reparaturbezugs inkohärent.

3.5.4 Parserentwicklung

Einige der Grundkonzepte für die Parserentwicklung basieren auf Ideen aus Kindt (1985, 1991a, 1991b; vgl. auch Eikmeyer, Kindt, Laubenstein, Polzin, Rieser &

Schade, 1990). Zur Konzeption der dort angeregten Mehrdimensionalen Schaltgrammatik gehört die Vorstellung, daß sich die Sprachverarbeitung auf verschiedenen Ebenen (in verschiedenen Dimensionen) parallel vollzieht. Zu diesen Dimensionen gehören beispielsweise im Bereich der Syntax "Konstituenz" und "Valenz". Die Informationen aus den einzelnen Dimensionen interagieren miteinander: sie ergänzen, beeinflussen und kontrollieren sich gegenseitig. Die Integration dieser verschiedenen Informationen geschieht dabei ausschließlich auf der Ebene der Wörter. Es existiert keine direkte Konzeptualisierung der oberhalb der Wortebene angesiedelten syntaktischen Einheiten wie Nominalphrasen, Präpositionalphrasen usw. Die Informationsgewinnung und -verarbeitung geschieht in allen Dimensionen nach den gleichen Prinzipien: Mit Hilfe von Ableitungsregeln werden aus Konstellationen bekannter Merkmale Informationen über weitere Merkmale gewonnen. Beispiele für solche Merkmale auf syntaktischer Ebene sind SINGULAR, NOMEN, PLURALBILDUNG_MIT_UMLAUT. Aus einer Ableitungsregel, wie z.B. DETERMINER + FLEXIONSMORPHEM_ER + PLURAL -> GENITIV, ergibt sich, daß, wenn die drei links vom Pfeil stehenden Merkmale vorliegen, auch das Merkmal GENITIV vorliegt.

Zur Veranschaulichung einiger Konzepte der Mehrdimensionalen Schaltgrammatik wurde im Projekt ein Parserprototyp entwickelt und implementiert (Laubenstein 1990). Diese Parserstudie ist in wesentlichen Punkten (Dimensionalität, Lexikon, Syntaxregeln) stark eingeschränkt und dient lediglich zur Veranschaulichung einiger grundlegender Verarbeitungsstrategien am Beispiel einfacher Nominalphrasen, die jeweils wortweise eingegeben werden. Die Implementierung erfolgte in Form eines konnektionistischen Netzwerkes, da sich u.a. die Grundentitäten der Mehrdimenionalen Schaltgrammatik, also Merkmale und Ableitungsregeln, relativ einfach in eine Netzwerkarchitektur übersetzen lassen und weil gleichzeitig ein weiteres Grundprinzip der Mehrdimensionalen Schaltgrammatik, die parallele Verarbeitung aller vorliegenden Informationen, in einer Netzwerkarchitektur automatisch gegeben ist.

Die jeweiligen Merkmale werden in dieser Architektur durch entsprechend benannte Knoten repräsentiert, und die Regeln werden durch Verbindungen zwischen den einzelnen Knoten abgebildet. Eine Regel A + B -> C wird also durch aktivierende Verbindungen der Knoten A und B auf den Knoten C wiedergegeben. Einander ausschließende Merkmale wie, z.B. Singular und Plural, führen zu symmetrischen inhibitorischen Verbindungen zwischen den zugehörigen Knoten.

Eine Implementierung von Präferenzregeln, wie sie im vorherigen Abschnitt am Beispiel (3.18) diskutiert wurden, ist in einem konnektionistischen Netzwerk mit relativ einfachen Mitteln zu erreichen. Das Netzwerkmodell erlaubt es, durch geringfügig unterschiedliche Leitungsstärken eine Interpretation zu bevorzugen, ohne auszuschließen, daß diese später nicht mehr 'überschrieben' werden kann, sofern nämlich 'harte' Informationen (in der konnektionistischen Implementierung repräsentiert durch deutlich stärkere Aktivierungen) eine andere Lesart erzwingen. Die Regeln zur Analyse der Wortform die sehen beispielsweise (stark vereinfacht) so aus:

```
die       ->  DAS + E
DAS       ->  DET
DET + E   ->  NOM | GEN | AKK
```

