Generierung natürlichsprachlicher Sätze in unifikationsbasierten Grammatiken

Ein konnektionistischer Ansatz

Diplomarbeit
Technische Universität München

Aufgabensteller:	*Prof. W. Brauer*
Betreuer:	*E. Klöck*
Bearbeiter:	*Andreas Stolcke*
	Ernst-Platz-Str. 17
	8000 München 50
	Tel. 0 89/1 49 16 81
Abgabe:	*15. 6. 1988*

1 Einleitung

1.1 Sprachverarbeitung und Generierung

Die Verarbeitung natürlicher Sprache (kurz: *Sprachverarbeitung*) ist traditionell eines der bedeutendsten Gebiete innerhalb der Künstlichen Intelligenz (KI). In den ersten Arbeiten auf diesem Gebiet wurde Sprachverarbeitung weitgehend gleichgesetzt mit *Sprachanalyse* bzw. *-erkennung*. Das Problem, natürlichsprachliche Eingaben des Benutzers zu entschlüsseln und in eine geeignete interne Form zu überführen, hatte eindeutig Vorrang vor der *Sprachgenerierung*, d. h. der Erzeugung von Ausgaben in natürlichsprachlicher Form.[1]

Dies erklärt sich vielleicht dadurch, daß eine triviale Form von Sprachgenerierung, der sog. *canned text*, seit jeher in Programmen verwendet wird, d. h. der Text liegt abgespeichert vor und wird zu einem vom Programmierer und vom Programmablauf bestimmten Zeitpunkt ausgegeben. Die Generierung besteht hier also vor allem aus Reproduktion. Diese Methode beruht im wesentlichen darauf, daß die Eingabe zwar durch den Benutzer bestimmt ist, bei der Ausgabe aber das Programm die Menge der möglichen Äußerungen bestimmen kann.

Eine Weiterentwicklung der *canned text*-Methode besteht darin, die Ausgabe aus Bausteinen zusammenzusetzen oder durch Textersetzung und -umstellung aus der Eingabe zu gewinnen. Auch für diese Verfahren gilt aber, daß sie Sprachgenerierung als eine Manipulation von Zeichenketten ohne linguistische Motivation betrachten. Dementsprechend beruht der Eindruck der "Natürlichkeit" solcher Ausgaben vor allem auf der Bereitschaft des Benutzers, auf die Suggestion eines intelligenten Sprechers einzugehen. Ein Paradebeispiel, in dem dieser Ansatz *ad absurdum* geführt wird, ist das Programm ELIZA [Weizenbaum1966a].

Selbst spätere Programme, die als Meilensteine der KI im allgemeinen und der Sprachverarbeitung im besonderen angesehen werden, betrachten die Spracherkennung als das Hauptproblem und handeln die sprachliche Ausgabe mit vergleichsweise einfachen Mitteln ab.

[1] Der Terminus *Sprache* in Verbindung mit Analyse, Generierung usw. wird hier in der Regel auf *natürliche* Sprache bezogen. Die allgemeinere Bedeutung von *Sprache* z. B. im Zusammenhang mit Übersetzerbau, Programmgenerierung usw. ist hier und im folgenden in der Regel *nicht* gemeint.

Als Beispiel kann Winograds SHRDLU [Winograd1972a] angesehen werden.

Auch mag die Tradition des Übersetzerbaus zu der anfänglich stark unterbewerteten Position der Generierung innerhalb der Sprachverarbeitung beigetragen haben. Aus der Sicht der Informatik ist es verlockend, die gesamte Sprachverarbeitung als ein Compilationsproblem zu betrachten. Dementsprechend wird dem Parsing ein sehr hohes Gewicht beigemessen und dabei übersehen, daß die "Codegenerierung" nur ein sehr unzulängliches Analogon zur Generierung natürlicher Sprache (im Gegensatz zu formalen Sprachen) darstellt.

Während des letzten Jahrzehnts haben sich jedoch die Arbeiten auf dem Gebiet der Sprachgenerierung erheblich intensiviert. Ein wesentlicher Faktor dabei war die Erkenntnis, daß die Erzeugung natürlichsprachlicher Ausgaben ein wesentliches Qualitätsmerkmal für Mensch-Maschine-Schnittstellen ist. Dies wird insbesondere in Verbindung mit teilweise schon kommerziell relevanten Produkten der KI wie Expertensystemen und Systemen zur computergestützten Unterweisung (CAI) deutlich.

Daneben gibt es ein verstärktes Interesse im Rahmen der kognitiven Psychologie, Modelle für das menschliche Sprachverhalten zu entwickeln. Die Generierung von sprachlichen Äußerungen bietet sich hier als ein Gebiet an, das relativ zugänglich für empirische Untersuchungen und Rückschlüsse auf zugrundeliegende Mechanismen ist.

1.2 Diese Arbeit

Ein erstes Ziel dieser Arbeit ist es, einen Überblick über die vielschichtigen Probleme der Sprachgenerierung zu geben. Dies geschieht im ersten Teil des Forschungsberichts in Abschnitt 2. Es folgt eine (notwendigerweise unvollständige) Übersicht über wichtige Arbeiten auf dem Gebiet der Sprachgenerierung, wobei sowohl die kognitionswissenschaftliche als auch die anwendungsorientierte Forschungsrichtung berücksichtigt wird.

Es wird sich zeigen, daß sehr unterschiedliche kognitive Modelle und Formalismen in den verschiedenenen Ansätzen Verwendung finden. Eines der kognitiven Paradigmen aus jüngerer Zeit, der sog. *Konnektionismus*, ist dabei vor allem bei bei der Modellierung psycholinguistischer Phänomene von Bedeutung.

Aus der theoretischen Linguistik dagegen kommt ein Formalismus zur abstrakten Sprachbeschreibung, der in verschiedenen Varianten eine zentrale Rolle in vielen Generierungssystemen spielt: die *unifikationsbasierten Grammatiken.* Dieser Formalismus wird in Abschnitt 3 vorgestellt.

Konnektionismus und unifikationsbasierte Grammatiken sind in vielerlei Hinsicht prototypisch für zwei grundverschiedene Verarbeitungsparadigmen innerhalb der KI und der Informatik, die beide sehr spezifische Vorzüge aufzuweisen haben. In Abschnitt 4 wird deshalb untersucht, inwieweit unifikationsbasierte Grammatiken mit dem konnektionistischen Paradigma vereinbar sind.

Danksagung

Ich möchte an dieser Stelle Erwin Klöck für die engagierte Betreuung der Arbeit danken. Ihm und allen übrigen Mitgliedern der Arbeitsgruppe um Christian Freksa danke ich für Unterstützung und wertvolle Diskussionen. Besonders verpflichtet bin ich Kai Zimmermann, der seinen Netzsimulator parallel zu dieser Arbeit entwickelt hat und dabei stets offen für immer neue Sonderwünsche meinerseits war.

2 Forschungsbericht

Der erste Teil dieses Kapitels stellt eine Einführung in die Problematik der Generierung dar, verbunden mit dem Versuch, aufgrund der zugrundeliegenden Konzepte eine grobe Typologie für Generierungssysteme zu erstellen.

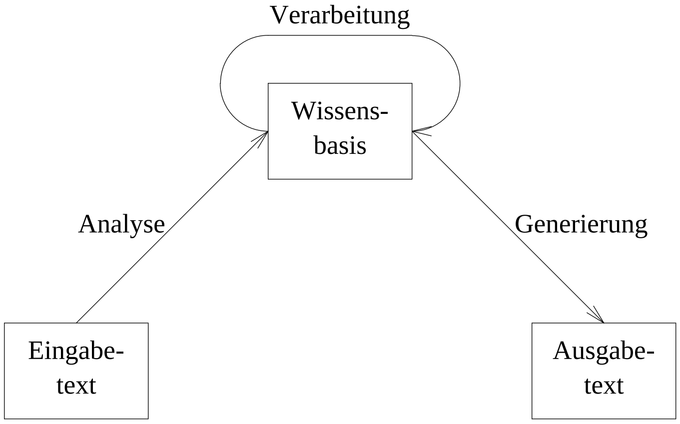
Damit wird ein zweifaches Ziel verfolgt: Mit dem hier geschaffenen theoretischen und konzeptuellen Hintergrund wird es in Abschnitt 2.3 möglich sein, eine vergleichende Betrachtung verschiedener existierender Systeme vorzunehmen, und schließlich kann auch die eigene Arbeit vor diesem Hintergrund eingeordnet werden.

2.1 Das Generierungsproblem

2.1.1 Zwei Paradigmen der Sprachverarbeitung

Eine Möglichkeit, das Problem zu sehen, ist die Generierung als Transformation von Datenrepräsentationen. Ausgehend von einer Wissensbasis, die die mitzuteilende Information in einer Form enthält, die aus vorherigen Verarbeitungsschritten resultiert, gilt es, diese Information in die (oder eine) natürlichsprachliche Form zu überführen. Im übrigen läßt sich die Analyse von natürlichsprachlichen Eingaben auf ganz analoge Weise charakterisieren, so daß sich als erstes Paradigma für ein sprachverarbeitendes System etwa folgendes Bild ergibt:

(1) *Paradigma I der Sprachverarbeitung*



Diese Art der Problembeschreibung liegt der Informatik sehr nahe; sie entspricht der Art und Weise, wie Aufgaben in der Theorie der Formalen Sprachen oder im Übersetzerbau charakterisiert werden.

Eine anderes Paradigma der Sprachverarbeitung ist motiviert durch Theorien der Psychologie und der linguistischen Pragmatik. Hierbei wird Sprachverarbeitung nicht als Transformation von Daten, sondern als Teil der Interaktion des Individuums mit seiner Umwelt aufgefaßt. Diese Interaktion ist bestimmt einerseits durch Wahrnehmung der Umwelt und andererseits durch eigenes Handeln in dieser Umwelt. Obwohl dabei die physische Interaktion (Wahrnehmung optischer und akustischer Reize, Handeln durch manuelle Manipulation u. ä.) am offensichtlichsten ist, kann man auch sprachliche Kommunikation in dieses Schema einordnen. In der *Sprechakttheorie* [Searle1969a] geschieht dies dadurch, daß man jede sprachliche Äußerung als einen sog. *Sprechakt* deutet, d. h. als sprachliche Handlung. Alltägliche Sprechakttypen sind z. B. Behauptung, Versprechen, Drohung, Frage u. dgl. Sprechakte sind durch (a) ihre Äußerungsbedingungen und (b) ihre Wirkung auf das Verhältnis zwischen Sprecher und Hörer charakterisiert. So beinhaltet die *Drohung*

(2) *Paß' auf, daß ich dich nicht erwische!*

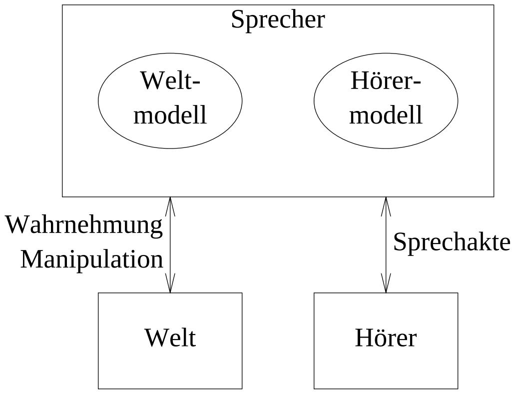
(a) die Vorbedingung, daß der Sprecher in der Lage ist, den Hörer "zu erwischen", und dies für den Hörer nicht erwünscht ist (andernfalls wäre dies ein *Versprechen* und keine Drohung); und (b) die Versicherung des Sprechers an den Hörer, daß jener tatsächlich die Absicht hat, diesen zu erwischen.[2]

Um Sprechakte und damit zusammenhängende Phänomene behandeln zu können, benötigt man in der Regel ein *Modell* der Welt und der in ihr agierenden Sprecher und Hörer. Solche Modelle verlangen selbst für einfache Domänen einen enormen Repräsentationsaufwand, denn es langt nicht, den aktuellen "realen" Weltausschnitt zu modellieren. Um nämlich auch das subjektive Glauben, Wissen und Wünschen (die sog. *propositionalen Einstellungen*) der Sprecher – das ihr Handeln bestimmt – erfassen zu können, müssen auch hypothetische *mögliche Welten*

[2] Die genaue Definition, was einen bestimmten Sprechakt ausmacht, verlangt noch eine Reihe weiterer Bedingungen, auf die hier der Einfachheit halber nicht eingegangen wird.

repräsentiert werden:

(3) *Paradigma II der Sprachverarbeitung*



Neben dem reinen quantitativen Aufwand taucht weiterhin das Problem auf, daß es bisher noch nicht gelungen ist, die propositionale Einstellungen auf voll befriedigende Weise theoretisch zu erfassen und zu formalisieren. Trotzdem gibt es Sprachgenerierungssysteme, die diesen Aufwand nicht scheuen und so in der Lage sind, ihre Äußerungen auf der Basis von Sprechakten zu planen [Appelt1985a].

Die beiden vorgestellten Sprachverarbeitungskonzepte stehen natürlich nicht isoliert voneinander da, sondern bei genauerer Analyse läßt sich das Paradigma II als das allgemeinere identifizieren. Das Paradigma I dagegen wird nur einem Aspekt der natürlichsprachlichen Kommunikation gerecht – der Transformation der sprachlichen Botschaft über verschiedene Formen der Repräsentation – ignoriert aber wichtige Voraussetzungen der Kommunikation, nämlich vor allem ein umfassendes Welt-/Hörermodell.

Für gewisse standardisierte Kommunikationssituationen kann es durchaus sinnvoll sein, auf das einfachere Konzept zurückzugreifen, z. B. dann, wenn das Spektrum der Sprechakttypen von vorneherein stark eingeschränkt ist. Typisch wäre hier ein Datenbankauskunftsystem, wobei immer wieder die Interaktion “Informationsgesuch des Benutzers – Informationsausgabe durch das System” durchlaufen wird.

Ähnliches trifft für alle Fälle zu, wo die Leistung der Generierung im wesentlichen aus der Erteilung von Auskunft besteht. Allerdings sollte man hier vorsichtig sein: Auch in solch “einfachen” Fällen bedarf es eines rudimentären Hörermodells, um z. B. festzulegen, in welchem Umfang oder in welcher Genauigkeit die Auskunft sinnvoll ist.

2.1.2 Was? und Wie?

Die Aufgabe der Generierung wird unabhängig vom verwendeten Paradigma (das oft nur implizit eine Rolle spielt) in der Regel durch zwei Teilprobleme strukturiert:

(i) *Was* soll gesagt werden? und

(ii) *Wie* soll es gesagt werden?

Die Frage (i) kann so interpretiert werden, daß in einem ersten Schritt die Auswahl der Daten aus der Wissenbasis getroffen werden muß, die es mitzuteilen gilt. (ii) zielt dann darauf ab, daß diese Daten anschließend natürlichsprachlich zu formulieren sind, wobei einerseits die Regeln der Ausgabesprache zu beachten sind, andererseits u. U. aus mehren möglichen Formulierungen die "beste", d. h. für die Situation adäquateste zu finden ist.

Die Dichotomie, die durch (i) und (ii) definiert ist, taucht in der Literatur in den verschiedensten Terminologien auf. Neben *What to say?* versus *How to say it?* werden z. B. in [McKeown1985a] die Begriffe *strategische* bzw. *taktische Komponente* der Generierung verwendet, Bezeichnungen, die m. E. nicht sehr glücklich gewählt sind. In [Hovy1985a] wird eine ähnliche Unterscheidung anhand der Begriffe *Sprachplanung* bzw. *-produktion* getroffen.

Natürlich werden diese Begriffspaare nicht voll synonym verwendet, bei allen liegt jedoch die Idee der Unterscheidung zwischen Form und Inhalt zugrunde.[3]

Einer der Hintergedanken bei dieser Unterscheidung ist meist eine Modularisierung im Hinblick auf Sprachabhängigkeit bzw. -unabhängigkeit. Vgl. folgendes Zitat aus [Hovy1985a]:

> [...] *it is necessary to have the planning and the production interact* [...] *in such a way that the former need not contain explicit syntactic knowledge, and that the latter need not contain explicit goal-related information.*

Man kann den Zusammenhang auch umkehren und definieren: Die inhaltsbestimmende Komponente hat den Teil der Generierung zu umfassen, der sprachunabhängig operiert. Die formbestimmende Komponente umfaßt entsprechend den sprachspezifischen Teil der Generierung.

Trotz dieser sicherlich aus konzeptuellen und praktischen Gründen berechtigten Unterscheidung zwischen Inhaltlichem und Formalem bzw. zwischen Sprachunabhängigem und -abhängigem ist diese einfache Dichotomie doch in vielerlei Hinsicht unzureichend. Dies wird u. a. dadurch deutlich, daß gelegentlich noch weitere Punkte zu (i) und (ii) hinzugenommen werden, z. B.

(iii) *Wann* ist etwas zu sagen? [McKeown1985b]

Dabei wird auf das Problem der Sequentialisierung von Inhalten innerhalb längerer Texte abgehoben. (iii) läßt sich durchaus unter (i) und (ii) subsumieren, jedoch nicht eindeutig: Insofern entschieden wird, *was als nächstes* zu sagen ist (*What to say next?*), fällt dies unter (i); betrachtet man jedoch einen längeren Textabschnitt als Einheit, so gehört die Frage der möglichen Sequentialisierungen zu (ii).

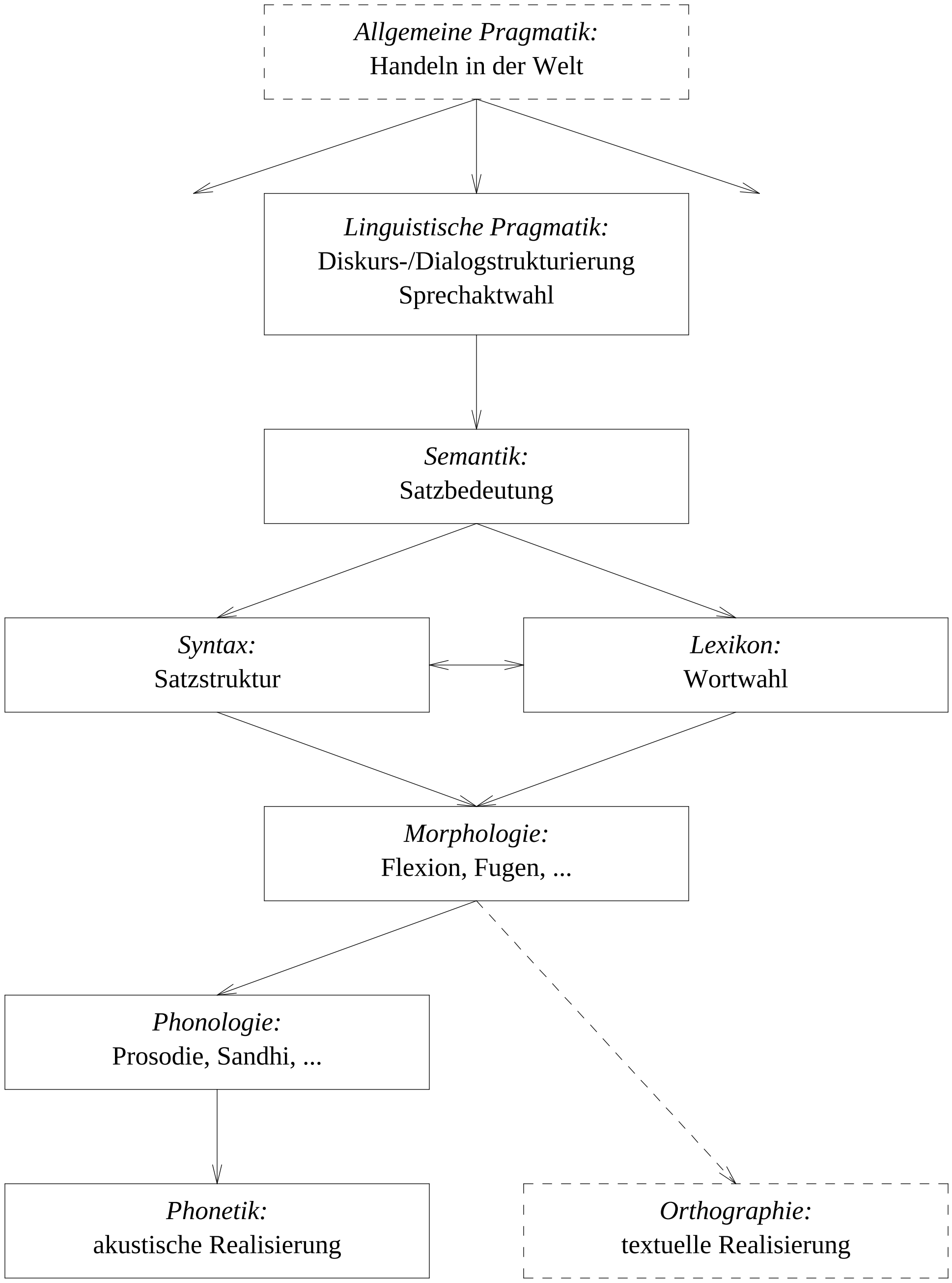
Die obige Diskussion zeigt, daß es nicht langt, das Generierungsproblem in lediglich zwei Abschnitte aufzugliedern. Geht man von einer Trennung in eine inhalts- und eine formbestimmende Generierungskomponente aus, die in einer zweistufigen, hierarchischen Abhängigkeit stehen, so ist es naheliegend, diese Hierarchie durch eine weitere Abstufung zu verfeinern. Dieses Vorgehen hat seine Entsprechung in der Tatsache, daß man auch sprachliche Einheiten selbst als in vielerlei Hinsicht hierarchisch strukturiert auffassen kann. Dies wird bereits von praktisch allen existierenden Generierungssystemen ausgenutzt, und zwar auf den verschiedensten Ebenen.

[3] Die Termini *Planung* und *Produktion* werden später nach einer genaueren Analyse eine zweite Dichotomie beschreiben, die bis zu einem gewissen Grad orthogonal zu der hier beschriebenen zwischen Form und Inhalt liegt.

So kann man z. B auf der Ebene von Textabschnitten (Artikel, Absätze usw.) diese als rekursiv aus Unterabschnitten aufgebaut betrachten [McKeown1983a]. Anderseits sind Texte auf der Ebene von einzelnen Sätzen entsprechend den gängigen linguistischen Theorien ebenfalls rekursiv in Nebensätze, Phrasen usw. strukturiert.

Tatsächlich gibt die Linguistik selbst in ihrer üblichen Gliederung ein Schema für die hierarchische Gliederung der Generierung ab. Diese Hierarchie ist in Abb. (4) wiedergegeben.

(4)



Linguistische Gliederung der Generierung

Die Darstellung legt es nahe, Generierung als einen Prozeß der schrittweisen Verfeinerung innerhalb dieser Hierarchie zu realisieren (vgl. [McDonald1987a]). Die Fragen des *Was* bzw.

des *Wie* sind dabei in der Hierarchie nicht eindeutig lokalisiert. Vielmehr kann man sie sich als über mehrere Ebenen “verschmiert” denken, wobei allerdings spezifische Schwerpunkte anzunehmen sind. So wird das *Was* hauptsächlich im Bereich der Pragmatik entschieden, während das *Wie* prototypisch (aber eben nicht ausschließlich) durch Syntax und Lexikon realisiert wird.[4] Allerdings beschränken sich die meisten Systeme auf einen mehr oder weniger großen Ausschnitt der Generierungshierarchie, so daß sich dann innerhalb dieses Ausschnitts guten Gewissens ein Teil als “strategische” und ein anderer als “taktische” Komponente identifizieren läßt. Auch diese Vereinfachung basiert im wesentlichen auf einer der Anwendung entsprechenden Einschränkung der pragmatischen Anforderungen an das Generierungssystem.

2.1.3 Die Schnittstelle

Selbst in relativ einfachen, zweischichtigen Systemen ergibt sich die Frage, wie die Schnittstelle zwischen der *Was*-Komponente und der *Wie*-Komponente aussehen soll. Bei Systemen, die konzeptuell mehrere Ebenen unterscheiden, wird diese Frage entsprechend für alle Schnittstellen zu entscheiden sein. Wir wollen sie jedoch hier an dem einfacheren Fall diskutieren.

Traditionell laufen auf Hierarchien operierende Verfeinerungsprozesse, wie sie in der Informatik gang und gäbe sind, *top-down* ab, d.h. jede Ebene erstellt eine Spezifikation für die jeweils nächste Stufe der Verfeinerung, so daß sie sich um deren Realisierung nach dem Prinzip des *information hiding* nicht zu kümmern braucht. Gerade dieses Prinzip wird aber im Zusammenhang mit der oben skizzierten linguistischen Hierarchie in Frage gestellt, und zwar in mehrfacher Hinsicht.

Empirische Untersuchungen natürlicher Sprachproduktion zeigen, daß unter gewissen Umständen eine Rückkoppelung zwischen dem formalen und dem inhaltlichen Teil der Generierung stattfinden muß, damit erfolgreich kommuniziert werden kann. Dies hängt damit zusammen, daß es nicht immer gewährleistet ist, daß das, was inhaltlich spezifiziert wurde, auch auf Anhieb formal korrekt realisiert werden kann. Grund dafür können Schwierigkeiten bei der Wortwahl, zu komplexe und deshalb nicht mehr überschaubare Satzkonstruktionen oder sogar eine Neubestimmung des Satzinhalts noch während der Formulierung sein. Dies manifestiert sich in gesprochener Sprache durch sog. *Selbstkorrekturen*, in denen der Sprecher seine Rede unterbricht und gewisse Teile wiederholt oder aber versucht, auf möglichst elegante Weise an das bereits Geäußerte anzuknüpfen (vgl. [Stolcke1986a]). In allen diesen Fällen muß die Frage über das weitere *Was* neu entschieden werden, wobei aber genaue Informationen über das bisherige *Wie* benötigt werden. Will der Sprecher die Situation “retten”, so ist in solchen Fällen der normale Informationsfluß umzukehren: Wo normalerweise das *Was* das *Wie* spezifiziert, legt jetzt das *Wie* der Realisierung fest, *was* als nächstes zu sagen ist.

Der zweite Punkt, der gegen eine strenge *top-down*-Kontrolle der Generierung spricht, ist ebenfalls durch psycholinguistische Erkenntnisse motiviert und hängt mit dem ersten Punkt zusammen. Es spricht vieles dafür, insbesondere auch neurophysiologische Fakten, daß die

[4] An dieser Stelle ist anzumerken, daß die letzte Instanz der Formbestimmung, nämlich die phonologisch-phonetische Komponente, in fast allen gängigen Systemen dadurch ausgeklammert wird, daß eine textuelle Ausgabe erfolgt, wobei deren Realisierung trivial gehalten ist insofern, als lediglich Oberflächenvollformen reproduziert werden. Auch eine spezielle morphologische Komponente wird auf diese Weise ausgespart.

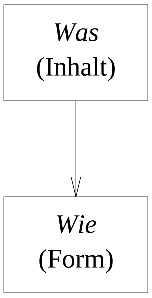
Leistungsfähigkeit menschlicher Sprachverarbeitung (und Informationsverarbeitung allgemein) auf einem hohen Grad an Parallelität beruht. Was nun die Schnittstelle zwischen "strategischer" und "taktischer" Komponente angeht, so würde es die Parallelität erheblich einschränken, wenn erst jeweils eine vollständige Spezifikation des Äußerungsinhalts erstellt werden müßte, bevor mit der Realisierung dieses Inhalts begonen werden kann. Plausibler ist daher eine parallele Aktivität beider Komponenten, wobei auch unvollständige Sprezifikationen realisiert werden. Das heißt aber, daß für die weitere Inhaltsbestimmung – ähnlich wie bei den oben erwähnten Korrekturen – eine Kontrolle über die bereits geäußerte Form notwendig ist, um kohärent fortfahren zu können, oder aber, daß die realisierende Komponente explizite Anforderungen für fehlende Teile der Spezifikation stellt. In jedem Fall muß also eine *Interaktion* beider Komponenten stattfinden.

Die Charakteristik der Schnittstelle zwischen inhaltsbestimmender und formbestimmender Komponente bei der Generierung erlaubt eine grobe Klassifizierung in drei grundlegende Typen:[5]

(a) *Integriert*:

Dieser Fall ist dann gegeben, wenn keine klare Unterscheidung der beiden Komponenten getroffen und die Schnittstelle daher eigentlich gar nicht existiert bzw. nur implizit im Programmcode des Systems versteckt ist. Ein System, das zumindest tendenziell diese Eigenschaft hat, ist KAMP [Appelt1985a].

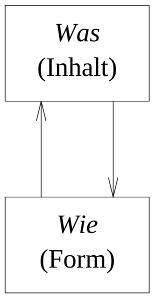
(b) *Sequentiell/unidirektional*:



Dieser Fall liegt dann vor, wenn die Entscheidung über das *Wie* der Äußerung nach vollständiger Spezifikation des Inhalts autonom getroffen wird. Beispiel hierfür ist die Schnittstelle zwischen strategischer und taktischer Komponente im Generierungssystem TEXT [McKeown1985a].

5 Die Idee einer solchen Klassifizierung und einige der hierfür verwendeten Termini gehen auf einen Vortrag von Norbert Reithinger zurück, den dieser im Rahmen des Münchner KI-Arbeitskreises am 25. 1. 88 hielt.

(c) *Interaktiv/bidirektional*:



Hier kooperieren beide Komponenten miteinander insofern, als einerseits das *Was* nur unvollständig spezifiziert wird und zusätzliche Informationen erst dann bereitgestellt werden, wenn sie explizit von der Realisierungskomponente angefordert werden. Außerdem kann sich die Inhaltsplanung auf die Form der Realisierung beziehen, um eigene Entscheidungen zu treffen. Dieses Prinzip wird in Hovys PAULINE realisiert [Hovy1984a].

Schnittstellen, bei denen die untere Komponente lediglich mit der oberen interagiert, um bei Bedarf Informationen anzufordern, aber der oberen Komponente selbst keine Informationen liefert, die diese in ihrer eigenen Operation beeinflussen würden, sollen als sog. *pseudo-interaktive* Schnittstellen bezeichnet werden. Dagegen soll für den Fall, daß eine gegenseitige Beeinflussung möglich ist, auch von *echter Interaktion* die Rede sein.

Es sei hier noch einmal betont, daß diese Klassifizierung der Schnittstellen allgemeiner Art sind, da sie überall dort sinnvoll ist, wo in einer Hierarchie zwei aneinander grenzende Komponenten kommunizieren.

2.1.4 Planung und Produktion

Wenn man die Sprachgenerierung als ein Problem der schrittweisen Verfeinerung von Spezifikationen auffaßt, wie es in den meisten Arbeiten geschieht, so liegt nahe, hierfür Techniken anzuwenden, die in der KI allgemein unter dem Begriff *Planung* zusammengefaßt werden. Darunter sind Verfahren zu verstehen, die in einem sog. *Problemraum* einen gegebenen Ausgangszustand in einen gewünschten Endzustand überführen und zwar unter möglichst optimaler Verwendung von Operatoren und Resourcen.[6]

Da dies ein sehr allgemeiner Ansatz ist und darüberhinaus die Idee von “Generierung als Planung mit sprachlichen Mitteln” intuitiv einigermaßen einleuchtend ist, findet das Prinzip verbreitete Anwendung, und zwar in der Form des *hierarchischen Planens*. Dieses beruht auf der Idee, daß man die normalerweise auftretende kombinatorische Explosion bei der Suche nach möglichen Operationen auf dem Problemraum dadurch einschränken kann, daß man das Problem nicht auf einer Ebene zu lösen versucht, sondern auf mehreren gestaffelten Abstraktionsebenen.

[6] Eine gute zusammenfassende Darstellung des Planungsproblems und verschiedener Lösungsstrategien findet sich in [Sacerdoti1977a].

Man versucht also, das Problem in zunächst abstrakter (d. h. auch: ungenauer) Form zu lösen und diese Lösung dann zu verfeinern, indem ihre einzelnen Schritte selbst wiederum als Instanzen des Planungsproblems auf der nächst niedrigeren Abstraktionsebene behandelt werden.

Hierarchische Planungsverfahren scheinen also dem Generierungsproblem besonders angepaßt zu sein, insbesondere wenn man die oben skizzierte, durch die Linguistik vorgegebene Problemstruktur zugrunde legt. Es gibt nun die unterschiedlichsten Möglichkeiten, hierarchisches Planen zu realisieren, jedoch hat ein System im Rahmen der Sprachgenerierung eine besondere Popularität erlangt: der Planer NOAH von Sacerdoti [Sacerdoti1977a]. NOAH bietet neben seiner hierarchischen, auf Problemabstraktion basierenden Methode noch zwei Besonderheiten, die ihn besonders leistungsfähig machen. Dies ist einerseits die Fähigkeit, *nichtlineare* Pläne zu generieren, d. h. Pläne, deren Einzelschritte nur partiell zeitlich geordnet sind. Die vollständige Linearisierung der Planschritte wird solange aufgeschoben, bis diese sich aus Randbedingungen oder aus Optimierungserwägungen ergibt. Die zweite Besonderheit stellen die sog. *Plankritiker* dar, die einen Plan, nachdem er durch lokale Expansion der Teilschritte des übergeordneten, abstrakteren Plans entstanden ist, einer globalen Untersuchung und evtl. Optimierung unterzieht. Durch diese Arbeitsteilung ist das Planungswissen modularisiert in lokales und globales Wissen, was die Anwendung des Planers auf neue Domänen enorm erleichtert (Sacerdoti benutzt ursprünglich eine Blockweltdomäne).

Da das Prinzip des Planens sehr allgemein ist, ließen sich ohne weiteres sämtliche Ebenen des Generierungsprozesses in einen einzigen, sehr umfangreichen Planungsprozeß integrieren. Dies hätte unbestreitbare konzeptuelle und praktische Vorteile: Da alle Ebenen nach dem selben, einheitlichen Prinzip verfahren, würden einige Aspekte des Schnittstellenproblems entfallen. Zum Beispiel wäre der hierarchische Planer ohne weiteres in der Lage, bei Schwierigkeiten in der Realisierung einer Ebene automatisch *Backtracking* einzuleiten und das Problem auf einer abstrakteren Ebene neu zu planen. Da das Verfahren so allgemein ist, könnte die gesamte Generierung in die Domänenplanung des Systems integriert werden. Das sprachliche Handeln wäre auf natürliche Weise ein Ausschnitt der globalen Aktionen des Systems in seiner Problemdomäne.

Ein noch eher utopischer Aspekt ergibt sich für den Fall, daß es gelänge, (a) die gesamte Kommunikation als Ergebnis eines Planungsprozesses zu erklären und (b) ein allgemeines Verfahren zur Rekonstruktion von Plänen zu entwickeln. Dann könnte das *Verstehen* von kommunikativen Äußerungen als die *Erkennung* (Rekonstruktion) von Kommunikationsplänen realisiert werden. Da keine der beiden Voraussetzungen bisher auch nur annähernd erfüllt sind (und es äußerst ungewiß ist, ob sie erfüllt werden können), ist dies natürlich nur Zukunftsmusik. Auf spezielleren Domänen gibt es jedoch schon erfolgreiche Bemühungen, Planerkennung als Erklärung und Mittel von Verständnisprozessen zu benutzen (vgl. [Wilensky1983a]).

Die vollständige Integration der Generierung in den Planungsprozeß hat jedoch auch Nachteile. Trifft man keine besonderen Vorkehrungen, so wird bei diesem Vorgehen das sprachliche Wissen nur sehr ungenügend modularisiert, denn die Gesamtheit der Planungsoperatoren, seien sie nun pragmatischer, semantischer oder syntaktischer Art, erscheint als unstrukturierte Menge. Außerdem ist das Wissen in den Planungsoperatoren in prozeduraler Form kodiert und daher schlecht zu handhaben. Ein Planungsoperator gibt in prozeduraler Form an, mit welchen Mitteln ein bestimmtes Planungsziel unter gewissen Vorbedingungen zu realisieren ist. Für bestimmte Bereiche der Generierung erscheint es jedoch natürlicher, linguistisches Wissen in deklarativer Form zu repräsentieren und von einem speziellen Mechanismus *interpretieren* zu lassen. Typisches Beispiel hierfür ist die Syntax natürlicher Sprachen, die üblicherweise mit Hilfe eines (deklarativen) Grammatikformalismus beschrieben wird. Die Generierung eines Satzes erfolgt dann dadurch, daß eine Spezifikation des Satzinhalts in Verbindung mit der zu verwendenden Grammatik von einem Interpreter in die entprechende syntaktische Struktur überführt wird. Dieses Verfahren soll in Anlehnung an die Terminologie von [Hovy1985a] als *Produktion* bezeichnet werden.[7]

Dieser Prozeß ließe sich auch durch eine Menge von Planungsoperatoren realisieren, in denen das Wissen über die möglichen syntaktischen Strukturen der Sprache implizit enthalten ist. Die Darstellung wäre jedoch weitaus weniger transparent und könnte sich nicht unmittelbar auf etablierte und ausgearbeitete linguistische Formalismen stützen.

Zusammenfassend lassen sich also zwei verschiedene Methoden charakterisieren, die im Rahmen der Generierung Anwendung finden: *Planung* als die Anwendung prozeduralen Wissens orientiert an den Gegebenheiten eines Problems und *Produktion* als die Interpretation deklarativen Wissens zur Erzeugung einer Ausgabe aus einer Eingabespezifikation. Es muß allerdings festgehalten werden, daß die Unterscheidung zwischen Planung und Produktion fließend sein kann, indem eine Reihe von Mischformen denkbar sind.[8]

Typischerweise werden Produktionsverfahren im Bereich der “unteren” Ebenen des Generierungsprozesses eingesetzt, also z. B. in der Syntax und Morphologie. Planungsverfahren dagegen sind gegenwärtig das Mittel der Wahl, wenn es um die Bestimmung von Diskursstruktur, Sprechakten u. ä. geht. Das liegt auch daran, daß Produktionsverfahren einen hohen Grad von theoretischer und formaler Durchdringung des Problems erfordern, während sich im Rahmen von Planungsoperatoren leichter *ad hoc*-Lösungen realisieren lassen.

Die empirisch zu beobachtende Häufigkeit der Anwendung beider Verfahren könnte dazu führen, daß man die Unterscheidung zwischen Planung und Produktion gleichsetzt mit der in Abschnitt 2.1.2 diskutierten Dichotomie zwischen Inhalt- und Formbestimmung in der Sprachgenerierung. Streng genommen sind beide Unterscheidungen jedoch unabhängig voneinander. So ist es z. B. durchaus denkbar, daß Inhalte durch einen deklarativen Formalismus generiert werden, wie dies ansatzweise mit McKeowns Textschemata [McKeown1983a] geschieht.

Auch die Ausgliederung von bestimmten Teilen der Generierung aus dem Planungsmechanismus und deren Realisierung mittels Produktion ist ein Weg, um sprachabhängiges Wissen von sprachunabhängigem zu trennen und damit zur besseren Gliederung und letztlich Portabilität des Systems beizutragen. So ist es z. B. üblich, die Endphase der Generierung durch eine sprachspezifische Grammatik zu steuern, die eine noch

---

[7] Die Bedeutung von *Produktion* im Rahmen von Phrasenstrukturgrammatiken ist in gewisser Weise kompatibel zu der hier verwendeten Terminologie, da es sich bei der Satzgenerierung in solchen Formalismen um ein Beispiel für das oben beschriebene Prinzip handelt.

[8] Vgl. die Unterscheidung zwischen prozeduralen vs. deklarativen Wissensrepresentationen.

sprachunspezifische semantische Spezifikation in die sprachliche Oberfläche übersetzt.

Allerdings ergibt sich beim Zusammenwirken von Planung und Produktion ein Schnittstellenproblem, das analog zu der Interaktionsproblematik bei Inhalt- und Formbestimmung charakterisiert werden kann. Die offensichtlichste Art des Zusammenwirkens beider Verfahren würde so aussehen, daß die Planungskomponente als Resultat eine Spezifikation für die Produktion liefert, die dann in Bezug auf den Produktionsformalismus interpretiert wird und das Endresulat erzeugt. Diese unidirektionale Schnittstellenkonzeption bringt jedoch auch hier erhebliche Probleme mit sich, wie z. B. in [Hovy1985a] erläutert wird.

Der Produktionsmechanismus muß typischerweise eine Reihe von Entscheidungen bezüglich mehrerer möglicher Alternativen der Expansion oder Realisation von sprachlichen Elementen treffen. Diese Entscheidungen müssen anhand der von der Planungskomponente gelieferten Spezifikation gefällt werden. Dies impliziert aber, daß die Planungskomponente die zu treffenden Entscheidungen und damit die entsprechende Spezifikationsinformation bereits kennt. Dies wiederum bedeutet praktisch, daß die Planungsphase einen Teil des Produktionsprozesses schon vorwegnehmen muß, was den Sinn der Trennung von Planung und Produktion *ad absurdum* führen würde. Auch bei diesem Problem besteht die Lösung in einer interaktiven, bidirektionalen Gestaltung der Schnittstelle, so daß die Produktionskomponente die nötige Information von der Planungskomponente bei Bedarf anfordert. Diese Anforderungen können die Planung selbst wiederum in Gang setzen, so daß sich hier das Bild von parallelen Prozessen ergibt, die durch eine interaktive Schnittstelle synchronisiert werden. In einem Fall ([Appelt1983a], siehe Abschnitt 2.3.1) ist dieses Modell tatsächlich sehr elegant verwirklicht worden.

2.2 Generierung und Psycholinguistik

2.2.1 Die psycholinguistische Dimension der Generierung

Als zusätzlichen Hintergrund für die Diskussion verschiedener Generierungssysteme werden in diesem Abschnitt einige psycholinguistische Arbeiten besprochen, die für das Generierungsproblem relevant sind. Es soll an dieser Stelle noch einmal daran erinnert werden, daß Sprachgenerierung in Verbindung mit in der Praxis einsetzbarer Software nur *eine* wichtige praktische Motivation für Forschung auf diesem Gebiet ist. Diese Arbeit orientiert sich primär an dem Ziel, das empirisch zu beobachtende sprachliche Verhalten von natürlichen Sprechern und damit auch die hierfür verantwortlichen *kognitiven Strukturen* zu modellieren. Diese Zielsetzung steht in gewisser Hinsicht in Konflikt mit dem Ziel, ein Generierungssystem zu entwickeln, das möglichst leistungsfähig ist und fehlerfreie Ausgaben produziert. Wir orientieren uns ausdrücklich nicht an der natürlichsprachlichen *Kompetenz* eines Sprechers, sondern an dessen *Performanz*. Tatsache ist aber, daß die typische Performanz natürlicher Sprecher höchst “ungenügend” und “fehlerhaft” ist, denn deren Äußerungen in alltäglichen Situationen sind gekennzeichnet durch laufend vorkommende Korrekturen, Abbrüche, Wiederholungen und anderweitig nicht wohlgeformte Syntax. (Für eine Systematik solcher syntaktischen Phänomene siehe [Rath1975a]). Man kann diese Unvollkommenheit menschlicher Performanz interpretieren als den Preis, den das menschliche Generierungssystem für seine enorme Flexibilität und

Variabilität zahlt. Im Gegensatz dazu sind künstliche Generierungssysteme wesentlich spezieller in ihrer Anwendbarkeit, produzieren dafür aber in ihrer beschränkten Domäne hundertprozentig grammatische Ausgaben (gemäß der ihnen eingegebenen Grammatik).

In diesem Zusammenhang ist sicherlich die Frage berechtigt, ob es außerhalb des reinen Erkenntnisstrebens überhaupt praktische Gründe gibt, die menschliche Performanz zu simulieren, da diese ja offensichtlich höchst unzureichend ist.[9] Auf diese Frage sind zwei Entgegnungen möglich: Zum einen scheint das Verständnis von mehr oder weniger fehlerbehafteten Äußerungen durch einen menschlichen Hörer im gleichen Maße problemlos zu sein, wie die Generierung solcher Äußerungen durch den Sprecher selbstverständlich ist. Mit anderen Worten: nicht nur die menschliche Sprachgenerierung, sondern auch die Spracherkennung ist so ausgelegt, daß “inkompetente” Äußerungen keine wesentliche Beeinträchtigung der Kommunikation bedeuten.[10]

Eher das Gegenteil ist der Fall. Man kann auf der Grundlage von transkribierten Gesprächen durchaus behaupten, daß gewisse Formen, die nach traditionellen Kriterien syntaktisch anomal sind, wie z. B. Ellipsen und Anakoluthe, eigentlich im Dienste der Sprachökonomie stehen. Dies wäre ein syntaktische Parallele zum semantisch-pragmatischen Phänomen der Metapher, die sich ebenfalls als Anomalie deuten läßt, die die Ausdrucksfähigkeit der Sprache und damit ihre Effizienz erweitert.

Darüberhinaus verfügen natürliche Sprecher typischerweise über ein Repertoire von Methoden, fehlerhafte Äußerungen zu korrigieren (*error correction devices*). Es gibt sogar Untersuchungen, die zeigen, daß scheinbare Fehler und die damit verbundenen Korrekturmaßnahmen vom Sprecher gezielt als Mittel der pragmatische Interaktion eingesetzt werden [Jefferson1974a].

Diese kurze Diskussion der eingangs gestellten Frage nach dem Sinn psycholinguistischer Sprechermodellierung sollte nicht den Eindruck erwecken, daß das Erreichen einer menschenähnlichen Virtuosität in Bezug auf sprachliche Performanz ein unmittelbar zu verfolgendes Forschungsziel ist. Dafür ist das verfügbare Wissen über menschliche kognitive Prozesse noch zu lückenhaft. Es sollte vielmehr die Beschäftigung mit psycholinguistischen Phänomenen auch dahingehend motiviert werden, daß es vielleicht in ferner Zukunft einmal möglich sein wird, bessere Mensch-Maschine-Schnittstellen zu bauen, die gerade wegen (und nicht trotz) ihrer Orientierung an menschlicher Performanz benutzerfreundlich sind.

2.2.2 Konnektionismus und Interaktionismus

James McClelland schlägt in seinem programmatischen Bericht [McClelland1987a] ein Sprachverarbeitungsmodell vor, das er als *interactive activation framework* bezeichnet. Dieses Modell ist auf den Prinzipien des *Konnektionismus* gegründet, einem Verarbeitungsparadigma,

---

[9] Diese Frage zielt insbesondere auf Anwendungen ab, in denen die Generierung *gesprochener* Sprache eine Rolle spielt, denn dort kommt die Diskrepanz zwischen Kompetenz und Performanz am deutlichsten zum tragen. Da heutzutage alle gängigen Systeme die *geschriebene* Sprache als Form der Ausgabe benuzten, stellen sich die hier angeschnittenen Fragen (noch) nicht in dem Maße.

[10] Zu möglichen Modellen für das Verstehen ungrammatischer Äußerungen vgl. [Taylor1984a].

das eine sehr große Zahl von relativ einfachen Verarbeitungseinheiten vorsieht, die über gewichtete Verbindungen verknüpft sind. Die Gesamtheit solcher Einheiten und ihrer Verbindungen wird auch als *neuronales Netz* bezeichnet, da die Verarbeitungseinheiten in konnektionistischen Modellen in Analogie zu den Nervenzellen des tierischen Zentralen Nervensystems gesehen werden.

Information wird in solchen Netzen durch eine numerische Größe repräsentiert, die *Aktivation*, die jeder einzelnen Einheit zugeordnet ist. Informationsverarbeitung wird dadurch realisiert, daß jede Einheit kontinuierlich die Aktivation der mit ihr verbundenen Einheiten (unter Berücksichtigung der Gewichtung der Verbindungen) aufsummiert und anhand des Resultats die eigene Aktivation neu bestimmt. Die Dynamik dieses fortlaufend parallel ablaufenden Prozesses wird als *Aktivationsausbreitung* (*spreading activation*) bezeichnet.

Eine der Intuitionen hinter diesem Modell ist die, daß die einzelnen Einheiten Hypothesen bzgl. bestimmter Aussagen über die jeweilige Domäne repräsentieren und ihre Aktivation dem Grad der Verläßlichkeit der jeweiligen Hypothese entspricht. Die Verarbeitung beruht nun darauf, daß sich miteinander kompatible Hypothesen gegenseitig unterstützen, indem sich die entsprechenden Einheiten durch *exzitatorische* Verbindungen (solche mit positivem Gewicht) gegenseitig verstärken. Entsprechend werden Einheiten, die konkurrierende Hypothesen repräsentieren, mit *inhibitorischen* Verbindungen (solchen mit negativem Gewicht) verknüpft. (Konnektionistische Modelle umfassen eine Fülle von Varianten was die Art der lokalen Informationsverarbeitung in den Einheiten und die Art der Verknüpfungen angeht. Es sollte hier lediglich ein kurzer Abriß der Prinzipien gegeben werden, deren Kenntnis für die folgende Diskussion nötig ist. Für eine detaillierte Beschreibung verschiedener Modelle wird auf [Rumelhart1986a] verwiesen.)

McClelland postuliert nun für Sprachverarbeitungsmodelle gewisse Eigenschaften und belegt diese anhand von empirischen Daten aus psycholinguistischen Experimenten. Wichtig in diesem Zusammenhang ist, daß er dabei zwar immer innerhalb des konnektionistischen Paradigmas diskutiert, daß seine Überlegungen zu den Eigenschaften und der Architektur des Verarbeitungsmodells aber durchaus auch auf nicht-konnektionistische, traditionelle, und das heißt in der Regel: symbol-manipulative Modelle anwendbar sind.