Da keine Entscheidung zwischen NOM, GEN und AKK möglich ist, werden diese drei Knoten von DET und E wesentlich schwächer aktiviert als z.B. DET von DAS. Da aber der präferierte NOM-Knoten etwas stärker aktiviert wird als GEN und AKK, wird dieser Knoten letztendlich zum höchstaktivierten Kasusknoten – sofern keine anderen Aktivierungen z.B. von einem benachbarten Wort auf AKK eintreffen.[24]

Als Testfall für die Revision von Entscheidungen bei Verwendung nichtmonotoner Regeln versucht der Parser zunächst, sämtliche eingegebenen Wörter als Teil einer einzigen Konstituente zu interpretieren, d.h. die im Input benachbarten Wörter tauschen Kasus-, Numerus- und Genusinformationen untereinander aus. Dabei wird jedes Wort in einem eigenen Teilnetz verarbeitet, das mit denjenigen Teilnetzen in Verbindung steht, in denen die im Input benachbarten Wörter analysiert werden. Die zwischen den benachbarten Teilnetzen über entsprechende Kontrollstrukturen bestehenden Verbindungen ermöglichen u.a. den Austausch von Numerus-, Kasus- und Genusinformationen, wobei gleichzeitig überprüft wird, ob die Informationen der kooperierenden Teilnetze einander widersprechen. Wenn sich durch diesen Informationsaustausch ein Widerspruch ergibt – ein Wort soll beispielsweise gleichzeitig Singular und Plural sein –, wird eine Konstituentengrenze zwischen den entsprechenden Wörtern angenommen und der Austausch von Informationen an dieser Stelle unterbrochen.

(3.20) [...] weil der Sekt, den Männer lieben, kalt sein muß.

Bei der Analyse von den Männer im Beispiel (3.20) versucht das System als erstes, die beiden Wörter zu einer einfachen Nominalphrase zusammenzufassen. Da den Männer aber in keinem Fall eine syntaktisch korrekte Nominalphrase darstellen kann, bricht das Kontrollsystem die Übertragung von Numerus-, Kasus- und Genusinformationen zwischen beiden Wörtern ab, und sie werden getrennt voneinander als zu unterschiedlichen Konstituenten gehörig weiterverarbeitet.

Ergibt sich jedoch ein Widerspruch an einer Stelle, an der keine Konstituentengrenze zulässig ist (z.B. benötigt der indefinite Artikel ein einen 'rechten Partner'), reagiert der Parser mit einer 'Fehlermeldung', indem ein entsprechender Kontrollknoten Aktivierung erhält. Eine resultierende Folgeaufgabe des Kontrollsystems ist es dann, gegebenenfalls einen anderen, beispielsweise einen diskontinuierlichen 'Partner' zu finden, und zu versuchen, den unpassenden Nachbarn im Sinne von Abschnitt

[24] Diese Implementierung von Präferenzen ist natürlich stark vereinfacht. Es ist aber problemlos möglich, die Bevorzugung einer Lesart von komplexeren Bedingungen abhängig zu machen.

3.5.3 als Einschub zu überbrücken:

(3.21) `ein grüne eh grüner Klotz`

In einer Überbrückungsreparatur wie (3.21) würde der Parser zunächst die Verarbeitungsergebnisse von `ein` und `grüne` kombinieren. Dieser Versuch führt jedoch zu einem Widerspruch und zu einer entsprechenden Fehlermeldung. Die dadurch eingeleitete syntaktische Reorganisation stellt schließlich die Verbindung zwischen `ein` und dem nächsten passenden Partner (`grüner`) her, und der Reparaturbezug (`grüne`) bzw. die Reparatureinleitung (`eh`) werden als Parenthese ohne Numerus-, Kasus- und Genusverbindungen zu ihren rechten bzw. linken Nachbarn isoliert. In gleicher Weise läßt sich dieses hier am Beispiel einer Reparatur beschriebene Überbrückungsverfahren auch auf die Verarbeitung syntaktisch korrekter, diskontinuierlicher Inputs übertragen.

3.6 Kohärenzprozesse im gesprochenen Deutsch

Sprachverarbeitung und Kohärenzprozesse laufen in sehr komplexen Systemen ab. Deshalb ist es forschungsstrategisch sinnvoll, bei der Untersuchung von Kohärenzphänomenen eine Negativmethodologie zu verwenden, d.h. Inkohärenzphänomene zu fokussieren und deren kommunikative Bearbeitung zu untersuchen. Diese Überlegung führte zur Auswahl der Reparaturen als Untersuchungsgegenstand. Reparaturen sind zudem aufgrund ihres häufigen Vorkommens in der gesprochenen Sprache für sich genommen ein linguistisch relevanter Gegenstand. Speziell ist erklärungswürdig, wie Inkohärenzen bzw. kommunikative Störungen mit ihnen beseitigt werden können. Zugleich bieten Reparaturen als sprachlich manifestierte Verfahren einen unmittelbaren empirischen Zugang zu Inkohärenzverursachung und Kohärenzherstellung.