Wichtigste Eigenschaft der Architektur des Modells ist die Organisation der Repräsentation und Verarbeitung in Form einer hierarchischen Schichtung. Auch bei McClelland ergibt sich diese Annahme aus der traditionellen Systematik von linguistischen Beschreibungsebenen und er nimmt explizit die Ebenen Semantik, Syntax/Lexikon, Wort, Buchstabe/Phonem und optische/akustische Merkmale an. Es ist unmittelbar einsichtig, daß sich diese hierarchische Strukturierung ohne weiteres mit der in Abschnitt 2.1.2 vorgestellten vereinbaren läßt bzw. mit dieser sogar teilweise identisch ist.

Zwei weitere Annahmen bezüglich der Architektur des Modells beziehen sich explizit auf seinen konnektionistischen Charakter. Zum einen ist dies ein sog. *verteilter Konnektionismus*, d. h. die Annahme, daß innerhalb der einzelnen Ebenen die einzelnen Elemente der Repräsentation nicht direkt durch bestimmte Einheiten verkörpert werden (ein sog. *lokaler Konnektionismus*), sondern durch *Aktivationsmuster* über einer Teilmenge der Einheiten repräsentiert sind. Zweitens wird eine einschränkende Annahme über die mögliche Verbindungsstruktur des Netzes gemacht: Zwischen verschiedenen Ebenen sind nur exzitatorische

Verbindungen erlaubt, während innerhalb einer Schicht nur inhibitorische Vernetzung zugelassen ist.[11]

Die letzte und wesentliche Annahme, die des *Interaktionismus*, läßt sich wiederum problemlos auf andere hierarchisch organisierte, z. B. auf Planung oder Produktion basierende Modelle übertragen. Es ist dies (1) die Forderung, daß nur unmittelbar benachbarte Ebenen der Hierarchie Information austauschen, und (2), daß dieser Austausch (in McClellands Modell also exzitatorische Aktivation) nicht unidirektional, sondern bidirektional verläuft. Für die Spracherkennung, bei der der globale Informationsfluß von unten (den perzeptorischen Merkmalen) nach oben (zu der semantisch-pragmatischen Interpretation) verläuft, bedeutet dies, daß die Verarbeitung jeder Ebene insbesondere auch von den Resultaten der jeweils *darüber* liegenden Ebene beeinflußt wird.

Im Fall der Generierung mit ihrem globalen *top-down*-Ablauf ist die Lage genau symmetrisch: Von jeder Ebene kann eine Rückkopplung auf die jeweils darüberliegende Ebene ausgehen. Auch hier ist unmittelbar eine Parallele zu der schon diskutierten interaktiven Schnittstelle zwischen verschiedenen Stufen der Planung bzw. Produktion bei der Generierung zu erkennen.

In McClellands Bericht liegt der Schwerpunkt auf der Verifikation der oben wiedergegebene Hypothesen anhand von psycholinguistischen Daten aus der Sprach*erkennung*. Obwohl sich diese Ergebnisse nicht direkt auf die Generierung übertragen lassen, ist es doch interessant, sich die zentrale Frage dabei zu vergegenwärtigen. Es geht dabei in erster Linie um die Probleme der *Disambiguierung* und des *Kontexteinflusses*. Bei der eindeutige Erkennung sprachlicher Konstrukte, seien es nun Phoneme, Wörter oder Phrasenstrukturen, kann häufig nicht allein aufgrund von Information aus den vorherigen Erkennungsebenen entschieden werden. Erst der Einfluß des Kontextes auf der nächst höheren Stufe erlaubt dann eine Disambiguierung, so daß z. B. ein Phonem innerhalb eines Worts, ein Wort innerhalb eines Satzes oder eine Phrasenstruktur mithilfe der zugehörigen semantischen Interpretation eindeutig identifiziert werden kann.[12]

Für Kontextphänomene sind (mindestens) zwei Erklärungen denkbar. Wird bidirektionale Interaktion zwischen den Ebenen ausgeschlossen, so bleibt nur der Schluß, daß Ambiguitäten explizit an den nächsten Verarbeitungsschritt weitergereicht werden müssen in der Hoffnung, daß sie dort (oder später) aufgelöst werden können. Die interaktionistische Eklärung sieht dagegen vor, daß die Ambiguität lokal in der entsprechenden Erkennungsebene aufgelöst wird, indem durch Rückkoppelung aus dem Kontext, der in der darüberliegenden Ebene repräsentiert ist, zusätzliche Information verfügbar wird. Wie im Fall der interaktiven Schnittstelle bei der Generierung ist dabei die Parallelität der Verarbeitung auf zwei benachbarten Ebenen sowohl

---

[11] Zwei zusätzliche speziell “konnektionistische” Eigenschaften haben mehr technischen Charakter und betreffen den Verarbeitungscharakteristik einzelner Einheiten. (1) Die Ein-/Ausgabefunktion, die die gewichtete Summe der empfangenen Aktivation in den neuen Aktivationswert umrechnet, muß monoton, aber nicht-linear sein. (2) Die Aktivation einzelner Einheiten ist einem zeitlichen Schwund unterworfen, d. h. sie nimmt bei fehlender Eingabe von selbst in der Zeit ab.

[12] Der Kontext kann natürlich auch indirekt über mehrere Ebenen wirken, z. B. wenn die syntagmatische Umgebung eines Wortes dieses disambiguiert und dadurch erst ein einzelnes Phonem dieses Wortes erkannt wird.

Voraussetzung als auch natürliche Konsequenz der Interaktion.

McClelland argumentiert, daß das interaktive Modell das elegantere ist und dabei in voller Übereinstimmung mit allen relevanten empirischen Daten steht. Darüberhinaus wird ein Experiment vorgestellt, das eindeutig die interaktive Erklärung favorisiert.[13] Zusammenfassend läßt sich sagen, daß McClelland am Beispiel von Performanz in der Spracherkennung überzeugend für einen interaktiven Ansatz argumentiert. Seine Bevorzugung von konnektionistischen Modellen ist plausibel, wenn man bedenkt, daß es gelungen ist, Sprachverarbeitung auf den unteren Ebenen (vgl. Abschnitt 2.2.3) mit solchen Modellen in bestechend eleganter und einfacher Weise zu modellieren. Was die Realisierung konnektionistischer Lösungen auf höheren Verarbeitungsstufen angeht, so kann man sich nur folgendem Zitat anschließen:

> [...] *certainly no adequate model* [...] *has been proposed to date. From what we know about the susceptibility of higher levels of language processing to contextual information, it seems fairly clear to me that any adequate model will have to incorporate the principles of interactive activation. What is not clear at this point is how these principles will need to be elaborated and supplemented to capture the structural complexities that arise at higher levels. This remains a central issue for future research.* [S. 36]

2.2.3 Interaktionismus in der Phonemproduktion

Gary Dell hat ein konnektionistisches Modell entwickelt, in dem McClellands Konzeption im Bereich der Sprachproduktion auf niederer Ebene, nämlich bei der Generierung von Phonemen aus Morphemen, verwirklicht ist [Dell1985a]. Anhand dieses Modells kann er zeigen, daß Interaktion in konnektionistischen Modellen nicht nur bei der Spracherkennung konkrete Vorteile hat (dort ist dies relativ naheliegend, vgl. die Diskussion in Abschnitt 2.2.2), sondern daß gerade auch bei der Generierung die Eigenschaften des Modells dahingehend beeinflußt werden, daß bestimmte performative Charakteristiken menschlicher Sprachproduktion sehr gut modelliert werden.

Ein einfaches, nicht-interaktives Modell für die Phonemproduktion besteht aus zwei Mengen von Einheiten: solchen, die Morpheme repräsentieren, und solchen, die einzelnen Phonemen entsprechen. Dabei müssen die Phoneme noch nach ihrer Position innerhalb des Morphems differenziert werden, so daß es z. B. zwei Einheiten $/k/_1$ und $/k/_2$ für das Phonem /k/ gibt, die der Position von /k/ am Anfang bzw. am Ende eines Morphems entsprechen.[14] Die Einheiten sind so vernetzt, daß Morpheme genau die Phoneme (exzitatorisch) aktivieren, die zu ihrer Realisierung gebraucht werden, während sich innerhalb jeder der beiden Ebenen inkompatible Einheiten durch inhibitorische Verbindungen gegenseitig hemmen, d. h. um die Aktivation konkurrieren.[15] So aktiviert z. B. das Morphem $\langle$*cat*$\rangle$ seine Phoneme $/k/_1$, /a/ und $/t/_2$ während

---

[13] Es handelt sich um eine Interaktion zwischen Wort- und Phonemerkennung, von der gezeigt wird, daß sie ein Phänomen erzeugt, das rein auf der phonemischen Ebene angesiedelt ist. Daher kann hier nur der interaktionistische Ansatz eine Erklärung bieten.

[14] Vermutlich ist der Begriff *Morphem* hier falsch gewählt, da aus dem Artikel insgesamt implizit hervorgeht, daß hier eigentlich *Silben* gemeint sind.

[15] In zwei Punkten weicht Dells Modell dabei von McClellands Forderungen ab; erstens ist es ein Beispiel für lokalen (nicht verteilten) Konnektionsmus und zweitens ist sowohl Inhibition zwischen den Ebenen als auch Exzita-

$/k/_1$ und $/t/_1$ sich gegenseitig hemmen. Die Verarbeitung wird dadurch angestoßen, daß an der Eingangsebene (den Morphemen) ein sich zeitlich veränderndes Aktivationsmuster angelegt wird, das den zu artikulierenden Worten entspricht. Die Ausgabe kann dann an der Aktivation in der Phonemebene abgelesen werden.

Es ist nun die Frage, welchen Sinn es macht, dieses einfache Modell mit unidirektionaler Aktivation zwischen den Ebenen so zu erweitern, daß eine Rückkoppelung zwischen Phonemen und Morphemen stattfindet. Dies würde bedeuten, daß die Phoneme an all die Morpheme Aktivation zurückgeben, von denen sie angesprochen werden. Wie auch im Fall der Spracherkennung kann Rückkoppelung dazu beitragen, den Auswirkungen von *Rauschen* innerhalb des Netzes entgegenzuwirken. Versprecher, die andernfalls durch Rauschen in der Phonemebene verursacht würden, tendieren aufgrund der Rückkoppelung über die Morphemebene dazu, Phonemfolgen zu erzeugen, die legalen Morphemen entsprechen. Das aber bedeutet, daß der Verprecher u. U. faktisch gar nicht erst auftritt.

Allerdings muß diese zusätzliche Robustheit mit einer zusätzlichen Fehlerquelle bezahlt werden, nämlich der Möglichkeit des *Übersprechens* (*cross-talk*) zwischen verschiedenen Morphemen, die eine ähnliche Phonemstruktur haben. Über den Umweg der Phonemebene ist es nun nämlich möglich, daß ein Morphem ein anderes aktiviert, sofern sie Phoneme gemein haben. Zusammen mit anderen Störeffekten wie z. B. Rauschen kann dies letzlich dazu führen, daß die falschen Phoneme geäußert werden.

Dell vermutet, daß die positiven Effekte der Rückkoppelung die negativen überwiegen, will hier jedoch keine abschließende Aussage wagen, zumal sein System ja nur einen sehr kleinen Ausschnitt des gesamten Generierungsprozesses modelliert. In jedem Fall aber kann festgehalten werden, daß Dells Netzwerk auf überzeugende Weise im Stande ist, genau die *Fehler* (Versprecher) zu reproduzieren, die man bei menschlichen Sprechern beobachtet. Dazu gehört insbesondere die Neigung von Versprechern, lexikalische Phonemfolgen zu produzieren, sowie die Häufung von Versprechern bei benachbarten Wörtern mit ähnlicher Phonemstruktur, wobei hierbei häufig ein Austausch von entsprechenden Phonemen zu beobachten ist.[16]

2.2.4 Interaktion zwischen Syntax und Lexikon

Ebenfalls noch auf relativ niedriger Stufe, aber die Ebenen von Syntax und Lexikon betreffend beschäftigt sich Kathryn Bock mit Interaktionsprozessen in der Sprachgenerierung [Bock1982a, Bock1985a]. Auch ihr Ausgangspunkt sind empirische Untersuchungen, die auf Rückkoppelungsprozesse zwischen verschiedenen Komponenten der Generierung zu beruhen scheinen.

Bock geht von der Hypothese aus, daß die Modalitäten des Zugriffs auf Elemente des Lexikons eine Rolle für die Wahl der zugehörigen syntaktischen Konstruktion spielen. Diese Hypothese könnte eine Reihe von Phänomenen beschreiben, die in Experimenten auftreten, in

tion innerhalb einer Ebene erlaubt.

[16] Unabhängig von der Rückkoppelung zwischen den Ebenen können konnektionistische Modelle in eleganter Weise die Phänomene der Antizipation und Projektion erklären, wenn man die naheliegende Annahme macht, daß Aktivation in der Zeit stetig aufgebaut wird und ebenso wieder abfällt. So können Einheiten über den Zeitpunkt ihrer eigentlichen Aktivierung hinaus in der syntagmatischen Umgebung vor- bzw. nachwirken.

denen die Sprecher bei der Produktion oder Reproduktion von Sätzen innerhalb gewisser syntaktischer "Freiheitsgrade" wählen müssen. Solche Freiheitsgrade sind z. B. durch die Reihenfolge von Konstituenten in Koordinationen oder die Alternative zwischen Aktiv- und Passivkonstruktionen (mit ihrer entsprechenden Abfolge von Nominalphrasen) gegeben.

In solchen Experimenten kann man nun beobachten, daß bestimmte Sequentialisierungen der Konstituenten bevorzugt werden, und zwar in Abhängigkeit von deren lexikalischem Material. So werden z. B. bei koordinierten Nominalphrasen bevorzugt die an erster Stelle genannt, die entweder (a) kürzer sind oder die (b) ein Konzept von größerer Konkretheit ausdrücken. Werden bestimmte Lexeme einem *priming* unterworfen, indem man im vorangehenden Kontext Wörter oder Konzepte verwendet, die entweder (c) einen phonologischen oder (d) einen semantischen Bezug zu dem Lexem haben, so beeinflußt auch dies die bevorzugte Reihenfolge, in der die Konstituenten angeordnet werden. Dabei erzeugt semantisches *priming* die Tendenz, das entsprechende Wort an erster Stelle zu äußern, während phonologisches *priming* den entgegengesetzten Effekt hat.

Bock interpretiert diese Phänomene so, daß die "Zugänglichkeit" (*accessability*) lexikalischer Formen dahingehend auf die Wahl der syntaktischen Form wirkt, daß die jeweils "leichter zugänglichen" Lexeme bevorzugt bzw. an erster Stelle in die Konstruktion eingebaut werden. Andererseits wird natürlich der Zugriff auf das Lexikon ganz erheblich von den kategorialen Erfordernissen der syntaktischen Struktur bestimmt, so daß man hier von einem weiteren Beispiel bidirektionaler Beeinflussung, d. h. Interaktion zwischen sprachverarbeitenden Komponenten sprechen kann.

Die Autorin schlägt eine Architektur vor, die einerseits die Integration von syntaktischen und lexikalischen Formen leisten soll und andererseits die verschiedenen beobachteten Eigenschaften der Sprecherperformanz durch die Interaktionsprozesse zwischen den Komponenten erklären soll. Das Modell sieht vor, daß die lexikalisch-syntaktische Integration auf zwei Ebenen abläuft, der *funktionalen Integration* und der *Konstituentenintegration*. Die funktionale Integration besteht im Füllen von Kasusrahmen mit Lexemen, während im Zuge der Konstituentenintegration die Wortoberflächen mit der syntaktischen Struktur verknüpft werden. Die oben beschriebenen Interaktionsphänomene lassen sich danach unterteilen, ob sie auf der funktionalen (b, d) oder der Konstituentenebene (a, c) stattfinden. In jedem Fall wird der Integrationprozess von der Zugänglichkeit der an ihm beteiligen lexikalischen Formen beeinflußt.

In [Bock1982a] werden weitergehende Vorschläge bezüglich der konkreten Implementierung des Interaktionsprozesses gemacht. Zu diesem Zweck wird ein Produktionensystem in Erwägung gezogen, das um ein Element erweitert ist, das konnektionistischen Charakter hat. Klassische Produktionensysteme[17] sind im wesentlichen eine Menge von Regeln, die jeweils aus einer Bedingung und einer Aktion bestehen, wobei der Aktionsteil ausgeführt wird, sobald die Bedingung "feuert". Bei Bock erhält jede Produktion noch einen "Aktivierungsgrad", der so interpretiert wird, daß von mehreren konkurrierenden Regeln solche bevorzugt feuern, deren Aktivation relativ am höchsten ist. Die angestrebte wechselseitige Abhängigkeit zwischen

[17] Die Bezeichnung ist historisch begründet und hat keinen unmittelbaren Bezug zum Begriff *Produktion* wie er in Abschnitt 2.1.4 eingeführt wurde.

Syntax und Lexikon wird dadurch realisiert, daß lexikalische Regeln die Aktivation derjenigen syntaktischen Regeln erhöhen, mit denen sie kompatibel sind (und umgekehrt). Der Bezug zu den oben vorgestellten konnektionistischen Modellen ist offensichtlich und es ist interessant festzustellen, daß in diesem Modell auf sehr natürliche Weise Konzepte aus traditionellen und konnektionistischen Ansätzen vereinigt sind.

2.3 Existierende Systeme

In diesem Abschnitt werden einige der existierenden Generierungssysteme vorgestellt und besprochen, wobei jeweils versucht werden soll, die Arbeiten in Bezug auf die Konzepte einzuordnen, die in den vorangegangen Abschnitten vorgestellt wurden.

Natürlich kann die hier getroffene Auswahl keine Vollständigkeit beanspruchen, andererseits wurde versucht, zumindest eine repräsentative Übersicht zu geben.

2.3.1 Sprechaktplanung in KAMP

In Douglas Appelts Sprachplanungssystem KAMP [Appelt1985a] finden sich viele der diskutierten Konzepte “in Reinkultur”. Will man verschiedene Systeme bezüglich der von ihnen abgedeckten linguistischen Komponenten vergleichen und nimmt man die in Abschnitt 2.1.2 vorgestellte linguistische Hierarchie als Maßstab, so umfaßt Appelts System unzweifelhaft den größten Ausschnitt dieser Hierarchie. Dies liegt vor allem daran, daß KAMP nicht nur Sprache generieren, sondern einem menschlichen Partner bei der Abwicklung komplexer Aufgaben (z. B. der Montage von Machinenteilen) assistieren soll. Die generierten Äußerungen sind dabei lediglich Teile eines umfassenden Aktionsplans, der darauf abzielt, den menschlichen Parter in die Lage zu versetzen, die geforderte Aufgabe zu lösen. Daraus ergibt sich unmittelbar die pragmatische Einbettung des Generierungsprozesses und die Tatsache, daß sämtliche sprachlichen und nichtsprachlichen Aktionen des Systems im Rahmen von Sacerdotis allgemeinem hierarchischem Planungsverfahren (siehe Abschnitt 2.1.4) abgewickelt werden.

KAMP umfaßt also die gesamte Hierarchie von der domänenspezifischen Pragmatik bis zur Generierung lexikalischer Oberflächen, wobei allerdings eine explizite Behandlung von Morphologie und Phonologie (wie allgemein üblich) umgangen wird. Darüber hinaus stellt KAMP eine ernstzunehmende Implementierung des allgemeineren Paradigmas II aus Abschnitt 2.1.1 dar, denn das System plant seine Äußerungen nicht als natürlichsprachliche Übersetzung interner Datenstrukturen, sondern als die Realisierung von Sprechakten. Sprechakte sind definiert durch die Art und Weise, in der sie die Intentionen und das Wissen des Hörers modifizieren. Sowohl diese sprecher-/hörerinternen Sachverhalte wie auch der Zustand der externen Welt sind mittels Modallogik formalisiert, so daß ein Theorembeweiser für die Verifikation der erstellten Pläne eingesetzt werden kann.[18]

[18] Die modallogische Formalisierung dient dabei nur zur *Verifikation* der Pläne. Deren *Erstellung* wird aus Effizienzgründen durch Heuristiken gesteuert, die auf einer vereinfachten Formalisierung beruhen.

Durch die Einbettung der Sprechaktplanung in den allgemeinen Planungsprozeß ist es möglich, sprachliche und nichtsprachliche Kommunikationsmittel zu integrieren. So können z. B. Zeigegesten in Verbindung mit deiktischer Referenz auf einen Gegenstand (*“Das da!”*) geplant werden. Die Fähigkeit des Planers NOAH, globale Planoptimierung durch Plankritiker vorzunehmen, spiegelt sich nun in der Möglichkeit wieder, mehrere Sprechakte unter einen einzigen zu subsumieren. Typischerweise werden z. B. Nominalphrasen benutzt, um einerseits auf vorhandene Objekte zu referieren und andererseits neue Information über das Objekt zu liefern. So kann die Information

(5) *Der Schraubenzieher liegt auf dem Tisch.*

und die anschließende Anweisung

(6) *Entferne den Deckel mit dem Schraubenzieher!*

zusammengefaßt werden zu

(7) *Entferne den Deckel mit dem Schraubenzieher auf dem Tisch!*

Gegenüber anderen Systemen hat KAMP den Nachteil, daß kein besonderer Wert auf die Kohärenz von größeren Texteinheiten gelegt wird. Die Konzeption hat zur Folge, daß immer nur einzelne Sätze generiert werden, die zwar kohärent in Bezug auf den Handlungskontext sind, aber nicht unbedingt bezüglich des sprachlichen Kontexts. Man kann jedoch argumentieren, daß sich Diskurs- und Dialogplanungswissen ohne weiteres in das übrige System integrieren ließen, wobei in der von Appelt diskutierten Anwendung deshalb dafür kein unmittelbarer Bedarf vorlag, weil die Kohärenz auf Planebene eine hinreichende Textkohärenz gewährleistet.

Ein schwerwiegender Nachteil von KAMP in seiner ursprünglichen Form ist jedoch die fehlende Trennung des sprachlichen Wissens vom restlichen Planungswissen (vgl. die Diskussion in Abschnitt 2.1.4). Insbesondere ist der Aufbau der syntaktischen Struktur so eng mit den übrigen Planungsoperatoren verflochten, daß von einer sauberen linguistischen Fundierung (zumindest auf Syntaxebene) keine Rede sein kann. Aus diesem Grund wird in [Appelt1983a] eine neue Kontrollstruktur vorgeschlagen, die den konventionellen Planungsprozeß erweitert, um die Interaktion mit einem deklarativen Grammatikformalismus zu ermöglichen.

In der *Functional Unification Grammar* (FUG, [Kay1984a]) erfolgt die Generierung einer syntaktischen Struktur durch die *Unifikation* zweier Datenstrukturen, nämlich der Grammatik selbst und der Spezifikation des zu generierenden Satzes. Die Unifikation ist dabei ein Prozeß, der die Informationen aus beiden Datenstrukturen vereinigt, sofern sie kompatibel ist, und als Resultat eine dritte Struktur erzeugt, die einerseits der Grammatik der Sprache und andererseits der Eingabespezifikation genügt.[19] Da es sich hierbei um die Interpretation eines streng deklarativen Formalismus handelt, haben wir es hier mit einem typischen Produktionsverfahren im Sinne von Abschnitt 2.1.4 zu tun.

---

[19] Eine exakte Beschreibung des Unifikationsprozesses folgt in Abschnitt 3.1 in Verbindung mit einer allgemeinen Diskussion von unifikationsbasierten Grammatiken. Für eine Einführung in unifikationsbasierte Formalismen wird auf [Shieber1986a] verwiesen.

Appelt hat den ursprünglichen Unifikationsalgorithmus so modifiziert, daß dieser in geeigneter Form mit dem Planer kommunizieren kann. Ausgangspunkt dabei ist eine Eingabespezifikation, die unvollständig ist und daher normalerweise die zu generierende Äußerung nicht hinreichend spezifizieren würde. Trifft der Unifikator bei der Abarbeitung dieser Spezifikation auf eine unvollständige Komponente, so unterbricht er sich und ruft den Planer mit einer entsprechenden Planungsaufgabe auf. Dessen Resultat wiederum vervollständigt die Spezifikation und läßt den Unifikator dann fortfahren. Auf diese Weise verhalten sich Planung und Produktion wie zwei quasi-parallele Koroutinen, die bei Bedarf die Kontrolle explizit an den jeweils anderen Prozeß abgeben. Es handelt sich hier um eine “echte” Interaktion zwischen Planung und Produktion, da im Laufe der Unifikation der Planer u. U. veranlaßt wird, bereits erstellte Pläne zu modifizieren. Dies tritt z. B. dann ein, wenn die Grammatik der Sprache die Subsumierung von informativen Sprechakten unter nominaler Referenz erlaubt.

2.3.2 Diskursplanung in TEXT

Eines der Probleme, die in Appelts System vernachlässigt wurden, die Planung von kohärenten Diskursen, steht bei Kathleen McKeowns TEXT im Fordergrund [McKeown1985a]. Es handelt sich dabei im wesentlichen um eine natürlichsprachliche Datenbankschnittstelle, die dem Benutzer auf Anfrage Informationen aus und über eine (militärische) Datenbank in Form von zusammenhängenden Texten gibt.

TEXT geht von einer nicht-natürlichsprachlichen Anfrage aus und bestimmt daraus, (1) welche Informationen aus der Datenbank für die Beantwortung relevant sind und (2) wie diese Information zu strukturieren ist. Als Ergebnis dieser “strategischen Komponente” wird eine Folge von vollständigen Satzspezifikationen generiert, die von der “taktischen Komponente” in Satzoberflächen überführt werden.

Damit ist klar, daß hier im Vergleich zu KAMP ein aus pragmatischer Sicht eingeschränktes Modell Anwendung findet, allerdings mit dem Ziel, sich auf einen speziellen Aspekt sprachlichen Wissens zu konzentrieren. TEXT verkörpert in typischer Form das Paradigma I der Sprachgenerierung, da man die Operation des Systems als eine Datentransformation charakterisieren kann. Die linguistische Hierarchie wurde im wesentlichen auf zwei Komponenten reduziert, wobei die erste einem Ausschnitt der linguistischen Pragmatik (Diskursplanung) entspricht und die zweite Komponente die formale Realisierung durch Syntax und Lexikon besorgt. Eine vorteilhafte Konsequenz aus dieser (nur) zweischichtigen Struktur ist jedoch die saubere Trennung des jeweils verwendeten linguistischen Wissens, was auch dadurch unterstützt wird, daß in beiden Komponenten ganz unterschiedliche Formalismen und Generierungsmechanismen Anwendung finden. Allerdings ist die Schnittstelle zwischen Inhaltsbestimmung und Formbestimmung streng sequentiell angelegt, obwohl McKeown behauptet, daß das System so erweitert werden könnte, daß Backtracking zwischen taktischer und strategischer Komponente stattfinden kann.

Der Diskurs wird von TEXT durch sog. *Schemata* strukturiert, wobei jedem Schema eine Strategie zur Realisierung eines kommunikativen Ziels oder rhetorischen Effekts entspricht. So wird z. B. die Struktur einer Objektdefinition durch das “Identifikations”-Schema strukturiert:

Identifikation:

(a) Identifiziere Klassenzugehörigkeit.

(b) Beschreibe Zusammensetzung und/oder Eigenschaften.

(c) Zeige Analogien auf.

(d) Gebe Beispiele.

Allgemein sind solche Schemata reguläre Ausdrücke über *rhetorischen Prädikaten* wie Vertiefung, Erklärung, Attributierung, Analogisierung usw. Die einzelnen Schritte eines Schemas werden entweder direkt durch Sätze oder rekursiv durch weitere Schemaanwendungen realisiert. So bekommt der Text eine rekursive, hierarchische Struktur, deren Tiefe den Umfang des generierten Diskurses bestimmt. Als Kontrollstruktur für die Abarbeitung der Schemata wird eine modifizierte Version von ATNs (*Augmented Transition Networks,* [Woods1970a]) verwendet.

Die Schemata sind eine deklarative Form der Wissensrepräsentation, die das hierarchische Planen von Textinhalt- und struktur steuern. Wir haben es also hier mit einer Zwischenform von Planungs- und Produktionsverfahren zu tun.

Durch die Schemaanwendung ist allerdings die Sequentialisierung der Information noch nicht eindeutig festgelegt. Einerseits können z. B. in einer Beschreibung die Attribute in verschiedenen Abfolgen realisiert werden. Andererseits ist auch innerhalb eines Satzes eine gewisse Freiheit bezüglich der Sequentialisierung von Nominalphrasen gegeben, indem z. B. zwischen Aktiv- und Passivformen gewählt werden kann. Für beide Arten der Sequentialisierung bedient sich TEXT sog. Fokusregeln (*focus constraints*), die angeben, wie in einem kohärenten Text der thematische Schwerpunkt beibehalten oder verschoben werden kann [Sidner1979a].

Aus diesen Regeln kann dann sowohl die Reihenfolge der zu generierenden Sätze als auch die interne Fokusstruktur der Sätze abgeleitet werden. Inhalt und Fokusinformation werden in einer funktionalen Beschreibung kodiert, die dann von einer FUG interpretiert werden. Die Interpretation dieser sog. *message* durch die taktische Komponente erfolgt autonom und bedarf keiner weiteren Interaktion mit der strategischen Komponenten (vgl. Appelts TELEGRAM, Abschnitt 2.3.1).

2.3.3 Portable Generierung in PENMAN

Auch William Manns Generierungssystem PENMAN basiert auf der Trennung von Textplanung und Satzproduktion, wobei besonderes Augenmerk auf die Portabilität der Produktionskomponente gerichtet wurde. Die Semantik, die die Satzproduktion steuert, sollte so ausgelegt werden, daß sie unabhängig von der Form der in der Textplanung verwendeten Repräsentation beibehalten werden kann. Daneben ist noch eine nachgeschaltete Textverbesserung vorgesehen, die die erzeugte Ausgabe kritisch evaluiert und deren Analysen mit der Textplanung rückgekoppelt werden. Dieser Teil des Systems soll jedoch erst in einem späteren Implementierungsstadium realisiert werden [Mann1983a].

Die Textplanungskomponente von PENMAN basiert ähnlich wie bei McKeown auf einer rekursiven Strukturierung des Inhalts, wobei als Kontrollstruktur wie auch in Appelts System der hierarchische Planer NOAH zum Einsatz kommt. Mann schreibt, daß sich die beiden wesentlichen Wissensquellen für die Textplanung, die Diskurstheorie und die Lesermodellierung, noch in der Entwicklung befinden und diese Komponente daher ebenfalls noch nicht implementiert ist.

Der Schwerpunkt von PENMAN liegt daher auf der portablen Produktion von Sätzen. Als zugrundeliegende linguistische Theorie dient Hallidays *systemische Grammatik* [Halliday1976a], die gerne in der Generierung verwendet wird, weil sie dem dort immer wieder stattfindenden Prozeß der Auswahl aus möglichen linguistischen Alternativen entgegenkommt.[20]

Eine systemische Grammatik besteht im wesentlichen aus einem gerichteten azyklischen Graphen, in dem jeder Knoten ein linguistisches Merkmal repräsentiert. Die einzelnen Kanten, die von einem Knoten zu seinen Nachfolgeknoten führen, stehen jeweils für eine Möglichkeit, dieses Merkmal zu realisieren. So könnte z. B. einer der ersten Knoten nahe der Wurzel des Graphen die Wahl des Satzmodus repräsentieren, wobei die weiterführenden Kanten für Indikativ, Konjunktiv, Imperativ usw. stehen. Über die Kanten Indikativ und Konjunktiv gelangt man dann vielleicht zu einem Knoten, an dem über mögliche Tempora entschieden wird usw. Die einzelnen Knoten mit ihren Alternativen bezeichnet man als *Systeme*, woraus sich der Name dieses Formalismus ableitet.

Eine Traversierung des Graphen entspricht einer Festlegung aller relevanten Merkmale eines Satzes und seiner Konstituenten und damit dessen vollständigen Spezifikation. Der Generator nimmt daher genau solch eine Traversierung vor und muß dabei an jedem Knoten entsprechend der Semantik des zu erzeugenden Satzes eine Alternative wählen. Damit dieser Prozeß unabhängig von nicht-linguistischen Verarbeitungsschritten erfolgen kann, wird die Kommunikation mit der nicht-linguistischen Umgebung über eine sog. Anfragesemantik (*inquiry semantics*) geführt.

Hierfür ist jedem System (Knoten) der Grammatik ein "Entscheidungsexperte" zugeordnet, der lokales Wissen über die Auswahlkriterien an diesem Punkt enthält. Wird das System traversiert, so richtet dieser Experte eine Anfrage an die Umgebung und entscheidet sich aufgrund der Antwort für eine Alternative. Wenn z. B. ein Experte, der für die Charakteristik einer Nominalphrase zuständig ist, die Wahl zwischen Definitheit und Indefinitheit dieser NP treffen muß, so kann er das u. a. dadurch tun, daß er die Anfrage "Ist das durch die NP referierte Objekt für den Hörer identifizierbar?" an die umgebenden Komponenten (hier das Hörermodell) richtet.

Die Menge der möglichen Anfragen definiert eine Schnittstelle, über die verschiedenen Anwendungen die Generierungskomponente benutzen können. Die Schnittstelle ist pseudointeraktiv, da die Satzgenerierung zwar Daten an die Textplanung übermittelt, aber das nur in Form von Anfragen, die allein dem Zweck dienen, Informationen von der Textplanung zu erhalten. Im Prinzip könnte zwar die implizite Information, die in den Anfragen enthalten ist, wiederum in die Textplanung eingehen, doch wäre dies eine mißbräuchliche Ausnutzung der Schnittstelle, die ihrer Konzeption zuwiderlaufen würde.

[20] Vgl. u. a. [Davey1979a].

2.3.4 Interaktion von Planung und Produktion in PAULINE

Eduard Hovy legt in seinem System PAULINE besonderen Wert auf die Wechselwirkungen zwischen den pragmatischen Charakteristika des Sprecher-Hörer-Verhältnisses und der Form des zu generierenden Textes [Hovy1984a, Hovy1985a]. Dabei werden im Grunde genommen nur Techniken benutzt, die schon in der klassischen Rhetorik bekannt waren. In beiden Fällen geht es darum, die sprachliche Form so zu wählen, daß bestimmte Sprecherintentionen optimal unterstützt werden und beim Hörer gewisse Effekte erzielt werden: Beinflussung seines Wissens, seiner Meinungen, seines emotionalen Zustandes oder seiner Beziehung zum Sprecher.

Die sprachliche Form umfaßt dabei praktisch alle linguistischen Beschreibungsebenen. Beispielsweise kann schon die Auswahl eines von mehreren möglichen Themen ein Mittel zur gezielten Beeinflussung darstellen. Anschließend muß dann sowohl der Satzbau als auch die Wortwahl der kommunikativen Situation angepaßt werden.[21] Man sieht schon an einfachen Beispielen, daß pragmatisch motivierte Entscheidungen praktisch kontinuierlich den sprachlichen Produktionsprozeß beeinflussen. Um zu vermeiden, daß die pragmatische Planung auch die formalen Aspekte der Produktion "voraussehen" und das heißt: vorausberechnen muß, schlägt Hovy einen interaktiven Mechanismus vor.[22] Das Verfahren zur Sychronisation und Kommunikation zwischen Planung und Produktion ist dem von Manns PENMAN sehr ähnlich, jedoch mit dem wichtigen Unterschied, daß "echte" Interaktion stattfindet, indem der Planer die sprachlichen Möglichkeiten, die ihm die Produktion bietet, in seine weitere Planung miteinbezieht.[23]

Die Produktion läuft dabei auf fünf Ebenen ab: Themenwahl, Bestimmung des Satzinhalts, Satzbau, Konstituentenbau und Wortwahl. Auf all diesen Ebene müssen Entscheidungen getroffen werden, die potentiell pragmatische Information benötigen. Die Interaktion mit dem Planer geschieht über sog. *Vorschläge*, die dieser zu jeder Entscheidung abgibt. Solche Vorschläge sind weder obligatorisch, noch müssen sie eindeutig sein; die Produktionskomponente wählt nach irgendeiner Heuristik (z. B. den erstbesten) aus.

Wie auch bei Manns PENMAN stellt sich das Problem, trotz der vorgesehenen Interaktion zwischen Planung und Produktion die Modularität der Konzeption zu wahren. So ist es z. B. nicht offensichtlich, wie der Planer Vorschläge für die Produktion der sprachlichen Form abgeben soll, ohne selbst über eigenes sprachspezifisches Wissen zu verfügen. Genau das aber würde einer sauberen Modularisierung des Generierungswissens zuwiderlaufen. Hovy hat dieses Problem dadurch gelöst, daß seine "Vorschläge" nicht explizit irgendwelche sprachlichen Formelemente beinhalten, sondern lediglich (prozedurale) Kriterien zur Auswahl solcher Formelemente. Diese Kriterien wiederum sind so gehalten, daß sie (möglichst) keines sprachspezifischen Wissens bedürfen. So könnte ein Vorschlag für die Entscheidung einer Wortwahl in etwa so umschrieben werden: *Wähle ein Wort, das dem Hörer bekannt (nicht*

[21] Gilt es z. B., den Hörer einzuschüchtern, so könnte hierzu eine besonders komplexe Syntax oder die Wahl von möglichst vielen Fremdwörtern dienen.

[22] Der Titel "Integrating Text Planning and Production" [Hovy1985a] ist im Zusammenhang mit der hier verwendeten Terminologie etwas irreführend, da es sich gerade *nicht* um einen *integrierten* sondern um einen *interaktiven* Ansatz handelt.

[23] Hovy bezeichnet dieses Prinzip als "Opportunismus".

bekannt) ist.

2.3.5 Reine Produktionssysteme

Zum Abschluß dieses Kapitels sollen noch zwei Systeme besprochen werden, die das Generierungsproblem nicht in derselben Breite angehen, wie das die oben erwähnten Systeme – zumindest ansatzweise – tun. Diese bestanden alle im wesentlichen aus einer Planungs- und einer Produktionskomponente, wobei die größten funktionalen Unterschiede in der durch die Planungskomponente bestimmten pragmatischen Zielsetzung zu finden waren.

Die in diesem Abschnitt vorgestellten Ansätze beschäftigen sich ausschließlich mit der Frage, wie eine *Satzbedeutung* (eine möglichst sprachunabhängige Repräsentation der Semantik eines Satzes) in eine sprachliche *Oberfläche* (eine lineare Folge von Wörtern) umzusetzen ist. (In der Hierarchie aus Abschnitt 2.1.2 entspricht dies den Ebenen von der Syntaxkomponente abwärts.)

Aufgrund ihres isolierten Charakters ordnen sich beide Ansätze implizit ein in das nichtinterkative, sequentielle Paradigma, wobei allerdings die Frage offenbleibt, inwieweit durch Modifikationen eine Einbindung in einen interaktiven Verarbeitungzusammenhang mit anderen Komponenten erreicht werden kann. Während die Frage der Kommunikation zwischen verschiedenen Komponenten bei der bisherigen Diskussion im Vordergrund stand – weil sie m. E. eine zentrale Rolle für die globalen Eigenschaften eines Generierungssystems spielt –, liegt der Schwerpunkt bei den hier besprochenen Systemen auf der Lösung spezieller, eher technisch motivierter Teilprobleme, die aber nichtdestoweniger von Interesse sind.

2.3.5.1 BABEL: Generierung aus CD-Graphen

Neil Goldmans BABEL [Goldman1975a] ist eines von drei Elementen eines sprachverarbeitenden Systems, das auf einer von Roger Schank entwickelten Form der semantischen Repräsentation, der *conceptual dependency* (CD), basiert [Schank1975a].[24]

Das Problem der Umsetzung von semantischen Darstellungen in sprachspezifische Formen taucht zwangsläufig in jedem Generierungssystem auf (es ist gewissermaßen deren kleinster gemeinsamer funktionaler Nenner). Dabei ist diese Aufgabe umso einfacher zu lösen, je weitgehender die gewählte Semantik zu der verwendeten Sprache isomorph ist. Tatsächlich ist es so, daß die meisten Systemes aus praktischen Erwägungen eine semantische Darstellung wählen, deren Objekte einigermaßen direkt mit Lexemen der Sprache korrelieren und deren semantische Relationen unmittelbar auf syntaktische Relationen abgebildet werden können. Genau dies ist bei BABEL nicht der Fall, und hierin liegt das Besondere seiner Aufgabe.

Die Eingabespezifikation, sog. CD-Graphen, sind Netzwerke, die die konzeptuellen Beziehungen und Abhängigkeiten zwischen Objekten und Relationen zwischen Objekten darstellen. Die eigentliche Besonderheit an dieser Darstellung ist ihre Feinkörnigkeit (feine Granularität),

---

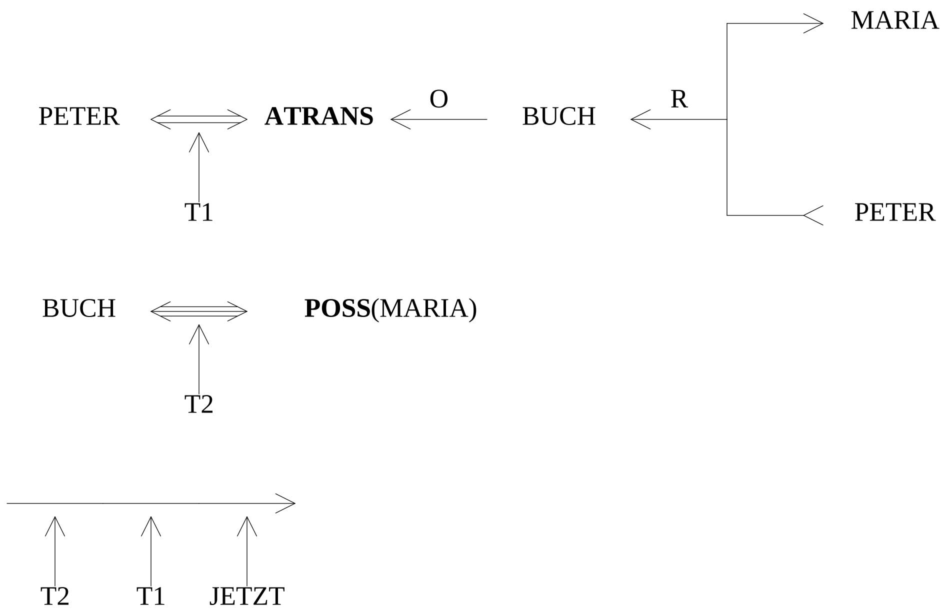
[24] Obwohl BABEL in dieses System eingebettet ist, werden dessen Elemente doch als autarke “Bausteine” betrachtet, die unabhängig voneinander spezielle Aufgaben erfüllen. Neben BABEL gibt es noch eine Sprachanalyse- und eine Inferenzkomponente, die durch geeignete Zusammenschaltung u. a. ein System zur Paraphrasierung bzw. zur Übersetzung von Eingabesätzen liefern.

denn CD verwendet nur eine handvoll primitiver Relationen, um alle möglichen Sachverhalte zu charakterisieren. Im Prozeß der *konzeptuellen Analyse* werden komplexe Sachverhalte in CD-Primitive und -Relationen zerlegt. Der Satz

(8) *Peter gab Maria das Buch zurück.*

wird analysiert als

(9)



Dabei steht **ATRANS** für einen "abstrakten Transfer" von Besitz und **POSS** für das Besitzprädikat. Die Objekte und Primitive des Graphen sind durch die Relationen $\Longleftrightarrow$ (Agens), $\xleftarrow{\text{O}}$ (Objekt, Patiens), $\xleftarrow{\text{R}}$ (Richtung, Rezipient), $\Longleftrightarrow$ (prädiziertes Objekt) und $\uparrow$ (Zeitpunkt) verknüpft und die Zeitpunkte auf einer Zeitachse angeordnet. Bei der Generierung aus CD-Repräsentationen besteht nun ein zentrales Problem darin, daß die Abbildung zwischen CD-Primitiven und Lexemen nicht eindeutig ist. **ATRANS** z. B. kann abhängig vom Kontext als *geben*, *nehmen*, *kaufen*, *leihen* usw. realisiert werden.

Goldman löst dieses Problem mithilfe von Entscheidungskaskaden (*discrimination nets*), in denen eine Folge von binären Fallunterscheidungen über dem CD-Graphen zur Auswahl eines Hauptverbs für den zu generierenden Satz führt. Der Lexikoneintrag dieses Verbs enthält Anweisungen, wie Teile des Graphen anschließend auf grammatische Funktionen (Subjekt, Objekt, präpositionale Ergänzungen) abgebildet werden, worauf dann diese Teilgraphen selbst wiederum auf Lexeme abgebildet werden. Resultat dieses Prozesses ist ein sog. Syntaxnetzwerk (*syntax network*), ein gerichteter Graph, der den Satz durch seine abstrakte grammatische Funktionen beschreibt.[25] Diese Netzwerke werden schließlich durch eine ATN-Variante (*augmented finite state transitions network*, AFSTN) linearisiert und in Oberflächenformen übersetzt.

[25] Diese Graphen haben eine formale und funktionale Ähnlichkeit mit den *functional descriptions* der FUG.

2.3.5.2 Produktion in konnektionistischen Netzen

Das letzte hier diskutierte Generierungsmodell ist wiederum auf die Umsetzung satzsemantischer Repräsentationen in Satzoberflächen beschränkt. Es ist jedoch eines der wenigen nicht-symbolmanipulativen Generierungssysteme überhaupt, da es das in Abschnitt 2.2.2 kurz vorgestellte Verarbeitungsparadigma des Konnektionismus verwendet.

Jugal Kalita und Lokendra Shastri stellen ein Netzwerk vor, in dem durch Aktivationsausbreitung in mehreren Schichten von Verarbeitungseinheiten schließlich eine zeitliches Aktivationsmuster erzeugt wird, das der Wortfolge des generierten Satzes entspricht [Kalita1987a]. Eingabe für den Generierungsprozeß ist ein anderes Aktivationsmuster, das die zu äußernden Konzepte (Prädikat, Subjekt, Objekt) spezifiziert. Das Modell verwendet eine lokale Repräsentation, d. h. jeweils eine Einheit entspricht einem Wort bzw. Konzept. Das Gesamtnetzwerk ist ähnlich wie bei [Dell1985a] (siehe Abschnitt 2.2.3) in (hier fünf) Schichten strukturiert, wobei die Aktivationsausbreitung im wesentlichen ohne Rückkoppelungen *top-down*, also nicht-interaktiv, erfolgt. Neben der Eingabeebene (Konzepte) und der Ausgabeebene (morphologische Wortformen) gibt es noch eine Ebenen, die globale Satzmerkmale repräsentiert (SVO-Ordnung, Aktiv-/Passivkonstruktion), sowie zwei Ebenen, in denen nicht-terminale bzw. terminale Konstituentenkategorien kodiert werden.

Die generierten Sätze haben alle die Konstituentenstruktur

(10) $$[[[\alpha]_{DET}[\beta]_N]_{NP}[\gamma]_V[[\delta]_{DET}[\varepsilon]_N]_{NP}]_S\ ,$$

was dadurch bedingt ist, daß das verwendete Netz keine echte Rekursion von Produktionsregeln zuläßt. Beschränkt man sich auf eine vorgegebene Struktur (oder auf eine endliche Anzahl solcher Strukturen), so kann das Problem der Syntaxgenerierung mit prinzipiell den gleichen Mitteln behandelt werden, wie die Generierung von phonologischen Strukturen aus morphologischen Formen. Die wesentlichen Probleme, die in Lokindas und Shastris System behandelt wurden, betreffen die zeitliche Sequentialisierung in Aktivationsmustern und die zumindest angenäherte Verwirklichung einer rekursiven Struktur, indem verschiedene Instantiierungen von NP-Realisierungen durch ein und dasselbe Teilnetzwerk abgewickelt werden.[26]

So gesehen ist mit diesem System ein erster Ansatz zur Implementierung allgemeiner Syntaxgenerierung in konnektionistischen Systemen gegeben, der deren prinzipielle Machbarkeit demonstriert, aber noch wesentlicher Erweiterungen bedarf.

---

[26] Die Implementierung einer allgemeinen Rekursion ist eng verknüpft mit einem anderen für konnektionistische Systeme typischen Problem: der Frage, wie Variablenbindungen realisiert werden können.

3 Unifikationsbasierte Grammatiken

In diesem Kapitel wird in knapper Form der Grammatikformalismus eingeführt, der später als Grundlage für Sprachgenerierung verwendet wird. Dieser Formalismus – sowie eine Reihe von verwandten Grammatikmodellen, die aufgrund ihres geographischen Ursprungs oft unter der Sammelbezeichnung "Bay Area Grammars" geführt werden – machen extensiven Gebrauch von der algebraisch-graphentheoretischen Operation der *Unifikation*, die im folgenden eingeführt wird. Für diesen Typus von Grammatiken, der Unifikation als wesentliches formales Hilfsmittel einbezieht, hat sich der Begriff *unifikationsbasierte Grammatik* (*unification-based grammar*, hier auch kurz: *Unifikationsgrammatik*) eingebürgert.

Eine didaktisch orientierte Einführung sowohl in die Unifikation als auch in verschiedene auf sie gestützte Grammatikmodelle findet sich in [Shieber1986a].