Die an sprachlichen Objekten manifestierte Erscheinungsform von Inkohärenz in Reparaturen, Versprechern und anderen devianten Strukturen wurde in den vorstehend beschriebenen Modellen auf die Erscheinungsform der Prozesse zurückgeführt, die die Inkohärenz verursachen bzw. überwinden. Dazu wurde der verwendete Prozeß- und der Kohärenzbegriff jeweils expliziert und für die Modellierung der Entstehung von Inkohärenz bzw. der Überwindung von Inkohärenz bei Produktion und Rezeption empirisch verankert. Auf diese Weise wurde es möglich, Bedingungen für kohärente Sprachverarbeitung zu formulieren.

Der allen Modellen gemeinsame Ansatz geht von einer Unterteilung des Verarbeitungssystems in ein Objekt- und ein Metasystem aus. Das Auftreten und die Beseitigung einer Inkohärenz wird dabei als Phänomen der Interaktion zwischen beiden Systemen aufgefaßt: Gerät das Objektsystem in einen inkohärenten Zustand, so wird dies vom Metasystem erkannt, und unter seiner Kontrolle wird das Gesamtsystem schließlich wieder in einen kohärenten Zustand überführt. In den hier be-

schriebenen Modellen wurden unterschiedliche Formen der Interaktion und der Aufgabenverteilung zwischen Objekt- und Metasystem diskutiert und auf Adäquatheitskriterien hin evaluiert.

Die Untersuchungen mit diesen Modellen zeigen somit Eigenschaften des natürlichen Sprachverarbeitungssystems auf, die maßgeblich für die Entstehung und Überwindung von Inkohärenzen verantwortlich sind und bei einer Modellierung von Sprachverarbeitung berücksichtigt werden müssen. In gleicher Weise lassen sich aus der Analyse und Modellierung von Reparaturmechanismen generelle Aussagen über Konzeption und Fähigkeiten des Sprachverarbeitungssystems ziehen, die zu einer Erklärung des komplexen Gegenstandes der Kohärenz beitragen.

Literatur

Aho, A.V. & Ullman, J.D. (1972). *The Theory of Parsing, Translation and Compiling.* Englewood Cliffs, NJ: Prentice-Hall.

Aho, A.V. & Ullman, J.D. (1977). *Principles of Compiler Design.* Reading, MA: Addison Wesley.

Baars, B.J., Motley, M.T. & MacKay, D.G. (1975). Output editing for lexical status in artifically elicited slips of the tongue. *Journal of Verbal Language and Verbal Behaviour, 14,* 382–391.

Beattie, G. (1977). The dynamics of interruption and the filled pause. *British Journal of Clinical Psychology, 16,* 283–284.

Berg, T. (1986). The problems of language control: Editing, monitoring, and feedback. *Psychological Research, 48,* 133–144.

Berg, T. (1988). *Die Abbildung des Sprachproduktionsprozesses in einem Aktivierungsflußmodell.* Tübingen: Niemeyer.

Cole, R.A. & Jakimik, J. (1980). A Model of Speech Perception. In R.A. Cole (Ed.), *Perception and Production of Fluent Speech* (pp. 133–163). Hillsdale, NJ: Erlbaum.

Dell, G.S. (1985). Positive feedback in hierarchical connectionist models: Applications to language production. *Cognitive Science, 9,* 3–23.

Dell, G.S. (1986). A spreading-activation theory of retrieval in sentence production. *Psychological Review, 93,* 283–321.

Dell, G.S. (1988). The retrieval of phonological forms in production: Tests of prediction from a connectionist model. *Journal of Memory and Language, 27,* 124–142.

Dell, G.S. & Reich, P.A. (1980). Toward a unified model of slips of the tongue. In V.A. Fromkin (Ed.), *Errors in Linguistic Performance* (pp. 273–286). New York, NY: Academic Press.

Earley, J. (1986). An Efficient Context-Free Parsing Algorithm. In B.J. Grosz, K. Sparck Jones & B.L. Webber (Eds.), *Natural Language Processing* (pp. 25–35). Los Altos, CA: Morgan Kaufmann.