3.1 Unifikation

Hinter dem Begriff "Unifikation" steht eine sehr allgemeine Intuition, die dazu geführt hat, daß der Terminus in den verschiedensten Modellen und Theorien in formal recht unterschiedlicher Weise verwendet wird; oft wird der Begriff auch lediglich informell benutzt. Im Rahmen dieser Arbeit wird unter Unifikation die im folgenden definierte algebraische Operation auf einer bestimmten Menge von Datenstrukturen, den sog. *f-Strukturen*, verstanden.

3.1.1 f-Strukturen

f-Strukturen haben sich in der Linguistik und auf anderen Gebieten als eine flexible Beschreibungsform für theoretische oder empirische Entitäten eingebürgert. Zugrunde liegt die Vorstellung, daß ein zu beschreibendes Objekt durch eine Menge F von *Merkmalen* (*features*) charakterisiert ist, denen *Werte* aus einer Menge V zugeordnet sind. Die Gesamtheit der Merkmale und der ihnen zugeordneten Werte beschreibt oder definiert – je nach ontologischer Perspektive – das Objekt vollständig. Formal gesehen ist eine solche f-Struktur (*feature structure*) ein Element der Menge V^F der partiellen Funktionen von F in V. Typographisch werden solche f-Strukturen als Matrizen von Merkmal-Wert-Paaren (*Merkmalsmatrizen*) dargestellt:

$$\begin{bmatrix} f: & a \\ g: & b \\ \vdots & \vdots \end{bmatrix},$$

wobei $f, g, \ldots \in F$ und $a, b, \ldots \in V$.[27]

f-Strukturen lassen sich sinnvollerweise dahingehend verallgemeinern, daß die Werte von Merkmalen selbst wiederum beliebige f-Strukturen sein können:

$$\begin{bmatrix} f: & a \\ g: & \begin{bmatrix} h: & c \\ \vdots & \vdots \end{bmatrix} \\ \vdots & \vdots \end{bmatrix}.$$

Wenn man außerdem die Elemente von V selbst als sog. atomare f-Strukuren zuläßt, ergibt sich folgende rekursive Definition.

(11) **Definition.** Die Menge $\mathbf{FS}(F, V)$ der *f-Strukturen* über einer Merkmalsmenge F und einem Wertebereich V ist folgendermaßen gegeben:

(a) Ist $a \in V$, so ist $a \in \mathbf{FS}(F, V)$. a heißt dann eine *atomare* f-Struktur.

(b) Sind $f_i \in F$ und $s_i \in \mathbf{FS}(F, V)$, $i = 1, \ldots, n$, so ist

$$s_0 := \begin{bmatrix} f_1: & s_1 \\ \vdots & \vdots \\ f_n: & s_n \end{bmatrix} \in \mathbf{FS}(F, V),$$

wobei $f_i = f_j$ immer $s_i = s_j$ für alle $i, j = 1, \ldots, n$ impliziert. In diesem Fall spricht man von einer *komplexen* f-Struktur. Die Merkmalsbelegungen in einer komplexen f-Struktur s_0 werden als $s_0(f_i) = s_i$ oder $s_0.f_i = s_i$ für $i = 1, \ldots, n$ notiert.[28]

(c) Nichts sonst ist in $\mathbf{FS}(F, V)$.

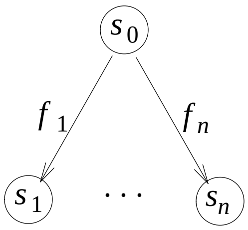
Eine weitere Interpretation und Darstellung von f-Strukturen erhält man, wenn man die Teilstrukturen aus dem rekursiven Aufbau mit den Knoten eines gerichteten Graphen identifiziert. Die Matrix (12) entspricht somit dem Graphen (13):

[27] In dieser Darstellung ist die Verwandschaft zu anderen im Bereich der KI für die Wissensrepräsentation gebräuchlichen Datenstrukturen offensichlich: vgl. *frames*, *concepts*, *units* usw. Auch in verschiedenen algorithmischen Sprachen finden sich mehr oder weniger direkte Entsprechungen: Verbunde, *records*, *structures* usw.

[28] Die erste Notation weist auf den Funktionscharakter einer f-Struktur hin; die zweite Schreibweise orientiert sich an der Analogie von f-Strukturen zu Verbund-Datenstrukturen und der hierfür üblichen Syntax von Programmiersprachen.

(12)
$$s_0 = \begin{bmatrix} f_1 : & s_1 \\ \vdots & \vdots \\ f_n : & s_n \end{bmatrix}$$

(13)



Jeder f-Struktur läßt sich auf diese Weise ein gerichteter azyklischer Graph (ein sog. DAG, *directed acyclic graph*) mit Kantenbeschriftung und einer Wurzel zuordnen, wobei die Knoten des Graphen aus $\mathbf{FS}(F, V)$ und die Kanten aus $\mathbf{FS}(F, V) \times F \times \mathbf{FS}(F, V)$ sind. Der Kante (s_i, f_j, s_k) entspricht die Merkmalsbelegung $s_i . f_j = s_k$. Da die Zuordnung eineindeutig ist, sind Merkmalsmatrizen und DAGs völlig äquivalente Darstellungen.

Allerdings entspricht nicht jeder DAG einer f-Struktur. Voraussetzung dafür ist, daß

(a) von Knoten $\in V$ keine Kanten ausgehen,

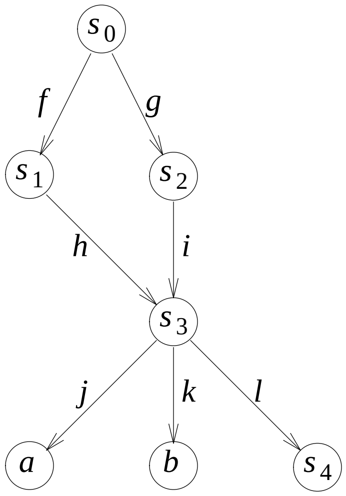
(b) Kanten, die von einem Knoten ausgehen, paarweise verschiedene Merkmale haben und

(c) der Graph eine Wurzel (einen Knoten, von dem aus alle anderen Knoten erreichbar sind) hat.

DAGs, die diese Eigenschaften haben, werden im folgenden mit den ihnen entsprechenden f-Strukturen identifiziert. Gelegentlich braucht man auch “mehrfach verwurzelte” f-Strukturen, d. h. solche, deren DAGs nicht der Bedingung (c) genügen. Diese allgemeinere Menge wird mit $\overline{\mathbf{FS}}(F, V)$ bezeichnet; trivialerweise gilt $\mathbf{FS}(F, V) \subset \overline{\mathbf{FS}}(F, V)$.

Will man die “Inhalte” einer f-Struktur bestimmen, so ist in der Regel die Matrizenschreibweise übersichtlicher. Es gibt jedoch Fälle, wo nur die Graphendarstellung eine befriedigende Wiedergabe der Struktur erlaubt, z. B. dann, wenn der DAG einer f-Struktur *kein* Baum (nicht wegeindeutig) ist, wie in folgendem Beispiel:

(14)



In solchen Fällen müssen Teilstrukturen der Merkmalsmatrix indiziert werden, so daß Identität durch gleiche Indizes ausgedrückt werden kann. Mit dieser Methode ließe sich eine zu (14) äquivalente Matrix etwa so schreiben:

(15)
$$\begin{bmatrix} f: & \begin{bmatrix} h: & \mathbf{1} \end{bmatrix} \\ g: & \begin{bmatrix} i:^{\mathbf{1}} & \begin{bmatrix} j: & a \\ k: & b \\ l: & [\ ] \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Indizes können beliebig gewählt werden, solange die Zuordnung zu Teilstrukturen konsistent (eindeutig) vorgenommen wird.

Jede Teilstruktur einer f-Struktur ist durch mindestens eine endliche Folge von Merkmalen bestimmt; umgekehrt bestimmt jede solche Folge eindeutig eine Teilstruktur, sofern alle Merkmale der Folge an den entsprechenden Stellen der f-Struktur einen definierten Wert haben. Solche Merkmalsfolgen entsprechen den Pfaden durch den entsprechenden DAG.

(16) **Definition.** Ein *Pfad* durch eine f-Struktur ist eine endliche (auch leere) Folge $\langle f_1, \ldots, f_n \rangle$ von Merkmalen $f_i \in F$. Der Wert $s.\mathbf{f}$ eines Pfades $\mathbf{f}$ angewendet auf eine f-Struktur s ist rekursiv definiert durch die Regeln:

(a) Ist $\mathbf{f} = \langle\rangle$ (der leere Pfad), so ist $s.\mathbf{f} := s$.

(b) Ist $\mathbf{f} = \langle f_1, f_2, \ldots, f_n \rangle$, $n > 0$, und $s.f_1$ definiert, so ist $s.\mathbf{f} := (s.f_1).\langle f_2, \ldots, f_n \rangle$.

(c) In allen übrigen Fällen ist $s.\mathbf{f}$ undefiniert.

Indem man die Pfade, die in einer f-Struktur definiert sind, und die ihnen zugeordneten Werte vollständig angibt, hat man eine weitere Methode (neben Merkmalsmatrizen und DAGs), mit der f-Strukturen eindeutig beschrieben und notiert werden können. Genaugenommen reicht es, wenn man dreierlei spezifiziert:

(a) Welche Pfade führen zu einer leeren f-Struktur []. Ist dieselbe leere f-Struktur über mehrere Pfade erreichbar, langt es, einen davon anzugeben.

(b) Welche Pfade führen zu atomaren Werten und welche sind dies. Führen zu einem atomaren Wert mehrere Pfade, so langt es, einen davon anzugeben.

(c) Welche Pfade führen zu gleichen Teilstrukturen (*Pfadäquivalenzen*).

Damit entsprechen der f-Struktur (14) bzw. (15) folgende Menge von Pfadspezifikationen:

(17)
$$s_0.f.h.j = a$$
$$s_0.f.h.k = b$$
$$s_0.f.h.l = []$$
$$s_0.f.h = s_0.g.i$$

3.1.2 Unifizierbarkeit und Unifikation

Um die Unifikation kompakt definieren zu können, ist es notwendig, die Menge der f-Strukturen $\mathbf{FS}(F, V)$ algebraisch zu strukturieren. Wir tun dies, indem wir zwischen den f-Strukturen eine partielle Ordnung (Halbordnung) definieren. Betrachten wir f-Strukturen als kodierte Information, so läßt sich die im folgenden formal definierte Ordnung informell so paraphrasieren: Die Struktur t “kommt nach” der Struktur s, wenn t “mehr Information” enthält als s. Dabei gibt es natürlich Fälle, wo keine Rangfolge festgestellt werden kann, weil die Inhalte nicht “vergleichbar” sind. Daher ist die Ordnung notwendigerweise nur partiell.

(18) **Definition.** Zwischen zwei f-Strukturen $s, t \in \mathbf{FS}(F, V)$ besteht die Relation $s \sqsubseteq t$, d. h. s ist *allgemeiner* als t bzw. s *subsumiert* t, wenn folgendes erfüllt ist:

(a) Für alle Pfade $\mathbf{f} \in F^*$, für die $s.\mathbf{f}$ definiert ist, ist auch $t.\mathbf{f}$ definiert.

(b) Für alle Pfade $\mathbf{f} \in F^*$, für die $s.\mathbf{f}$ atomar ist, gilt $s.\mathbf{f} = t.\mathbf{f}$.

(c) Für alle Paare von Pfaden $\mathbf{f}, \mathbf{g} \in F^*$ mit $s.\mathbf{f} = s.\mathbf{g}$, gilt auch $t.\mathbf{f} = t.\mathbf{g}$.

Gilt $t \sqsubseteq s$, so schreibt man auch $s \sqsupseteq t$ (s ist *spezieller* als t), ansonsten sind s und t nicht vergleichbar.

Auf den Nachweis, daß $\sqsubseteq$ eine Halbordnung ist, wird hier verzichtet, aber in Verbindung mit der oben diskutierten Charakterisierung von f-Strukturen durch Pfadangaben ist leicht zu sehen, daß Reflexivität, Antisymmetrie und Transitivität von $\sqsubseteq$ unmittelbar aus der Definition folgen.

Die oben angegebene informelle Paraphrasierung der Definition erklärt sich wie folgt: (a) verlangt, daß in t “mindestens” die Merkmale definiert sind, die schon in s definiert sind. (b) verlangt, daß die atomaren Werte in t mit denen in s übereinstimmen, soweit sie in s auch schon definiert waren. (c) verlangt, daß in t “mindestens” die Pfade äquivalent sind, die schon in s äquivalent sind. Merkmale, Werte und Pfadäquivalenzen sind aber genau die Arten von Information, die in einer f-Struktur enthalten sind; daher die Umschreibung, daß t mindestens soviel Information enthält wie s.

Es ist unmittelbar einleuchtend, daß die leere f-Struktur [] die allgemeinste aller komplexen f-Strukturen (Merkmalsmatrizen) ist. Nach Definition (18) ist jedoch auch $[] \sqsubseteq v$ für alle atomaren v. Das heißt, [] ist das *kleinste Element* in $\mathbf{FS}(F, S)$ bezüglich $\sqsubseteq$. Ein *größtes Element* gibt es dagegen in $\mathbf{FS}(F, V)$ nicht, da f-Strukturen beliebig “groß” werden können.

Wenn zwei f-Strukturen nicht vergleichbar sind, so kann das daran liegen, daß ihre Inhalte inkompatibel sind in dem Sinne, daß Merkmale unterschiedlich belegt sind (Bedingung (b) ist verletzt). Oder aber die beiden Strukturen haben Inhalte, die “orthogonal” zu einander sind dadurch, daß Informationen aus der einen Struktur in der anderen fehlen und umgekehrt. Dieser zweite Fall ist dadurch charakterisiert, daß man eine dritte f-Struktur finden kann, die die Inhalte der beiden anderen Strukturen zusammenfaßt. Dieser Sachverhalt läßt sich mithilfe der soeben eingeführten Halbordnung formalisieren.

(19) **Definition.** Zwei f-Strukturen $s, t \in \mathbf{FS}(F, V)$ heißen *unifizierbar*, wenn es eine Struktur $x \in \mathbf{FS}(F, V)$ gibt, mit der $s \sqsubseteq x$ und $t \sqsubseteq x$ gilt.

Die Definition läßt sich problemlos auf Mengen von (mehr als zwei) f-Strukturen verallgemeinern. Man beachte, daß Unifizierbarkeit zwar reflexiv und symmetrisch, aber *nicht* transitiv (also keine Äquivalenzrelation) ist. [] ist mit allen f-Strukturen unifizierbar, aus der Transitivität würde die paarweise Unifizierbarkeit beliebiger f-Strukturen folgen.

In der Sprache der Ordnungstheorie bedeutet Unifizierbarkeit nichts anderes als die Existenz einer *oberen Schranke* bezüglich der Halbordnung $\sqsubseteq$. Während es zu zwei inkompatiblen, also nicht unifizierbaren f-Strukturen eine solche obere Schranke nicht gibt, ist die Existenz einer *unteren Schranke* durch das kleinste Element [] immer gesichert. Für die Definition der Unifikation als algebraischer Operation ist es wichtig, daß es zu zwei (oder mehr) f-Strukturen sogar immer eine *größte untere Schranke* (ein *Infimum*) gibt.

(20) **Lemma.** Zu zwei f-Strukturen $s, t \in \mathbf{FS}(F, V)$ gibt es ein $x \in \mathbf{FS}(F, V)$ mit $x \sqsubseteq s$ und $x \sqsubseteq t$, so daß für alle $y \in \mathbf{FS}(F, V)$ mit $y \sqsubseteq s, t$ gilt: $y \sqsubseteq x$. (x ist die speziellste f-Struktur, die allgemeiner ist als s und t.)

Beweis. Ohne in die Details zu gehen, hier eine Skizze der Konstruktion des Infimums aus zwei f-Strukturen. Zu jeder der beiden f-Strukturen s und t bilde man folgende drei Mengen:

(a) die Menge der Pfade, die in der Struktur definiert sind;

(b) die Menge aller Pfad-Wert-Paare, so daß die Werte atomar sind;

(c) die Menge aller Paare von Pfaden, so daß die Pfade äquivalent sind.

Man bilde jeweils den Schnitt dieser Mengen mit ihren Pendants. Die resultierenden drei Schnittmengen bestimmen eindeutig eine f-Struktur x, von der sich leicht zeigen läßt, daß sie allgemeiner als s und t ist. Außerdem kann jede andere f-Struktur y mit dieser Eigenschaft höchstens die durch diese drei Schnittmengen gegebenen Pfadbelegungen und -äquivalenzen enthalten, d. h. y wäre allgemeiner als x.

Das Lemma wird gebraucht, um sicherzustellen, daß es zu zwei (oder mehreren) unifizierbaren f-Strukturen nicht nur Strukturen gibt, die spezieller als beide zusammen (nämlich obere Schranken) sind, sondern daß man unter allen solchen Strukturen eine als die allgemeinste auszeichnen kann. Die Existenz dieser *kleinsten oberen Schranke* (des *Supremums*) bezüglich der Halbordnung $\sqsubseteq$ folgt für unifizierbare f-Strukturen aus dem Lemma zusammen mit einem bekannten Satz der Ordnungstheorie. Dieser besagt, daß sich das Supremum als das Infimum aller oberen Schranken bilden läßt. Daher läßt sich nun definieren:

(21) **Definition.** Für zwei unifizierbare f-Strukturen $s, t \in \mathbf{FS}(F, V)$ ist das *Unifikat* $s \sqcup t$ (das Ergebnis der *Unifikation*) von s und t diejenige f-Struktur $x \in \mathbf{FS}(F, V)$, für die gilt: $s \sqsubseteq x$ und $t \sqsubseteq x$ und $x \sqsubseteq y$ für alle y mit $s, t \sqsubseteq y$. (x ist die allgemeinste f-Struktur, die von s und t

subsumiert wird.)

Eine konstruktive Definition von $s \sqcup t$ ließe sich nach dem Prinzip des Beweises von Lemma (20) geben; eine zweckmäßigere Konstruktion wird in Abschnitt 4.2.1 besprochen.

Während die Unifizierbarkeit eine totale Relation auf $\mathbf{FS}(F, V)$ darstellt, ist die Unifikation als eine partielle Operation auf dieser Menge zu verstehen. Die Eigenschaften der Kommutativität, Assoziativität und Idempotenz sind leicht nachzuweisen, außerdem fungiert [] als neutrales Element. Wie die Unifizierbarkeit läßt sich auch die Unifikation in naheliegender Weise auf Mengen von mehr als zwei f-Strukturen verallgemeinern.

3.1.3 Beispiele

Die Unifikation ist eine Operation, die die Inhalte mehrerer f-Strukturen (atomare Werte und Pfadäquivalenzen) vereinigt, soweit diese Inhalte zueinander kompatibel sind. Diese Intuition hinter der oben gegeben formalen Definition soll anhand einiger Beispiele illustriert werden.

(22)
$$\begin{bmatrix} \mathit{Farbe}: & \mathit{rot} \end{bmatrix} \sqcup \begin{bmatrix} \mathit{Form}: & \mathit{Kreis} \end{bmatrix} = \begin{bmatrix} \mathit{Farbe}: & \mathit{rot} \\ \mathit{Form}: & \mathit{Kreis} \end{bmatrix}$$

(23)
$$\begin{bmatrix} \mathit{Farbe}: & \mathit{rot} \end{bmatrix}, \begin{bmatrix} \mathit{Form}: & \mathit{Kreis} \\ \mathit{Farbe}: & \mathit{blau} \end{bmatrix} \text{ nicht unifizierbar}$$

(24)
$$\begin{bmatrix} \mathit{Form}: & \mathit{Quadrat} \\ \mathit{Länge}:^{\mathbf{1}} & [\ ] \\ \mathit{Breite}: & \mathbf{1} \end{bmatrix} \sqcup \begin{bmatrix} \mathit{Länge}: & \begin{bmatrix} \mathit{Betrag}: & 5 \end{bmatrix} \end{bmatrix}$$
$$\sqcup \begin{bmatrix} \mathit{Breite}: & \begin{bmatrix} \mathit{Einheit}: & \mathit{cm} \end{bmatrix} \end{bmatrix} = \begin{bmatrix} \mathit{Form}: & \mathit{Quadrat} \\ \mathit{Länge}:^{\mathbf{1}} & \begin{bmatrix} \mathit{Betrag}: & 5 \\ \mathit{Einheit}: & \mathit{cm} \end{bmatrix} \\ \mathit{Breite}: & \mathbf{1} \end{bmatrix}$$

3.1.4 Exkurs: Termunifikation

Im Kontext von Theorembeweisern und Logikprogrammierung (beispielsweise in der Sprache PROLOG) wird der Begriff der Unifikation mit einer Bedeutung gebraucht, die von der hier verwendeten formal abweicht. Für die Leser, denen der Unifikationsbegriff aus diesem Kontext geläufig ist, soll hier kurz der Zusammenhang zwischen beiden Formalismen erläutert werden.

Die Unterschiede sind vor allem darauf zurückzuführen, daß die algebraischen Objekte bzw. Datenstrukturen, auf denen operiert wird, keine Merkmalsmatrizen sind, sondern *Terme*, wie sie in der Prädikatenlogik verwendet werden. Diese haben die Form $f(a_1, a_2, \ldots, a_n)$, bestehend aus einem atomaren *Funktor* f und den *Argumenten* a_i, die selbst wieder Terme sind oder aber *Variable*. Ein Term mit n Argumenten hat die *Stelligkeit* n. Für $n = 0$ spricht man auch von *Atomen*. Zwei Terme unifizieren, wenn sie gleiche Stelligkeit haben, ihre Funktoren identisch sind und alle Argumente paarweise unifizieren. Eine Variable unifiziert mit allen Termen und fungiert dabei als neutrales Element, wobei allerdings Terme, die mit *derselben* Variable unifizieren auch selbst untereinander unifizieren müssen. Das Unifikat ist dann definiert als ein Term von gleicher Stelligkeit und mit gleichem Funktor wie die beiden Ausgangsterme, wobei auf den Argumentpositionen die Unifikate der entsprechenden Argumente der Ausgangsterme stehen.

Die Termunifikation läßt sich als ein Spezialfall der oben definierten Unifikation deuten, indem man einen Isomorphismus von der Menge der Terme in eine Teilmenge der f-Strukturen konstruiert. Stellt A die Menge der verwendeten Atome dar, wird jeder Term in eine f-Struktur über der Merkmalsmenge $\{Funktor, N, Arg_1, Arg_2, \ldots\}$ und dem Wertebereich $A \cup \mathbb{N}$ nach folgendem Schema abgebildet:

$$f(a_1, a_2, \ldots, a_n) \rightarrow \begin{bmatrix} N: & n \\ Funktor: & f \\ Arg_1: & a_1{}' \\ Arg_2: & a_2{}' \\ \vdots & \vdots \\ Arg_n: & a_n{}' \end{bmatrix}$$

Falls a_i ein Term ist, so wird $a_i{}'$ das Bild von a_i unter eben dieser Abbildung. Ist a_i eine Variable, so wird für $a_i{}'$ die leere f-Struktur [] eingesetzt, wobei allerdings darauf zu achten ist, daß gleiche Variablen auf identische [] abbgebildet werden. (Eine Variable, die mehrfach in einem Term auftaucht, entspricht einem DAG, der nicht wegeindeutig ist.)

Unter dieser so geschaffenen Abbildung entspricht die Termunifikation genau der Unifikation von entsprechenden f-Strukturen, d. h. die angegebene Abbildung ist tatsächlich ein Isomorphismus. Obwohl weniger offensichtlich, läßt sich auch eine Abbildung in die umgekehrter Richtung angeben, die es erlaubt, die Unifikation von f-Strukturen auf die Unifikation von Termen zurückzuführen. Im Rahmen einer Implementierung von Unifikationsgrammatiken in PROLOG wird dieses Verfahren in [Eisele1986a] besprochen.

3.2 Unifikationsgrammatiken

Der unifikationsbasierte Grammatiktyp, der im folgenden vorgestellt wird, ist als kleinster gemeinsamer Nenner einer wachsenden Menge von linguistischen Formalismen zu betrachten, die in neuerer Zeit aus der Idee entstanden sind, die sehr einfache, aber mächtige Operation der Unifikation als strukturbildendes und -beschreibendes Mittel für Sprachen einzusetzen. Typischerweise wird die Unifikation dabei mit Chomsky-artigen Grammatikformalismen kombiniert. Der hier angegebene Formalismus entspricht weitgehend dem am SRI entwickelten PATR-II-System [Shieber1983a].

Um die Verwendung von Unifikation als formales Hilfsmittel der linguistischen Theorie zu motivieren, enthält der folgende Abschnitt einige Bemerkungen zur Art und Weise, wie formale Grammatiken typischerweise in der Linguistik eingesetzt werden.