Eikmeyer, H.-J. (1987a). *Die Simulation der Produktion von "covert repairs".* Manuskript. Bielefeld: Universität Bielefeld.

Eikmeyer, H.-J. (1987b). *CheOPS: An Object-oriented Programming Environment in C-PROLOG.* KoLiBri Arbeitsberichte der Forschergruppe "Kohärenz", Nr. 4. Bielefeld: Universität Bielefeld.

Eikmeyer, H.-J. (1989). *Ein Prozeßmodell für die Produktion von "covert repairs".* Manuskript. Bielefeld: Universität Bielefeld.

Eikmeyer, H.-J., Kindt, W., Laubenstein, U., Polzin, Th., Rieser, H. & Schade, U. (1990). Reparaturen und Kohärenz in gesprochener Sprache. In Forschergruppe Kohärenz (Hrsg.), *Kohärenz. KoLiBri Arbeitsberichte der Forschergruppe "Kohärenz", Nr. 1.* (pp. 5–33). Bielefeld: Universität Bielefeld.

Eikmeyer, H.-J. & Schade, U. (1991). Sequentialization in connectionist language production models. Erscheint in: *Cognitive Systems.*

Forschergruppe Kohärenz (Ed., 1987). *"n Gebilde oder was" – Daten zum Diskurs über Modellwelten.* KoLiBri Arbeitsberichte der Forschergruppe "Kohärenz", Nr. 2. Bielefeld: Universität Bielefeld.

Grosz, B.J., Sparck Jones, K. & Webber, B.L. (Eds., 1986). *Natural Language Processing.* Los Altos, CA: Morgan Kaufmann.

Günther, U., Kindt, W., Schade, U., Sichelschmidt, L. & Strohner, H. (1991). *Elliptische Koordination. Einige Strukturen und Prozesse lokaler Textkohärenz.* KoLiBri Arbeitsberichte der Forschergruppe "Kohärenz", Nr. 32. Bielefeld: Universität Bielefeld.

Hoare, C.A.R. (1985). *Communicating Sequential Processes.* Englewood Cliffs, NJ: Prentice Hall.

Johnson-Laird, P.N. (1983). *Mental Models. Towards a cognitive science of language, inference, and consciousness.*. Cambridge, UK: Cambridge University Press.

Kay, M. (1986). Algorithm Schemata and Data Structures in Syntactic Processing. In B.J. Grosz, K. Sparck Jones & B.L. Webber (Eds.), *Natural Language Processing* (pp. 35–71). Los Altos, CA: Morgan Kaufmann.

Kindt, W. (1985). Grammatische Prinzipien sogenannter Ellipsen und ein neues Syntaxmodell. In R. Meyer-Hermann & H. Rieser (Eds.), *Ellipsen und fragmentarische Ausdrücke, Band 1* (pp. 161–290). Tübingen: Niemeyer.

Kindt, W. (1991a). *Grundzüge der Mehrdimensionalen Schaltgrammatik.* Erscheint in: KoLiBri Arbeitsberichte der Forschergruppe "Kohärenz", Nr. 21. Bielefeld: Universität Bielefeld.

Kindt, W. (1991b). *Informationsdynamik bei der grammatischen Verarbeitung.* Erscheint in: KoLiBri Arbeitsberichte der Forschergruppe "Kohärenz", Nr. 36. Bielefeld: Universität Bielefeld.

Kindt, W. & Laubenstein, U. (1990). *Reparaturen und Koordinationskonstruktionen. Ein Beitrag zur Strukturanalyse des gesprochenen Deutsch.* KoLiBri Arbeitsberichte der Forschergruppe "Kohärenz", Nr. 20. Bielefeld: Universität Bielefeld.

Kindt, W. & Weingarten, R. (1984). Verständigungsprobleme. *Deutsche Sprache, 12,* 193–218.

Klein, W. (1985). Ellipse, Fokusgliederung und thematischer Stand. In R. Meyer-Hermann & H. Rieser (Eds.), *Ellipsen und fragmentarische Ausdrücke, Band 1* (pp. 1–25). Tübingen: Niemeyer.

Laubenstein, U. (1990). *Ein konnektionistischer Parser für die Mehrdimensionale Schaltgrammatik.* Manuskript. Bielefeld: Universität Bielefeld.

Laver, J.D.M. (1980). Monitoring systems in the neurolinguistic control of speech production. In V.A. Fromkin (Ed.), *Errors in Linguistic Performance* (pp. 287–305). New York, NY: Academic Press.