3.2.1 Grammatik und linguistische Beschreibung

Die Linguistik ist relativ rasch davon abgekommen, Sprachbeschreibung in der Form einer einzigen monolothischen Grammatik etwa eines Chomsky-Typs zu betreiben. Obwohl man aufgrund der Turingmaschinen-Äquivalenz von Chomsky-0-Grammatiken erwarten sollte, daß eine vollständige Charakterisierung einer beliebigen natürlichen Sprache mit solchen Mittel prinzipiell möglich ist, hat es sich gezeigt, daß linguistische Generalisierungen über gewissen strukturellen Phänomenen besser im Rahmen von weniger mächtigen Formalismen ausgedrückt werden können.

So lassen sich z. B. weite Teile der natürlichsprachlichen Syntax mit kontextfreien Grammatiken beschreiben, deren Möglichkeiten sich nur bezüglich einiger eng abgegrenzter Phänomene als inadäquat oder gar ganz unzureichend erweisen. So ließen sich z. B. sog. Fernabhängigkeiten (*long-distance dependencies*) durchaus im Rahmen von kontextfreien Grammatiken korrekt beschreiben, jedoch würde dies die Grammatiken ungebührlich aufblähen, und wichtige linguistische Generalisierungen kämen in einer solchen Grammatik nicht mehr zum Ausdruck.[29]

Aus diesem Grund hat schon Chomsky selbst in der *generativen Transformationsgrammatik* (TG, siehe z. B. [Radford1981a]) einem hybriden Formalismus den Vorzug gegeben. Dabei generiert eine kontextfreie Basisgrammatik eine Zwischensprache, die sog. *Tiefenstrukturen*, die dann von Strukturmanipulationsregeln, den *Transformationen*, in die eigentliche Sprache, die *Oberflächenstrukturen*, abgebildet werden. Da die Transformationsregeln sehr allgemein sind, erreicht der Formalismus insgesamt die Mächtigkeit von Chomsky-0-Sprachen. Phänomene, die sinnvollerweise als Chomsky-2-artig beschrieben werden können (und aus Gründen der Beschreibungsökonomie auch so behandelt werden sollten), sind in der Basisgrammatik formalisiert, während z. B. Fernabhängigkeiten erst durch Transformationen generiert werden.

Obwohl die TG lange Zeit als "Stand der Technik" in der Linguistik galt (und für viele heute noch gilt), gibt es verschiedene Gründe, nach anderen Methoden der sprachlichen Beschreibung zu suchen. Zusätzlich zu den beiden oben genannten Komponenten, Basisgrammatik und Transformationsregeln, entstanden nämlich mit der Zeit immer weitere Zusatzkomponenten, die die Beschreibungsadäquatheit des Gesamtapparats sicherstellen sollten. Resultat war ein Komplex von mehreren sequentiell zusammenarbeitenden Teilapparaten, die generativ die Sprache charakterisieren. Die streng sequentielle Struktur des Formalismus läßt diesen als Grundlage für ein effizientes und psychologisch plausibles Modell der natürlichen Sprachverarbeitung ungeeignet erscheinen.[30] Die explizit *generative* Form der Beschreibung wirft zudem zusätzliche Komplexitätsprobleme für die Sprach*analyse* auf. Zu diesen Kritikpunkten kommen metatheoretische Überlegungen, die den gesammten Formalismus als zu wenig geschlossen und elegant betrachten.

3.2.2 Grammatiken und Unifikation

Auch Unifikationsgrammatiken haben eine zweischichtige formale Struktur, jedoch von wesentlich geschlossenerer Art. Basis ist auch hier eine Chomsky-Grammatik, die eine Obermenge der zu beschreibenden Sprache dadurch erzeugt, daß lediglich die Konstituentenstruktur (im folgenden kurz *c-Struktur*), d. h. der Syntaxbaum der Sätze der Sprache charakterisiert wird. In üblichen Chomsky-Grammatiken ist die c-Struktur das einzige formale Mittel zur

---

[29] Mit *Fernabhängigkeit* bezeichnet man das Phänomen, daß z. B. in Fragesätzen topikalisierte Satzglieder von der Stelle im Satz, an der sie "eigentlich" stehen und von der sie grammatikalisch abhängen, beliebig weit entfernt sein können. Vgl.

Wohin hat Peter gesagt, daß Maria gesagt hat, daß Paul gesagt hat, daß . . . , daß Klaus geht __.

Abgesehen von seiner Position verhält sich *Wohin* so, als ob es an der markierten Stelle im Satz als Ergänzung zu *geht* generiert worden wäre.

[30] Vgl. die Diskussion von sequentiellen vs. interaktiven Schnittstellen in Abschnitt 2.1.3.

Beschreibung des Sprachschatzes: Die Produktionen einer Grammatik definieren die Menge der möglichen c-Strukturen (Syntaxbäume) und diese wiederum bestimmen implizit die Menge der Sätze der Sprache (als Folge der terminalen Knoten des Syntaxbaums).

In Unifikationsgrammatiken ordnet man zusätzlich jedem Satz eine *funktionale Struktur* oder *Merkmalsstruktur* (*feature structure*, *functional structure*, im folgenden kurz *f-Struktur*) zu. Während die c-Struktur ein Baum ist, ist die f-Strukur ein gerichteter azyklische Graph mit Kantenbenennung wie in Abschnitt 3.1.1 eingeführt, allerdings ohne Wurzel. Beide Strukturen sind durch eine Abbildung verbunden, die jedem c-Strukturknoten (kurz: c-Knoten) einen f-Strukturknoten (f-Knoten) zuordnet. Diese Abbildung muß nicht injektiv sein und ist in der Regel auch nicht surjektiv, d. h. es gibt in der f-Struktur im allgemeinen eine große Anzahl von Knoten, die nicht über die Zuordnung einem c-Knoten direkt entsprechen.[31]

Betrachtet man zu einem c-Knoten den zugeordneten f-Knoten und alle weiteren f-Knoten, die von diesem erreichbar sind, so erhält man einen Untergraph der f-Struktur mit Wurzel. Diesen Untergraph bezeichnet man als *die dem c-Knoten zugeordnete f-Struktur*. Unter der *f-Struktur des Satzes* versteht man gelegentlich auch nur den Untergraphen, der der Wurzel der c-Struktur zugeordnet ist.

Man kann also die f-Struktur als eine Attributierung der c-Struktur mit Merkmalsmatrizen betrachten, wobei verschiedene Attributwerte gleiche Teilstrukturen besitzen. Da c- und f-Struktur parallel definiert und aufgebaut werden, macht es keinen Sinn, eine Unterscheidung zwischen “Oberflächenstruktur” und “Tiefenstruktur” zu treffen, da beide gleichermaßen “an der Oberfläche” liegen.

Die Charakterisierung der möglichen f-Strukturen wird in die Regeln der Grammatik aufgenommen, die dafür einen entsprechenden Zusatz zu den üblichen Textersetzungsanweisungen (Produktionen) erhalten. Jede Grammatikregel besteht aus einer Produktionsregel entsprechend dem zugrundeliegenden Chomsky-Typ (in der Regel kontextfrei) und einer Menge von *Gleichungen* über den f-Strukturen, die den beteiligten c-Knoten zugeordnet sind.

Eine Regel, wie sie in einer einfachen linguistisch motivierten Grammatik vorkommen könnte, wäre z. B.[32]

(25) $$\begin{array}{l} S \rightarrow NP\ \ VP \\ \qquad S.head = VP.head \\ \qquad S.head.subj = NP.head \end{array}$$

Dabei stehen S und NP in den Gleichungen als Abkürzung für die f-Strukturen, die dem S- bzw. dem NP-Knoten in der c-Struktur zugeordnet ist.[33] Die Regel (25) wird in deklarativer Form als

[31] Strenggenommen muß zwischen f-Strukturen im Sinne von Abschnitt 3.1.1 und f-Strukturen als Teil einer zu einer c-Struktur gehörigen grammatischen Beschreibung unterschieden werden. Die genaue Bedeutung wird jedoch immer aus dem Zusammenhang hervorgehen und deshalb im folgenden nicht explizit gemacht.

[32] In diesem und den folgenden Beispielem wurden die Namen für Kategorien, Merkmale und Werte nicht ganz willkürlich gewählt, obwohl der Leser ohne linguistischen Hintergrund sie als willkürlich betrachten kann. Es wurden Beispiele gewählt, die bei Kenntnis einschlägiger linguistischer Theorie und Terminologie sinnvolle Assoziationen erzeugen sollen.

[33] Wenn ein nichtterminales Symbol in der Produktion mehrfach vorkommt, ist diese abgekürzte Schreibweise nicht mehr eindeutig; man muß durch Indizierung o. ä. die Eindeutigkeit des Bezuges herstellten.

eine Bedingung an die möglichen c- bzw. f-Strukturen von Sätzen des Sprachschatzes gelesen: Wenn in der c-Struktur ein Vaterknoten von der Kategorie S zwei Sohnknoten mit Kategorien NP und VP (in dieser Reihenfolge von links nach rechts) hat, so müssen die f-Strukturteile, die auf gegenüberliegenden Seiten eines Gleichheitszeichens stehen, identisch sein (denselben f-Knoten darstellen).

Die Bezeichnung "Unifikationsgrammatik" ist an dieser Stelle noch unmotiviert. Sie wird erst verständlich, wenn man die Regeln einer Grammatik in prozeduraler Form interpretiert, um einen Satz zu generieren oder zu analysieren.

Eine mögliche Strategie zum Analysieren eines Satzes aus dem Sprachschatz geht so vor, daß c- und f-Struktur inkrementell anhand der Grammatikregeln aufgebaut werden. Regel (25) liest sich jetzt so: Liegen zwei c-Strukuren mit Wurzelknoten NP bzw. VP vor, so erweitere die c-Struktur um einen gemeinsamen Vaterknoten S und ordnete diesem die f-Struktur

$$\left[head: \left[subj: [\ ] \right] \right]$$

zu, wobei die in den Gleichungen angegebenen Unterknoten identifiziert werden. Weil bei diesen Identifikationen von f-Knoten aber keine inkonsistenten f-Strukturen entstehen dürfen, müssen die entsprechenden Teilstrukturen kompatibel in ihrer Merkmalsbelegung sein, damit die Regel überhaupt anwendbar ist. Mit anderen Worten: gleichgesetzte Teilstrukturen müssen unifizierbar sein. Die Gleichungen in einer Grammatikregel sind also als *Unifikationsgleichungen* zu interpretieren und stellen eine Bedingung an die Anwendbarkeit der Regel dar.

Leider reicht die Unifizierbarkeit von Teilstrukturen entsprechend den Gleichungen einer Regel noch nicht aus, um die Anwendbarkeit dieser Regel zu gewährleisten. Dies liegt daran, daß die Unifizierbarkeitsrelation nicht transitiv ist, wohl aber die Identifikation von f-Knoten. So impliziert Regel (25) die Identifikation von $VP.head.subj$ mit $NP.head$. Weitere implizite Identifikationen können durch die Anwendung anderer Regeln resultieren. Gefordert ist also gewissermaßen die "transitive Hülle" aller Unifikationsgleichungen.

Man kann die Kompatibilität aller beteiligten f-Strukturen prüfen, indem man nicht nur *Unifizierbarkeit* testet, sondern die *Unifikation* selbst jeweils ausführt. Dabei werden die Gleichungen der Reihe nach abgearbeitet und versucht, gleichgesetzte f-Strukturen zu unifizieren. Ist dies in irgendeinem Fall nicht möglich, so ist die Regel nicht anwendbar. Andernfalls werden die unifizierten Teilstrukturen durch das Unifikat ersetzt, so daß bei der weiteren Abarbeitung nicht mehr die ursprünglichen f-Strukturen betrachtet werden, sondern deren Unifikat.

Man nennt dieses Verfahren *destruktive Unifikation*, da hierbei aus zwei Datenstrukturen nicht eine dritte erzeugt wird, sondern stattdessen die Datenstrukturen selbst als Seiteneffekt modifiziert werden. Die Identifikation von f-Knoten wird dabei miterledigt, da ja nach der destruktiven Unifikation die beiden gleichgesetzten Teilstrukturen durch eine einzige neue Struktur, nämlich das Unifikat, erzetzt sind. Arbeitet man auf den modifizierten f-Strukturen weiter, werden automatisch auch die impliziten Gleichheiten berücksichtigt, die durch die Anwendung weiterer Regeln entstehen.

Zusammenfassend läßt sich sagen, daß die Gleichungen in einer Grammatikregel deklarativ als Bedingung an die möglichen f-Strukturen zu interpretieren sind, in prozeduraler Lesart aber als Anweisungen zur destruktiven Unifikation von f-Teilstrukturen verstanden werden können. Dadurch kann inkrementell die gesamte f-Struktur parallel zur c-Struktur aufgebaut werden, und zwar entweder wie oben beschrieben *bottom-up* zur Analyse, oder in analoger Weise *top-down* zur Generierung von Sätzen.[34]

3.2.3 Von Chomsky-Grammatiken zu Unifikationsgrammatiken

Bevor die hier informell vorgestellten Elemente einer Unifikationsgrammatik exakt definiert werden, soll noch eine formale Vereinfachung vorgenommen werden. Diese betrifft die Form der Produktionen der Grammatik, die ja die Bedingungen an die c-Struktur der Sätze kodieren. Wenn man die Knoten des Syntaxbaumes zunächst unabhängig von ihrer Kategorie betrachtet, so beinhaltet eine Produktion folgende Bedingungen:

(a) Gewisse (Vater-)Knoten dominieren gewisse andere (Sohn-)Knoten. Hierdurch wird die sog. ID-Relation (*immediate dominance*) zwischen den Knoten bestimmt.

(b) Die Sohnknoten steht in einer bestimmten Reihenfolge. Dies ist die sog. LP-Relation (*linear precedence*).

(c) Vater- und Sohnknoten sind gewisse Kategorien (nichtterminale Symbole) bzw. Lexeme (terminale Symbole) zugewiesen.

Auf (c) bezogen ist dann eine implizite Bedingung an die gesamte c-Struktur:

(d) Zwei Produktionen, die auf einem Knoten operieren (in der einen als Sohnknoten, in der anderen als Vaterknoten) müssen diesem Knoten dieselbe Kategorie zuweisen.

Diese Formulierung einer Produktion als Menge von Bedingungen an die c-Struktur soll nun folgendes nahelegen: Die Bedingungen (c) über die Kategorie (oder das terminale Zeichen) eines Knotens läßt sich auch in die f-Struktur aufnehmen, die dem Knoten zugeordnet ist. Sie taucht dann in Form von Unifikationsgleichungen zu den entsprechenden Produktion wieder auf. Die Bedingung (d) an die Kongruenz von Kategorien, die von zwei aneinander anschließenden Produktionen für einen Knoten erwartet werden, ist dann lediglich ein Spezialfall der globalen Regel, daß die Unifikationsgleichungen aller angewendeten Produktionen simultan erfüllt sein müssen.

Per Konvention kodiert man die Kategorie eines Knotens als Wert eines speziellen Merkmals *cat* in der f-Struktur, so daß die oben angegebenen Regel (25) nun folgende Form annimmt:

[34] Der Vollständigkeit halber sei erwähnt, daß auch eine *top-down*-Analyse bzw. eine *bottom-up*-Generierung denkbar ist, jedoch erscheinen diese Verfahren eher “unnatürlich” (vgl. [Kay1980a]).

(26) $X_1 \rightarrow X_2\, X_3$

$X_1.cat = S$
$X_2.cat = NP$
$X_3.cat = VP$
$X_1.head = X_3.head$
$X_1.head.subj = X_2.head$

Die Variablen X_i haben jetzt nur noch die Funktion, die Zuordnung von Knoten der c-Struktur zu Knoten der f-Struktur (Merkmalsmatrizen) vorzunehmen.[35]

Formal ist nun eine Unifikationsgrammatik und ihr Sprachschatz definiert wie folgt:

(27) **Definition.** Eine *unifikationsbasierte Grammatik* $G = (F, V, R, f_S, S, f_T, T)$ ist gegeben durch ein Menge F von Merkmalen, einen Wertebereich V, einer Menge R von *Regeln* (der unten angegeben Form), dem *Startmerkmal* $f_S \in F$, dem *Startsymbol* $S \in V$, dem *terminalen Merkmal* $f_T \in F$, und der Teilmenge $T \subset V$ der *terminalen Symbole*.

Eine Zeichenfolge $x \in T^*$ ist ein *Satz* des durch G erzeugten *Sprachschatzes* $L(G)$, wenn folgendes gilt:

(a) Es gibt eine *c-Struktur*: einen Baum über einer Menge $\mathbf{C}$ von Knoten, eine *f-Struktur*: einen gerichteten azyklischen Graphen aus $\overline{\mathbf{FS}}(F, V)$ mit Knotenmenge $\mathbf{F}$, und eine Abbildung $f : \mathbf{C} \rightarrow \mathbf{F}$.

(b) Für alle nichtterminalen Knoten $c_0 \in \mathbf{C}$ mit Söhnen $c_1, c_2, \ldots, c_n \in \mathbf{C}$ gibt es eine Regel $r \in R$ der Form

$$r = \begin{cases} X_0 \rightarrow X_1\, X_2\, \cdots\, X_n \\ \qquad E_1 \\ \qquad \vdots \\ \qquad E_k \end{cases},$$

so daß alle Gleichungen E_i erfüllt sind, wenn man jedes X_j ersetzt durch die dem Knoten c_j zugeordnete f-Struktur $f(c_j) \in \mathbf{F}$. Die E_i sind dabei Gleichungen in den Variablen X_j, den Werten aus V und den Pfaden aus F^*.

(c) Für die Wurzel $w \in \mathbf{C}$ der c-Struktur gilt $f(w).f_S = S$.

(d) Wenn man von allen terminalen Knoten $t_i \in \mathbf{C}$ in der Reihenfolge ihres Auftretens von links nach rechts in der c-Struktur den Wert $f(t_i).f_T$ nimmt (soweit diese definiert und aus T sind), so ergibt sich die Zeichenfolge x.

Zu dieser Definition einige Bemerkungen: Ausgezeichnete Elemente aus V und entsprechende Merkmale aus F übernehmen die Rolle des Startsymbols bzw. der terminalen Symbole in Chomsky-Grammatiken. Punkt (d) definiert, wie aus einer c- und zugehörigen f-Struktur die

[35] Natürlich sind diese Variablen als “lokale Variable” zu verstehen, deren “Bindungsbereich” nur jeweils eine Regel umfaßt.

terminale Zeichenfolge abzuleiten ist. Hierfür wären mehrere Alternativen denkbar, z. B. daß die terminale Zeichenfolge nicht nur aus den terminalen Knoten, sondern aus allen Knoten der c-Struktur durch eine Tiefentraversierung von links nach rechts gewonnen wird.

In der Praxis ist allerdings die hier angegebene Definition sogar noch zu allgemein. In der Regel richtet man die Grammatik so ein, daß jeder Knoten ein *cat*-Merkmal hat, wobei ein Wert aus T einen terminalen Knoten im Sinne einer Chomsky-Grammatik impliziert. Der terminale Charakter dieser Knoten kann dadurch sichergestellt werden, daß *cat*-Werte $\in T$ nur auf rechten Seiten von Grammatikregeln auftauchen.

Eine weitere Konvention vereinfacht die Aufschreibung und Strukturierung von Regeln: In gängigen linguistischen Analysen benötigt man für die terminalen Knoten der c-Struktur, die den Lexemen entsprechen, keine eigenen f-Strukturen, da die zugehörige Information direkt in der f-Struktur des dominierenden (nichtterminalen) Knotens kodiert werden kann. Dadurch ist es möglich, in die rechten Seiten von Produktionen die terminalen Symbole unmittelbar hinzuschreiben, ohne für sie jeweils eine eigene f-Strukturvariable einzuführen. Eine typische terminale Regel würde z. B. folgendermaßen aussehen

(28)
$$\begin{aligned} & X_1 \to X_2 \\ & \quad X_1.cat = N \\ & \quad X_2.cat = Grammatik \\ & \quad X_1.head.gen = fem \\ & \quad X_1.head.num = sing \end{aligned}$$

und könnte kürzer so geschrieben werden:

(29)
$$\begin{aligned} & X \to Grammatik \\ & \quad X.cat = N \\ & \quad X.head.gen = fem \\ & \quad X.head.num = sing \end{aligned}$$

Eine weitere Ersparnis an Schreibarbeit – bei besserer Lesbarkeit – läßt sich erreichen, wenn man die *cat*-Werte allgemein nicht explizit in Form von Unifikationsgleichungen angibt, sondern sie anstelle der X-Variablen direkt zur Bezeichnung der c-Strukturknoten verwendet. Regel (29) nimmt dann folgende Form an:

(30)
$$\begin{aligned} & N \to Grammatik \\ & \quad N.head.gen = fem \\ & \quad N.head.num = sing \end{aligned}$$

Auf notationeller Ebene ist man damit zum Ausgangspunkt der Überlegungen, den Chomsky-Grammatiken mit Attributierung, zurückgekehrt (vgl. Regel (25)), jedoch sollte klargestellt werden, daß es sich hierbei lediglich um eine abgekürzte Schreibweise von Unifikationsgleichungen für das Merkmal *cat* handelt. A priori haben die c-Strukturknoten in einer Unifikationsgrammatik keine “Kategorien”, und *cat* ist formal ein Merkmal wie alle anderen.

3.3 Generierung in Unifikationsgrammatiken

3.3.1 Semantische Repräsentation in Unifikationsgrammatiken

Innerhalb der linguistischen Sprachbeschreibung werden Unifikationsgrammatiken in der Regel auf der Ebene der Syntax und des Lexikons eingesetzt, d. h. sie charakterisieren die möglichen Folgen von Lexemen und deren Strukturen, wobei letztere zweischichtig sind und jeweils aus einer c- und einer zugeordneten f-Struktur bestehen. Die Zweischichtigkeit der Strukturbeschreibung hat als formale Motivation in erster Linie die gegenüber Chomsky-2-Grammatiken erhöhte Mächtigkeit.

Aus linguistischer Sicht sind Unifikationsgrammatiken besonders geeignet, um Phänomene zu beschreiben, deren kompakte Charakterisierung eine *Abstraktion* von der c-Struktur erfordern. Genau diese Abstraktion läßt sich in der f-Struktur repräsentieren.[36] Auch die Semantik eines Satzes läßt sich als eine Abstraktion (auf sehr hohem Niveau) seiner c-Struktur auffassen, so daß es naheliegt, zusätzlich zu der syntaktischen auch eine semantische Beschreibung in die f-Struktur aufzunehmen. Dies ist auch deshalb sinnvoll, weil bestimmte syntaktische Bedingungen am einfachsten in Bezug auf bestimmte Aspekte der zugrundeliegenden Semantik formuliert werden können. In der *Lexical-Funktional Grammar* (LFG), einem erweiterten unifikationsbasierten Grammatikformalismus, ist ein solcher Teilaspekt (die Prädikat-Argument-Struktur von Sätzen) zum festen Bestandteil des Formalismus gemacht worden [Kaplan1982a].

Allerdings wird die Semantik, soweit sie sich in ein und demselben Formalismus wie die Syntax beschreiben läßt, immer eine relativ große strukturelle Ähnlichkeit zu dieser Syntax behalten. Komplexe Abbildungen von beliebigen semantischen Repräsentationen in die syntaktische Form sind aufgrund der Einfachheit des Formalismus sicherlich nicht sinnvoll.

Stattdessen sind f-Strukturen in einem mehrstufigen System semantischer und pragmatischer Repräsentationen als unterste Stufe der Beschreibung denkbar, auf der in schon relativ äußerungsnaher Form der Inhalt des zu generierenden Satzes repräsentiert ist. Ein Beispiel für eine sehr weitgehende Integration von Semantik in den LFG-Formalismus findet sich in [Erben1987a].

Ist eine Grammatik gegeben, deren Regeln die Korrelation zwischen Syntax und Semantik korrekt und vollständig charakterisieren, so würde das Ergebnis einer Analyse eines Satzes mithilfe dieser Grammatik nicht nur seine syntaktische Struktur, sondern auch die korrespondierende semantische Struktur (kodiert in der f-Struktur) liefern.

Als Umkehrung dieses Prozesses ist eine Generierung denkbar, die zu einer als f-Struktur angegeben Semantik die Sätze erzeugt, die dieser Semantik entsprechen. Hier wäre die Semantik also nicht Ergebnis (oder Abfallprodukt) der syntaktischen Analyse, sondern fungiert als

---

[36] Diese Motivation ist besonders im Hinblick auf die Beschreibung sprachübergreifender Universalien relevant. Obwohl die möglichen c-Strukturen von verschiedenen Sprachen sehr unterschiedlich sein können, ist es oft möglich, unabhängig von der jeweiligen Sprache sog. funktionale Strukturen zu entdecken. Ein Beispiel ist die Unterscheidung von Satzgliedern in Subjekt und Objekt, die sich unabhängig davon treffen läßt, ob diese Funktionen durch die Position (im Englischen) oder die Flektion (im Deutschen) der Satzglieder definiert sind.

Restriktion des Generierungsprozesses derart, daß ein produzierter Satz nicht nur den syntaktischen Regeln der Sprache genügen, sondern außerdem zu der vorgegebenen Semantik kompatibel sein muß. Die Kompatibilität zwischen f-Strukturen läßt sich formal durch deren Unifizierbarkeit ausdrücken, woraus sich eine natürliche Einbettung des hier informell beschriebenen Verfahrens in den Formalismus ergibt. Die in Abschnitt 2.3 besprochenen Systeme von Appelt und McKeown, die sich für die Satzproduktion der Functional Unification Grammar (FUG) bedienen, verfahren im wesentlichen nach diesem Prinzip.

3.3.2 f-Strukturen als Spezifikation

Das Prinzip der Äußerungsspezifikation durch f-Strukturen läßt sich nun leicht im Rahmen des im letzten Abschnitt definierten Grammatikformalismus präzisieren:

(31) **Definition.** Sei $G=(F,V,R,f_S,S,f_T,T)$ eine Unifikationsgrammatik und $s_0 \in \mathbf{FS}(F,V)$ eine f-Struktur und $x \in L(G)$ eine Satz aus dem Sprachschatz von G. Dann *genügt* x *der Spezifikation* s_0 genau dann, wenn es gemäß Definition (27) zu x wenigstens eine c-Struktur **C** und eine f-Struktur **F** mit zugehöriger Abbildung $f:\mathbf{C}\rightarrow\mathbf{F}$ gibt für die gilt:

Die der Wurzel w von **C** zugeordnete f-Struktur $f(w)$ unifiziert mit s_0.

Eine Satzspezifikation in Form einer f-Struktur s_0 ist damit eine zusätzliche Randbedingung für f-Strukturen (und mittelbar auch für die c-Strukturen) der möglichen Sätze derart, daß alle Grammatikregeln, die zur Generierung eines Satzes beitragen, mit der in s_0 vorgegebenen Information kompatibel sein müssen. Kompatibilität ist dabei im Sinne der Erläuterungen zu Definition (27) zu verstehen.

Die Kompatibilität eines Satzes x zu einer Spezifikation s_0 läßt sich jedoch auch ohne die Bedingung aus Definition (31) beschreiben, wie der folgende Satz zeigt:

(32) **Satz.** Zu jeder Unifikationsgrammatik $G=(F,V,R,f_S,S,f_T,T)$ und jeder Spezifikation $s_0 \in \mathbf{FS}(F,V)$ gibt es eine Grammatik G_0, so daß $x \in L(G_0)$ dann und nur dann, wenn $x \in L(G)$ und x der Spezifikation s_0 genügt.

Beweis. Setze

$$\begin{aligned} G_0 = (\, & F \cup \{f_{S_0}\}, \\ & V \cup \{S_0, \bar{S}_0\}, \\ & R' \cup \{r_0\}, \\ & f_{S_0}, S_0, \\ & f_T, T\,) \end{aligned}$$

mit einem neuen Startmerkmal f_{S_0}, einem neuen Startsymbol $S_0 \notin V$, einem weiteren zusätzlichen Symbol $\bar{S}_0$ und einer zusätzlichen Regel der Form

$$r_0 = \begin{cases} X_0 \to X_1 \\ \qquad X_0.f_{S_0} = S_0 \\ \qquad X_1.f_{S_0} = \bar{S}_0 \\ \qquad X_1.f_S = S \\ \qquad E_1, \ldots, E_{k_0} \end{cases} .$$

Dabei sind E_i Gleichungen, die für X_1 genau die atomaren Werte und die Pfadäquivalenzen fordert, die in s_0 enthalten sind, d. h. k_0 ist gerade die Anzahl der Pfade in s_0, für die atomare Werte definiert sind, plus der Anzahl der Paare von Pfaden in s_0, die auf dieselbe komplexe Teilstruktur führen.

Die Regeln in R' entstehen aus denen in R durch Hinzufügen einiger Gleichungen. Eine Regel

$$r = \left\{ \begin{matrix} X_0 \to X_1 \, X_2 \, \cdots \, X_n \\ E_1, \ldots, E_k \end{matrix} \right\} \in R$$

wird erweitert zu

$$r' = \left\{ \begin{matrix} X_0 \to X_1 \, X_2 \, \cdots \, X_n \\ E_1, \ldots, E_k \\ X_0.f_{S_0} = \bar{S}_0 \\ \vdots \\ X_n.f_{S_0} = \bar{S}_0 \end{matrix} \right\} \in R' .$$

Ist $X \in L(G_0)$, so gilt aufgrund der Bedingungen an das Startmerkmal f_{S_0} für die c-Struktur folgendes: Alle Knoten außer der Wurzel sind durch Regeln aus R' entstanden, d. h. sie und die zugehörigen f-Strukturen genügen auch der ursprünglichen Regelmenge R, da ja lediglich neue Bedingungen hinzugekommen sind. Die Wurzel der c-Struktur muß durch einmalige Anwendung der Regel r_0 entstanden sein, und das bedeutet unter anderem, daß der erste und einzige Sohnknoten als Wert des alten Startmerkmals f_S das alte Startsymbol S hat. Insgesamt gilt also, daß die c-Struktur unterhalb der Wurzel der alten Grammatik G genügt, und da die terminalen Knoten der c-Struktur auch die terminalen Knoten dieses Unterbaums sind, gilt $x \in L(G)$.

Außerdem folgt aus der Anwendung von r_0 auf die Wurzel, daß für die f-Struktur des Unterbaums die aus s_0 abgeleiteten Gleichungen gelten. Daß heißt aber, daß die f-Struktur von x bezüglich G der Spezifikation s_0 genügt.

Umgekehrt sei nun $x \in L(G)$ und der Spezifikation s_0 gemäß. Dann kann man die c- bzw. die f-Struktur von x so erweitern, daß die erweiterten Strukturen der neuen Grammatik G_0 genügen. Die c-Struktur bekommt eine neue Wurzel als Vaterknoten der alten. Die zugeordnete f-Struktur ist eine Matrix, die genau die in r_0 für X_0 geforderten Merkmale hat. Die alte f-Struktur von x genügt den übrigen Bedingungen von r_0 nach

Voraussetzung. Die zusätzlichen Bedingungen der Regeln in R' fordern den Wert $\bar{S}_0$ für das Merkmal f_{S_0} in allen f-Strukturen von x außer der zur neuen Wurzel gehörigen. Da dieses Merkmal aber in der alten Grammatik G gar nicht vorkam, ist die f-Struktur diesbezüglich völlig unspezifiziert, d. h. wir können es nach Belieben definieren, ohne die Gültigkeit der alten Unifikationsgleichungen zu beeinträchtigen.

Damit ist also auch $x \in L(G_0)$ und der Satz insgesamt bewiesen.

Satz (32) zeigt, wie man eine Spezifikation in der Praxis am einfachsten in die Generierung einfließen lassen kann, ohne dafür den Produktionsalgorithmus für die Generierung beliebiger Sätze modifizieren zu müssen. Es genügt, die Grammatik zu *augmentieren*, d. h. um eine neue Startregel zu erweitern, und die Spezikation in Unifikationsgleichungen zu dieser Regel zu übersetzen.

Die Einführung eines neuen Startmerkmals f_{S_0} ist eine beweistechnische Notwendigkeit, um sicherzustellen, daß die neue Startregel genau einmal an der Wurzel der c-Struktur angewendet wird und nicht in die übrige Struktur eingeht. In der Praxis kann darauf meist verzichtet werden, denn üblicherweise sieht die Grammatik ein *cat*-Merkmal vor, das auch Startmerkmal ist und in jeder Regel eindeutig für alle beteiligten c-Strukturknoten festgelegt ist. *cat* kann dann als Startmerkmal der augmentierten Grammatik übernommen werden und die Regeln aus R brauchen nicht modifiziert zu werden.

3.3.3 Ein Beispiel

Die Spezifikation durch f-Strukturen und die entsprechende Augmentierung der Grammatik soll nun anhand eines kleinen Beispiels verdeutlicht werden. Zu diesem Zweck wurde wieder versucht, linguistisch “sinnvolle”, wenn auch möglichst einfache und daher notwendigerweise naive Grammatikregeln zu verwenden.

3.3.3.1 Die Grammatik

Die Sätze, die mit der folgenden Grammatik beschrieben werden sollen, bestehen alle aus einem Verb (V) und zwei Nominalphrasen (NP), die Subjekt und Objekt zu dem Verb bilden. Wir nehmen weiterhin an, daß Verb und Objekt zusammen eine sog. Verbalphrase (VP) als Teilstruktur bilden. Wie üblich hat jeder c-Strukturknoten ein *cat*-Merkmal, um seine syntaktische Kategorie zu identifizieren, und ein *head*-Merkmal, unter dem alle weiteren grammatischen Eigenschaften dieses Knotens kodiert sind (vgl. (25)). Die *head*-Strukturen von Subjekt bzw. Objekt werden in der *head*-Struktur des Verbs unter den Merkmalen *subj* bzw. *obj* abgelegt. Damit lauten die ersten beiden Regeln für S bzw. VP folgendermaßen:

(R_S) $\quad S \rightarrow NP\ VP$
$$S.head = VP.head$$
$$S.head.subj = NP.head$$

(R_{VP}) $\quad VP \rightarrow V\ NP$
$$VP.head = V.head$$
$$VP.head.obj = NP.head$$

Man beachte, daß die *head*-Struktur des Verbs durch die angegebenen Unifikationen letztlich zur *head*-Struktur des ganzen Satzes wird.

Der Einfachheit halber nehmen wir an, daß *NP* und *V* jeweils genau durch ein Lexem realisiert werden, d. h. wir müssen noch terminale Regeln für diese Kategorien angeben. An dieser Stelle spätestens muß auch die Repräsentation der Semantik festgelegt werden. Als eine erste Näherung für die Darstellung natürlichsprachlicher Semantik kann die Prädikatenlogik angesehen werden, die umso unproblematischer ist, je enger der betrachtete Sprachausschnitt gewählt wird. Im hier behandelten Beispiel, wird das Verb auf ein zweistelliges Prädikat abgebildet und Subjekt bzw. Objekt auf dessen Argumente. Auf diese Weise entspricht ein Satz wie

(33) *Peter liebt Maria*

der prädikatenlogischen Formel

(34) **lieben(peter, maria)**

Dabei sind **peter** und **maria** Konstanten, die die durch *Peter* bzw. *Maria* referierten Individuen bezeichnen.

Um die prädikatenlogische Repräsentation in die Grammatik integrieren zu können, werden Formeln wie (34) auf naheliegende Weise in eine entsprechende f-Struktur abgebildet:

(35)
$$\begin{bmatrix} pred: & \mathbf{lieben} \\ arg_1: & \mathbf{peter} \\ arg_2: & \mathbf{maria} \end{bmatrix}$$

Die Semantik eines Satzes soll in dieser Form unter dem Merkmal *sem* in der *head*-Struktur kodiert sein. Da die Semantik des Satzes kompositional verstanden wird, d. h. aus der Semantik seiner Teile aufgebaut ist, bekommen auch Subjekt- und Objekt-*head* das Untermerkmal *sem*.

Die terminalen Regeln (oder Lexikoneinträge), aus denen sich zusammen mit den Regeln (R_S) und (R_{VP}) der Satz (33) erzeugen läßt, lauten jetzt

(R_{Peter}) $NP \rightarrow Peter$
$NP.head.sem = \mathbf{peter}$

(R_{Maria}) $NP \rightarrow Maria$
$NP.head.sem = \mathbf{maria}$

(R_{liebt}) $V \rightarrow liebt$
$V.head.sem.pred = \mathbf{lieben}$
$V.head.sem.arg_1 = V.head.subj.sem$
$V.head.sem.arg_2 = V.head.obj.sem$

Es ist zu bemerken, daß die Semantikkomponente der hier angegeben kleine Grammatik ausschließlich in den lexikalischen Regeln verankert ist. Insbesondere ist auch die Beziehung zwischen Syntax und Semantik dort festgelegt, was dazu führt, daß wesentliche Änderungen der generativen Kapazität allein durch Modifikation des Lexikons erreicht werden können. Dieser "lexikalische" Charakter ist eine typische Eigenschaft von Unifikationsgrammatiken und resultiert aus der konsequenten Ausnutzung der Unifikation als Beschreibungsmittel.

Am Beispiel von (R_{liebt}) sieht man, wie eine Gleichung, die nur auf lokale Größen Bezug nimmt, letztlich mittels der Unifikationen, die aus den übrigen Regeln resultieren, die Struktur des ganzen Satzes beeinflußt (deklarativ gesprochen: eine Bedingung an diese globale Struktur darstellt). Eine Folge dieser Eigenschaft ist die Möglichkeit, eine Art "Passivkonstruktion" nur durch Hinzufügung entsprechender Verblexeme zu realisieren. Mit Passivkonstruktionen sind in diesem Zusammenhang Sätze gemeint, in denen die Zuordnung von Subjekt und Objekt zu den Argumentpositionen des Verbs genau umgekehrt zum obigen Beispiel (33) vorgenommen ist, wie etwa in

(36) *Maria wird geliebt von Peter* .

Wenn wir unter Vernachlässigung aller linguistischen Intuition und der Einfachheit halber annehmen, daß *wird geliebt von* hier ein einziges, komplexes Lexem darstellt, dann sollte die Grammatik (36) dieselbe Semantik zuweisen wie (33), und auch die syntaktische Struktur sollte (bis auf die terminalen Elemente) mit der von (33) übereinstimmen. Dies wird durch den neuen Lexikoneintrag

($R_{geliebt}$)
$$V \rightarrow \textit{wird geliebt von}$$
$$V.head.sem.pred = \mathbf{lieben}$$
$$V.head.sem.arg_1 = V.head.obj.sem$$
$$V.head.sem.arg_2 = V.head.subj.sem$$

erreicht.

3.3.3.2 Die Spezifikation

Eine Spezifikation für einen Satz, der die Proposition

(37) **lieben(maria, peter)**

ausdrücken soll, wird durch die f-Struktur

(38)
$$\left[\; head: \left[\; sem: \begin{bmatrix} pred: & \mathbf{lieben} \\ arg_1: & \mathbf{maria} \\ arg_2: & \mathbf{peter} \end{bmatrix} \right] \right]$$

kodiert. Diese Spezifikation unifiziert z. B. mit den f-Strukturen der Sätze

(39) (a) *Maria liebt Peter*
(b) *Peter wird geliebt von Maria*

nicht aber mit denen von (33) und (36).

Im übrigen können Spezifikationen auch nicht-semantische Merkmale enthalten. Falls die Grammatik eine Repräsentation von Tempora in Form eines *head*-Merkmals *temp* vorsehen würde, so wäre eine gegenüber (38) genauere Spezifikation z. B.

(40)
$$\left[\textit{head}: \left[\begin{array}{ll} \textit{temp}: & \textit{pres} \\ \textit{sem}: & \left[\begin{array}{ll} \textit{pred}: & \textbf{lieben} \\ \textit{arg}_1: & \textbf{maria} \\ \textit{arg}_2: & \textbf{peter} \end{array}\right] \end{array}\right]\right]$$

Aus methodologischer Sicht wäre es wünschenswert, eine saubere Trennung von Syntax und Semantik anzustreben und dann zu fordern, daß ausschließlich semantische Information in der Spezifikation verwendet wird. Dies setzt im Fall von (40) voraus, daß die Semantik von temporalen Sachverhalten in sprachunabhängiger Weise als Teil des *sem*-Merkmals spezifiziert werden kann, ohne auf die syntaktische Information des *temp*-Merkmals zurückzugreifen. Solange für dieses (nicht einfache) Problem aber keine befriedigende Lösung vorliegt, ist es in praktischen Hinsicht sicherlich von Vorteil, Semantik und Syntax in der Spezifikation mischen zu können. Dies ist deshalb problemlos, weil in Unifikationsgrammatiken jegliche Art von Information uniform als f-Struktur repräsentiert ist.

Mehrere Spezifikationen können, sofern sie kompatibel sind, kombiniert werden, indem man die entsprechenden f-Strukturen unifiziert. Dies entspricht einer Konjunktion der durch sie repräsentierten Bedingungen.

Wie sieht nun die augmentierte Grammatik aus, die genau die Sätze generiert, die der Spezifikation (38) genügen? Da die Regeln der Beispielgrammatik einheitlich das Merkmal *cat* zu Beschreibung der Kategorie jedes c-Strukturknoten vorsehen, langt es, die neue Startregel den schon vorhandenen Regeln hinzuzufügen. Diese lautet

(R_{S_0}) $S_0 \rightarrow S$
$$\begin{aligned} &S.head.sem.pred = \textbf{lieben} \\ &S.head.sem.arg_1 = \textbf{maria} \\ &S.head.sem.arg_2 = \textbf{peter} \end{aligned}$$

Typischerweise enthalten die Startregeln augmentierter Grammatiken Unifikationsgleichungen, die lediglich die atomaren Werte bestimmter Pfade festlegen (in (R_{S_0}) die gesamte *sem*-Struktur). Denkbar wären aber auch Pfadäquivalenzen, wie z. B. in

(R_{S_0}') $S_0 \rightarrow S$
$$\begin{aligned} &S.head.sem.pred = \textbf{lieben} \\ &S.head.sem.arg_1 = S.head.sem.arg_2 \end{aligned}$$

4 Unifikation und Generierung in neuronalen Netzen

4.1 “Symbolismus” und Konnektionismus

Wie in Abschnitt 3.2.3 gezeigt wurde, stellen Unifikationsgrammatiken ein formal relativ einfaches, dabei aber flexibles und mächtiges Mittel zur Beschreibung natürlicher Sprachen dar. Unifikationsbasierte Grammatiken haben sich inzwischen in verschiedenen Spielarten in der Linguistik etabliert (FUG [Kay1984a]; LFG [Kaplan1982a]; GPSG [Gazdar1985a]) und haben dort einen erheblichen Anteil an der linguistischen Theoriebildung. Gleichzeitig stellen Unifikationsgrammatiken eine Art Höhepunkt in einer langen Entwicklung symbolorientierter Formalismen zur Sprachbeschreibung dar.

Es ist daher interessant zu untersuchen, inwiefern sich dieser Formalismus mit dem relativ neuen, auf *sub*symbolische Verarbeitung zielenden Paradigma des Konnektionismus vereinbaren läßt. In den Abschnitten 2.2.2 ff. wurde auf verschiedene Arbeiten verwiesen, die innerhalb dieses Paradigmas Ansätze zur Sprachverarbeitung machen. Es wurde geschildert, daß diese Ansätze Eigenschaften haben, die sie als Modelle für typisch “menschliche” (im Gegensatz zu mehr formalen) Aspekte der Sprachverarbeitung interessant erscheinen lassen. Dabei wurde allerdings auch bemerkt, daß diese Ansätze bislang eher auf linguistisch “niederen” Ebenen angesiedelt sind, und daß sich für ihre Ausweitung bereits auf dem Gebiet der Syntax (u. a. aufgrund von rekursiven Strukturen) erhebliche Probleme bieten.

Es kann durchaus sein, daß die traditionellen Mittel der linguistischen Beschreibung, zumal sie sich immer auch an den gängigen Paradigmen der Informationsverarbeitung orientiert haben, einer grundlegenden Revision bedürfen, bevor sie für konnektionistische Realisierungen relevant werden. Aus praktischen Erwägungen ist es jedoch sicher sinnvoll, den Fundus linguistischen Wissens, der in Form überkommener Theorien zur Verfügung steht, nicht einfach über Bord zu werfen, sondern ihn auf seine Brauchbarkeit für den neuen Ansatz hin abzuklopfen. Man sollte sich jedoch einiger grundsätzlicher Probleme im Zusammenhang mit diesem Vorgehen bewußt sein.

Der konnektionistische Ansatz bezieht einen Hauptteil seines Reizes aus der relativen Leichtigkeit, mit der sich bestimmte Aspekte der menschlichen kognitiven *Performanz* in ihm modellieren und erklären lassen, die mit den symbolorientierten Methoden der KI nur schwer nachzubilden sind. Symbolische Modelle eignen sich dagegen scheinbar relativ gut, um die *Kompetenz* menschlichen Verstandes zu erfassen. Beispielsweise entsprechen die Verfahren, die in automatischen Theorembeweisen zum Einsatz kommen, eher dem Vorgehen eines menschlichen Mathematikers, der einen Beweis *nachprüft*, als dem "intuitiv" geleiteten *Finden* eines solchen.

In Analogie dazu kann man mit gutem Grund behaupten, daß die meisten linguistischen Theorien in erster Linie Theorien der sprachlichen Kompetenz sind und gar nicht erst versuchen, performative Aspekte zu erklären. So gesehen ist von vornherein keineswegs zu erwarten, daß symbolorientierte Formalismen eine brauchbare Grundlage für konnektionistische Modelle abgeben.

Andererseits muß letzlich auch symbolische Verarbeitung, zu der Menschen zumindest auf der Ebene bewußten Denkens fähig sind, auf die Interaktion von Neuronen zurückzuführen sein. Obwohl die genaue Beziehung zwischen subsymbolischer und symbolischer Verarbeitung beim Menschen noch sehr unklar ist, ist es daher ein legitimes Forschungsziel zu untersuchen, wie *auch* Symbolverarbeitung in neuronalen Netzen möglich ist. Dies ist besonders da der Fall, wo es sich um symbolische Verfahren handelt, die für gängige Theorien von essentieller Bedeutung sind.

Neben Untersuchungen, die zu verstehen versuchen, wie neuronale Netze ihre spezifisch "konnektionistischen" Methoden, z. B. in Lernprozessen, entwickeln, gibt es daher eine Reihe von Arbeiten, die die Realisierung von traditionellen Methoden der KI in neuronalen Netzen zum Ziel haben.[37]

Die vorliegende Arbeit versteht sich in erster Linie als ein Teil dieser zweiten Strömung auf dem Gebiet des Konnektionismus. Allerdings wird sie auch von der Motivation geleitet, daß Kompetenztheorien bei geeigneter Anpassung und Verallgemeinerung innerhalb des konnektionistischen Paradigmas einen Hinweis auf brauchbare Performanzmodelle geben können. Dabei stellt sich z. B. auch die Frage nach der Lernbarkeit der verwendeten Methoden. Letzlich scheint es am erfolgversprechendsten, wenn man die Kluft zwischen symbolischer und neuronaler Verarbeitung durch aktives Bemühen auf beiden Seiten zu überbrücken sucht.

4.2 Das Verfahren

Bevor auf die Generierung von Sätzen in neuronalen Netzen eingegangen wird, ist als zentrale Voraussetzung ein Verfahren zur Unifikation in diesem Verarbeitungsmodell zu entwickeln. In Abschnitt 3.2.3 wurden Unifikationsgrammatiken so definiert, daß die c-Struktur nur einen minimalen Anteil am Formalismus hat und fast die gesamte grammatische Struktur in die f-Struktur verlagert ist. Die Erzeugung einer vollständigen f-Struktur wiederum wurde auf die Operation der (destruktiven) Unifikation zurückgeführt.

[37] Beispiele sind kontextfreie Parser [Charniak1987a], rekursive Datenstrukturen [Touretzky1986a], Produktionensysteme [Touretzky1985a] und Theorembeweisen mittels Resolution [Ballard1986a].

Aber auch unabhängig von einer linguistischen Anwendung kommt den f-Strukturen als Datenstrukturen und der Unifikation als elementarer Operation eine wichtige Bedeutung zu, z. B. im Bau von Theorembeweisern. Auch auf diesem Gebiet bemüht man sich um die Ausnutzung der herausragenden Eigenschaft von neuronalen Netzen, der hochgradig parallelen Verarbeitung. Obwohl es bereits Ansätze auf diesem Gebiet gibt (vgl. [Ballard1986a]), ist eine allgemeine Lösung des Unifikationsproblems – sei es nun auf Terme oder f-Strukturen bezogen – bisher nicht bekannt.[38]

Der hier entwickelte Ansatz leistet die Unifikation beliebiger rekursiver f-Strukturen, wobei man nicht auf die paarweise Ausführung der Operation beschränkt ist, sondern auch endliche Mengen von f-Strukturen parallel bearbeiten kann. Gerade diese Eigenschaft wird sich später bei der Anwendung auf Unifikationsgrammatiken als besonders nützlich erweisen. Das Verfahren beruht auf einer speziellen Repräsentation der Unifikationsoperation, die im folgenden entwickelt wird.

4.2.1 Unifikation als Knotenäquivalenz

Für die folgende Diskussion ist es am zweckmäßigsten, f-Strukturen als DAGs darzustellen. Um ein intuitives Verständnis für die graphentheoretische Interpretation der Unifikation zu bekommen, betrachten wir ein bereits unter (24) präsentiertes Beispiel, die Unifikation dreier f-Strukturen. Die entsprechenden DAGs sind in Abb. (41) dargestellt.