Levelt, W.J.M. (1983). Monitoring and self-repair in speech. *Cognition, 14,* 41–104.

Levelt, W.J.M. (1989). *Speaking: From Intention to Articulation.* Cambridge, MA: MIT Press.

Lisken, S. & Rieser, H. (1990). *Ein inkrementeller Parser zur Analyse von simulierten Reparaturen.* KoLiBri Arbeitsberichte der Forschergruppe "Kohärenz", Nr. 29. Bielefeld: Universität Bielefeld.

MacKay, D.G. (1987). *The Organisation of Perception and Action.* New York, NY: Springer.

Motley, M.T., Baars, J.B. & Camden, C.T. (1981). Syntactic criteria in prearticulatory editing: Evidence from laboratory-induced slips of the tongue. *Journal of Psycholinguistic Research, 10,* 503–522.

Motley, M.T., Baars, J.B. & Camden, C.T. (1983). Experimental verbal slip studies: A review and an editing model of language encoding. *Communication Monographs, 50,* 79–101.

Motley, M.T., Camden, C.T. & Baars, J.B. (1979). Personality and situational influences upon verbal slips: A laboratory test of Freudian and prearticulatory editing hypotheses. *Human Communication Research, 4,* 195–202.

Motley, M.T., Camden, C.T. & Baars, J.B. (1981). Toward verifying the assumptions of laboratory induced slips of the tongue: The output-error and editing issues. *Human Communication Research, 8,* 3–15.

Motley, M.T., Camden, C.T. & Baars, J.B. (1982). Covert formulation and editing of anomalies in speech production: Evidence from experimentally elicited slips of the tongue. *Journal of Verbal Learning and Verbal Behavior, 21,* 578–594.

Rumelhart, D.E. & McClelland, J.L. (1986). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations.* Cambridge, MA: MIT Press.

Sacks, H., Schegloff, E.A. & Jefferson, G. (1974). A Simplest Systematics for the Organization of Turn Taking for Conversation. *Language, 50,* 696–735.

Schade, U. (1988). Ein konnektionistisches Modell für die Satzproduktion. In J. Kindermann & C. Lischka (Hrsg.), *Workshop Konnektionismus.* Arbeitspapiere der GMD, Nr. 329 (pp. 207–220). St. Augustin: Gesellschaft für Mathematik und Datenverarbeitung.

Schade, U. (1990a). *Konnektionistische Modelle der Sprachproduktion.* Dissertation. Bielefeld: Universität Bielefeld.

Schade, U. (1990b). Kohärenz und Monitor in konnektionistischen Sprachproduktionsmodellen. In G. Dorffner (Ed.), *Konnektionismus in Artifical Intelligence und Kognitionsforschung (KONNAI). Proceedings: 6. Österreichische Artificial-Intelligence-Tagung* (pp. 18–27). Berlin: Springer.

Schade, U. & Eikmeyer, H.-J. (1991). "wahrscheinlich sind meine Beispiele soo sprunghaft und und und eh ehm zu zu telegraph" – Konnektionistische Modellierung von "covert repairs". Zur Veröffentlichung eingereicht.

Schegloff, E.A. (1979). The relevance of repair to syntax-for-conversation. In T. Givón (Ed.), *Syntax and Semantics. Vol. 12* (pp. 261–286). New York, NY: Academic Press.

Schegloff, E.A., Jefferson, G. & Sacks, H. (1977). The preference for self-correction in the organization of repair in conversation. *Language, 2,* 361–382.

Siegman, A. (1979). Cognition and hesitation in speech. In A. Siegman & S. Feldstein (Eds.), *Of Speech and Time. Temporal Speech Patterns in Interpersonal Contexts* (pp. 151–178). Hillsdale, NJ: Erlbaum.

Stemberger, J.P. (1985a). An interactive activation model of language production. In A.W. Ellis (Ed.), *Progress in the Psychology of Language. Vol. 1* (pp. 143–186). Hillsdale, NJ: Erlbaum.

Stemberger, J.P. (1985b). *The Lexicon in a Model of Language Production.* New York, NY: Garland Publishing.

Stemberger, J.P. (1990). Wordshape errors in language production. *Cognition, 35,* 123–157.

Tomita, M. (1987). An Efficient Augmented-Context-Free Parsing Algorithm. *Computational Linguistics, 13,* 31–47.