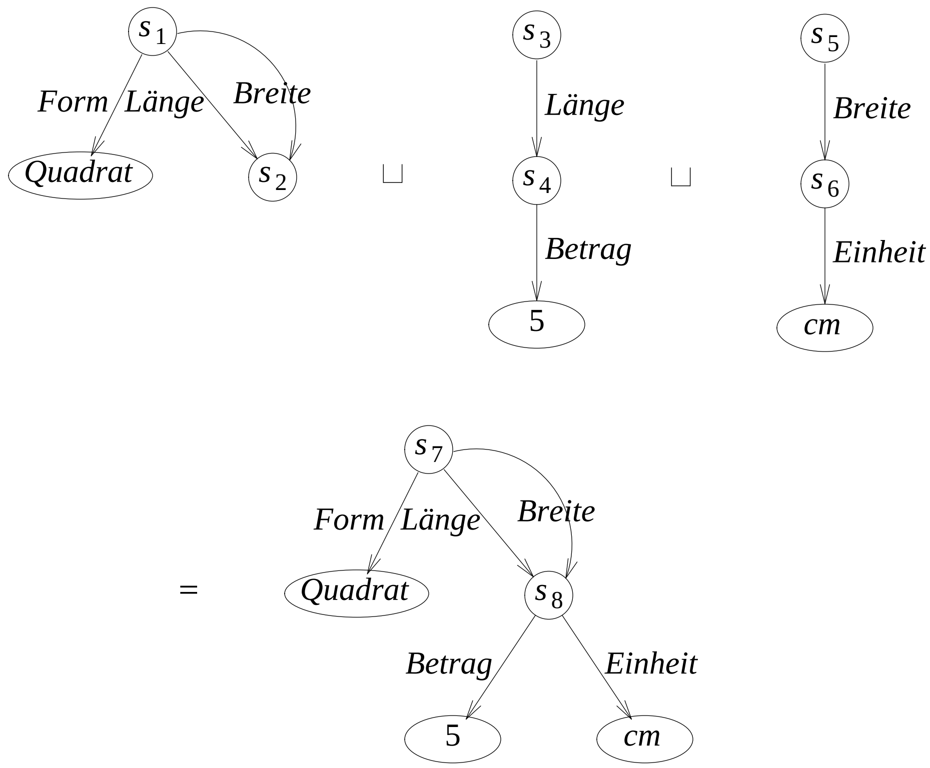
Zwischen den Knoten der drei Operanden und den Knoten des Unifikats ist eine Entsprechung festzustellen. Offensichtlich entsprechen die Knoten s_1, s_3 und s_5 dem Knoten s_7 im Unifikat. Analog entsprechen s_2, s_4 und s_6 dem Knoten s_8. Diese Entsprechungen entstehen, wenn man einmal die Wurzeln einander zuordnet, dann die Kanten verfolgt und Knoten, die über gleiche Pfade erreicht werden, ebenfalls einander zuordnet.

Diesen Zuordnungen der Operandenknoten zu Unifikatsknoten implizieren aber auch eine Zuordnung der Operandenknoten untereinander, und zwar in Form einer Partitionierung in *Äquivalenzklassen*. Man erhält im Beispiel so die Klassen $\{s_1, s_3, s_5\}$, $\{s_2, s_4, s_6\}$, $\{Quadrat\}$, $\{5\}$ und $\{cm\}$. Jeder dieser Klassen entspricht genau ein Knoten des Unifikats. Man kann sogar die Äquivalenzklassen selbst als das Resultat der Unifikation betrachten, wenn man die Menge der "Kanten" zwischen zwei Klassen aus der Vereinigung der Kanten zwischen den Klassenelementen bildet.

Die Idee, Unifikation als die Bildung geeigneter Äquivalenzklassen über der Menge der Knoten der zu unifizierenden Strukturen zu begreifen, wird nun präzisiert. Zunächst ist es wesentlich, die Klassenbildungen zu charakterisieren, die für eine Unifikation in Frage kommen. Wir betrachten dafür nicht einzelne f-Strukturen, die unifiziert werden sollen, sondern fassen alle f-Strukturen als Teilstrukturen eines einzigen (nicht-zusammenhängenden) Graphen aus $\overline{\mathbf{FS}}(F, V)$ auf. Dies sind f-Strukturen, die keine als Wurzel ausgezeichneten Knoten besitzen; sie

[38] Auf den angegeben Artikel wird zwar oft in dem Sinne verwiesen, daß das Problem der Unifikation mit konnektionistischen Mitteln gelöst sei, in der Tat wird aber dort nur mit einstelligen Termen operiert, in denen eine Variable durch atomare Werte substituiert wird. Der rekursive Aspekt von f-Strukturen, Termen und Unifikation ist also in keiner Weise berücksichtigt.

(41)



werden im folgenden als *verallgemeinerte f-Strukturen* bezeichnet. In einem Graphen aus $\overline{\mathbf{FS}}(F, V)$ können wir einen Knoten x mit der f-Struktur identifizieren, zu der er die Wurzel bildet. Dies ist deshalb möglich, weil in einem solchen Graphen die Kanten so markiert sind, daß alle Knoten, die von x erreichbar sind, einen Teilgraph bilden, der Definition (11) genügt. Wenn wir also im folgenden etwas salopp von der *f-Struktur* (eines Knotens) x sprechen, dann ist immer dieser von x ausgehende Teilgraph gemeint.

(42) **Definition.** Sei s eine verallgemeinerte f-Struktur aus $\overline{\mathbf{FS}}(F, V)$ mit Knotenmenge $\mathbf{F}$. Eine Äquivalenzrelation ~ auf $\mathbf{F}$ heißt mit s *verträglich*, wenn für alle $x, x', y, y' \in \mathbf{F}$ und $f \in F$ gilt:

(a) Wenn $x.f = x'$, $y.f = y'$ und $x \sim y$, so ist auch $x' \sim y'$.

(b) Sind x und y atomar und ist $x \sim y$, so ist $x = y$.

(c) Ist x atomar und y komplex und ist $x \sim y$, dann ist $y.f$ undefiniert.

Die Verträglichkeit einer Äquivalenzrelation bedeutet, daß die Kanten zwischen den Äquivalenzklassen nicht in Konflikt stehen, und ist Voraussetzung dafür, daß die Äquivalenzklassen selbst als Knoten einer (verallgemeinerte) f-Struktur interpretiert werden können.

Es muß nun gezeigt werden, daß die Äquivalenzklassenbildung tatsächlich ausreicht, um Unifizierbarkeit und Unifikation exakt zu charakterisieren. Dies leistet der folgende Satz.

(43) **Satz.** Geben sei eine verallgemeinerte f-Struktur $s_0 \in \overline{\mathbf{FS}}(F, V)$ mit Knotenmenge $\mathbf{F}$ und Knoten $s_1, s_2, \ldots, s_n \in \mathbf{F}$. Dann gilt: Die f-Strukturen $s_1, \ldots, s_n$ unifizieren genau dann,

wenn es eine zu s_0 verträgliche Äquivalenzrelation ~ auf **F** gibt derart, daß $s_1 \sim \cdots \sim s_n$ gilt.

Beweis. Der Beweis beschränkt sich auf den Fall $n = 2$, ist aber ohne Schwierigkeiten zu verallgemeinern.

“Wenn, dann”. Angenommen, ~ ist eine verträgliche Äquivalenzrelation, mit der $s \sim t$ gilt für zwei Knoten $s, t \in \mathbf{F}$. Wir zeigen, daß die Menge der Äquivalenzklassen über **F**, **F**/~, eine f-Struktur bilden, die isomorph zu einer Struktur $s_\sim \in \mathbf{FS}(F, V)$ ist, mit der $s \sqsubseteq s_\sim$ und $t \sqsubseteq s_\sim$ gilt.

Wir betrachten die Klassen in **F**/~ als Knoten eines Graphen und definieren in diesem folgende Kanten:

(*) $$[x].f = [y] :\Leftrightarrow \exists x' \in [x], y' \in [y]: x'.f = y'$$

für alle $x, y \in \mathbf{F}$. Die Verträglichkeit von ~ stellt dabei sicher, daß der so gegebene Graph selbst wiederum eine verallgemeinerte f-Struktur darstellt, denn

(a) Aus $[x] = [y]$ folgt, daß für alle $f \in V$ auch $[x].f = [y].f$ gilt.

(b) Immer nur ein atomares x kann in einer Äquivalenzklasse sein und eine solche Klasse $[x]$ hat keine abgehenden Kanten.

Es ist nun leicht zu sehen, daß sich die von einem Knoten $x \in \mathbf{F}$ ausgehenden Pfade auf seine Äquivalenzklasse $[x] \in \mathbf{F}/\sim$ übertragen, d. h.

(c) Wenn $x.\mathbf{f} = y$ gilt, dann auch $[x].\mathbf{f} = [y]$.

Beweis. Wir zeigen (c) durch Induktion über die Länge des Pfades **f**.

Sei **f** der leere Pfad $\langle\rangle$. $x.\mathbf{f} = y$ bedeutet dann aber $x = y$, also auch $[x] = [y]$ und $[x].\mathbf{f} = [y]$.

Sei nun $\mathbf{f} = \langle f, \mathbf{f}' \rangle$ mit $f \in F$ und $\mathbf{f}' \in F^*$. Dann gibt es ein $x' \in \mathbf{F}$, so daß $x.f = x'$ und $x'.\mathbf{f}' = y$. Nach (*) und Induktionsvoraussetzung folgt daraus $[x].f = [x']$ und $[x'].\mathbf{f}' = [y]$, also auch $[x].\mathbf{f} = [y]$.

Wir definieren nun die neue f-Struktur $s_\sim \in \mathbf{FS}(F, V)$. Wir betrachten die Äquivalenzklassen, die nach (*) über Kanten von $[s] = [t]$ erreichbar sind. Dann identifizieren wir jede Äquivalenzklasse mit einem Knoten der neuen f-Struktur und übertragen entsprechend die durch (*) definierten Kanten. Wegen (b) können wir dabei alle $v \in V$ mit ihren Klassen $[v]$ identifizieren, ohne inkonsistent zu werden.

$s_\sim$ ist also isomorph zu einem Teil des ursprünglichen Graphen über **F**/~. Die angegebe Konstruktion war notwendig, um formal wieder eine f-Struktur aus $\mathbf{FS}(F, V)$ zu erzeugen; wir bezeichnen die Knoten von $s_\sim$ jedoch weiterhin durch entsprechenden Äquivalenzklassen $[x]$.

Mit (c) folgt für die so gebildeten f-Struktur

(d) Es gilt $s \sqsubseteq s_\sim$ und $t \sqsubseteq s_\sim$.

Beweis. Wir zeigen $s \sqsubseteq s_\sim$ anhand von Definition (18).

Bedingung (18)(a): Sei $s.\mathbf{f}$ definiert, d. h. $s.\mathbf{f} = x$ für irgendein x. Aus (c) folgt $[s].\mathbf{f} = [x]$ und nach der Konstruktion von $s_\sim$ ist $s_\sim.\mathbf{f}$ definiert.

Bedingung (18)(b): Sei $s.\mathbf{f}=v$ für ein $v \in V$. Aus (c) folgt $[s].\mathbf{f}=[v]$ und nach Konstruktion $s_{\sim}.\mathbf{f}=v$.

Bedingung (18)(c): Sei $s.\mathbf{f}=x=s.\mathbf{g}$ für irgendein x. Aus (c) folgt $[s].\mathbf{f}=[x]=[s].\mathbf{g}$ und nach Konstruktion $s_{\sim}.\mathbf{f}=s_{\sim}.\mathbf{g}$. (c) wie behauptet $s \sqsubseteq s_{\sim}$ und $t \sqsubseteq s_{\sim}$.

$t \sqsubseteq s_{\sim}$ folgt analog, da $[s]=[t]$.

s und t haben also als eine obere Schranke $s_{\sim}$ und sind damit unifizierbar.

"Genau, wenn". Seien $s, t \in V$ zwei f-Strukturen, die unifizierbar sind, d. h. es gibt eine Struktur $r \in \mathbf{FS}(F, V)$, so daß $s, t \sqsubseteq r$. Jedes r mit dieser Eigenschaft induziert auf $\mathbf{F}$ eine Relation $\sim_r$ wie folgt:

$$(**) \quad x \sim_r y \;:\Leftrightarrow \begin{cases} r.\mathbf{f}=r.\mathbf{g} & \text{falls } \exists\, \mathbf{f},\mathbf{g} \in \mathbf{F}\text{:}\ (s.\mathbf{f}=x \vee t.\mathbf{f}=x) \wedge (s.\mathbf{g}=y \vee t.\mathbf{g}=y) \\ x=y & \text{sonst} \end{cases}$$

Reflexivität, Symmetrie und Transitivität von $\sim_r$ folgen aus den entsprechenden Eigenschaften von =. (Der zweite Zweig der Definition soll die Reflexivität für die Knoten aus $\mathbf{F}$ sichern, die nicht von einem der Knoten s oder t erreichbar sind, also nicht zu den f-Strukturen s bzw. t gehören.)

Wir zeigen, daß $\sim_r$ die geforderten Eigenschaften hat.

(e) $\sim_r$ ist mit s_0 verträglich.

Beweis (anhand von Definition (42)). Wir betrachten zunächst Knoten, die erreichbar sind. Sei $x \sim_r y$ und $\mathbf{f},\mathbf{g} \in F^*$ mit $r.\mathbf{f}=r.\mathbf{g}$, $s.\mathbf{f}=x$ und $t.\mathbf{g}=y$.[39]

Bedingung (42)(a): Sind x und y beide atomar, so gilt wegen $s, t \sqsubseteq r$, daß $x=s.\mathbf{f}=r.\mathbf{f}=r.\mathbf{g}=t.\mathbf{g}=y$.

Bedingung (42)(b): Bilde die verlängerten Pfade $\mathbf{f}'=\langle \mathbf{f}, f\rangle$ und $\mathbf{g}'=\langle \mathbf{g}, f\rangle$. Mit diesen gilt dann $s.\mathbf{f}'=x'$ und $t.\mathbf{g}'=y'$. Wegen $s, t \sqsubseteq r$ sind $\mathbf{f}', \mathbf{g}'$ auch in r definiert und es gilt $r.\mathbf{f}'=r.\mathbf{g}'$. Damit aber gilt auch $x' \sim y'$.

Bedingung (42)(c): Sei x atomar und y komplex. Wäre $y.f$ definiert, dann wäre wegen $s, t \sqsubseteq r$ auch $(r.\mathbf{g}).f$ definiert. $r.\mathbf{g}=r.\mathbf{f}$ ist aber atomar, also kann $y.f$ nicht definiert sein.

Für den zweiten Zweig von (**) mit $x=y$ sind die Bedingungen von Definition (42) trivialerweise erfüllt.

Schließlich gilt

(f) $s \sim_r t$.

Beweis. Setze in (**) $\mathbf{f}=\mathbf{g}=\langle\rangle$.

Damit ist insgesamt gezeigt, daß $\sim_r$ eine Äquivalenzrelation von der im Satz genannten Art darstellt.

---

[39] Die anderen drei Möglichkeiten mit $s.\mathbf{f}=x \wedge s.\mathbf{g}=y$, $t.\mathbf{f}=x \wedge s.\mathbf{g}=y$ bzw. $t.\mathbf{f}=x \wedge t.\mathbf{g}=y$ werden völlig analog behandelt.

Um aus Satz (43) einen Unifikationsalgorithmus ableiten zu können, der auf der Bildung von Äquivalenzklassen über der Knotenmenge der zu unifizierenden Strukturen basiert, muß die Aussage des Satzes in zweierlei Hinsicht verschärft werden. Dies geschieht in den folgenden beiden Lemmata.

Wenn im folgenden die Unifikation von f-Strukturen $s_0, s_1, \ldots, s_n$ betrachtet wird, können Aussagen in der Regel nur über *erreichbare* Knoten gemacht werden. Damit sind jeweils diejenigen Knoten x gemeint, zu denen es einen Pfad $\mathbf{f} \in F^*$ gibt, so daß $x = s_i.\mathbf{f}$ für eines der i ist. Nicht erreichbare Knoten gehören nicht zu den betrachteten f-Strukturen und sind daher auch nicht von Interesse für die Betrachtung.

Der Beweis von Satz (43) zeigt, wie man einerseits aus einer verträglichen Äquivalenzrelation $\sim$ eine f-Struktur $s_{\sim}$ konstruieren kann, die von den zu unifizierenden Strukturen subsumiert wird. Andererseits wird durch eine subsumierte f-Struktur r eine verträgliche Äquivalenzrelation $\sim_r$ induziert. Für die weitere Entwicklung muß nachgewiesen werden, daß die beiden Operationen – Konstruktion der f-Struktur und Induktion der Äquivalenzrelation – invers zu einander sind.

(44) **Lemma.** Es gelten die Voraussetzungen und Bezeichnungen von Satz (43).

(a) Ist $\sim$ eine mit s_0 verträgliche Äquivalenzrelation mit $s_0 \sim \cdots \sim s_n$, so ist auf der Menge der erreichbaren Knoten aus $\mathbf{F}$ die Relation $\sim_{s_\sim}$ identisch mit $\sim$.[40]

(b) Ist $r \in \mathbf{FS}(F, V)$ eine f-Struktur mit $s_0, \ldots, s_n \sqsubseteq r$, dann gilt $s_{\sim_r} \sqsubseteq r$.

Beweis. Wie im Beweis zu (43) seien $s, t \in \mathbf{F}$ die zu unifizierenden Knoten mit $s \sim t$.

Ad (a). Seien beliebige $x, y \in \mathbf{F}$ gegeben, die von s oder t erreichbar sind, d. h. wir nehmen an, daß $x = s.\mathbf{f}$ und $y = t.\mathbf{g}$.[41] $[s] = [t]$ ist die Wurzel von $s_{\sim}$. Wir zeigen, daß $x \sim y$ genau dann, wenn $x \sim_{s_\sim} y$.

Sei $x \sim y$. Wegen Punkt (43)(c) folgt aus $x = s.\mathbf{f}$ und $y = t.\mathbf{g}$, daß $[x] = [s].\mathbf{f}$ und $[y] = [t].\mathbf{g}$. Wegen $x \sim y$ folgt jetzt $s_{\sim}.\mathbf{f} = [s].\mathbf{f} = [x] = [y] = [t].\mathbf{g} = s_{\sim}.\mathbf{g}$ und damit $x \sim_{s_\sim} y$.

Sei nun $x \sim_{s_\sim} y$. Aus (**) ergibt sich $s_{\sim}.\mathbf{f} = s_{\sim}.\mathbf{g}$ und wegen Punkt (43)(c) $[x] = [s].\mathbf{f} = s_{\sim}.\mathbf{f} = s_{\sim}.\mathbf{g} = [t].\mathbf{g} = [y]$. Das heißt aber $x \sim y$.

Ad (b). Wir zeigen zunächst die Gültigkeit zweier Hilfsbehauptungen über die durch r induzierten Äquivalenzklassen und die mittels (*) zwischen ihnen definierten Kanten:

(c) Wenn $[s].\mathbf{f} = [x]$, dann ist x von s oder t erreichbar.

Beweis. Induktion über die Länge von $\mathbf{f}$.

Sei $\mathbf{f} = \langle\rangle$. Es gilt also $[s] = [x]$, d. h. $s \sim_r x$. Ist $s = x$, so ist x trivialerweise von s erreichbar. Ist $s \neq x$, so folgt aus (**), daß x von s oder t erreichbar ist.

---

[40] Auf Teil (a) des Lemmas kann streng genommen im weiteren Verlauf auch verzichtet werden. Er ist jedoch ohne großen Aufwand zu zeigen und vervollständigt das intuitive Verständnis von der Beziehung zwischen den diskutierten Operationen.

[41] Für die anderen Fälle vgl. Fußnote 39.

Sei $\mathbf{f} = \langle \mathbf{f}', f \rangle$. Es sei $[z] := [s].\mathbf{f}'$, $[z].f = [x]$. z ist nach Voraussetzung erreichbar. Nach (*) gibt es $z' \sim_r z$ und $x' \sim_r x$, so daß $z'.f = x'$. Ist $z' = z$, ist z' erreichbar. Ist $z' \neq z$, folgt aus (**), daß z' trotzdem von s oder t erreichbar ist. z' ist also in jedem Fall erreichbar und damit auch x'. Ist $x = x'$, ist x ebenfalls erreichbar. Ist $x \neq x'$, so folgt die Erreichbarkeit wie oben aus (**).

(d) Wenn $[s].\mathbf{f} = [x]$ und x über einen Pfad $\mathbf{g}$ erreichbar ist, gilt $r.\mathbf{f} = r.\mathbf{g}$.

Beweis. Induktion über die Länge von $\mathbf{f}$.

Sei $\mathbf{f} = \langle\rangle$, also $[s] = [x]$; x sei über $\mathbf{g}$ erreichbar. s ist trivialerweise über $\mathbf{f}$ erreichbar. Aus $s \sim_r x$ und (**) folgt die Behauptung.

Sei nun $\mathbf{f} = \langle \mathbf{f}', f \rangle$ mit $[s].\mathbf{f}' = [z]$ und $[z].f = x$; x sei über $\mathbf{g}$ erreichbar. Nach (*) gibt es $z' \sim_r z$ und $x' \sim_r x$, so daß $z'.f = x'$. Wegen $[z'] = [z]$ ist nach Voraussetzung z' über ein $\mathbf{g}'$ erreichbar mit $r.\mathbf{g}' = f.\mathbf{f}'$. Daraus folgt, daß x' über $\langle \mathbf{g}', f \rangle$ erreichbar ist. Aus $x \sim_r x'$ und (**) folgt $r.\mathbf{g} = r.\langle \mathbf{g}', f \rangle$ und insgesamt $r.\mathbf{g} = r.\langle \mathbf{g}', f \rangle = (r.\mathbf{g}').f = (r.\mathbf{f}').f = r.\langle \mathbf{f}', f \rangle = r.\mathbf{f}$.

Mithilfe von (c) und (d) kann nun leicht anhand von Definition (18) nachgewiesen werden, daß $s_{\sim_r} \sqsubseteq r$.

Bedingung (18)(a): Sei $s_{\sim_r}.\mathbf{f}$ definiert, d. h. $[s].\mathbf{f} = [x]$ für irgendein x. Nach (c) und (d) ist x über ein $\mathbf{g}$ erreichbar mit $r.\mathbf{g} = r.\mathbf{f}$. Also ist auch $r.\mathbf{f}$ definiert.

Bedingung (18)(b): Sei $s_{\sim_r}.\mathbf{f} = v$ mit $v \in V$, d. h. $[s].\mathbf{f} = [v]$. Nach (c) und (d) gilt $s.\mathbf{g} = v$ oder $t.\mathbf{g} = v$ mit $r.\mathbf{f} = r.\mathbf{g}$. Wegen $s, t \sqsubseteq r$ folgt $r.\mathbf{g} = v$ und insgesamt $r.\mathbf{f} = v$.

Bedingung (18)(b): Sei $s_{\sim_r}.\mathbf{f}_1 = s_{\sim_r}.\mathbf{f}_2$, d. h. $[s].\mathbf{f}_1 = [x] = [s].\mathbf{f}_2$ für irgendein x. Nach (c) ist x über ein $\mathbf{g}$ erreichbar, mit dem nach (d) gilt: $r.\mathbf{f}_1 = r.\mathbf{g} = r.\mathbf{f}_2$.

Damit ist Teil (b) des Lemmas bewiesen.

Der zweite Punkt, in dem die Aussage von Satz (43) verschärft und damit präzisiert werden muß, betrifft das Verhältnis zwischen zwei nach (**) induzierte Äquivalenzrelationen, wenn die induzierenden f-Strukturen in einer Subsumptionsbeziehung stehen. Man stößt dabei auf eine Halbordnung zwischen Äquivalenzrelationen, die *Feinheit*.

Man sagt von zwei Äquivalenzrelatonen $\sim_1$ und $\sim_2$, daß $\sim_1$ *feiner* ist als $\sim_2$ (bzw. $\sim_2$ *gröber* als $\sim_1$), wenn aus $x \sim_1 y$ stets folgt, daß $x \sim_2 y$. Anschaulich heißt das, daß die Äquivalenzklassen von $\sim_1$ immer vollständig in denen von $\sim_2$ enthalten sind. $\sim_1$ stellt also eine feinere Partitionierung der betrachteten Menge dar als $\sim_2$.

Wie zu vermuten, spiegelt sich die Spezialisierung der f-Struktur in einer *Vergröberung* der induzierten Äquivalenzrelation wieder.

Umgekehrt überträgt sich die “feiner als”-Ordnung zwischen verträglichen Äquivalenzrelationen auf die “allgemeiner als”-Ordnung, wenn nach (*) eine f-Struktur konstruiert wird.

Diese Zusammenhänge werden im folgenden Lemma formuliert.

(45) **Lemma.** Es gelten die Voraussetzungen und Bezeichnungen von Satz (43).

(a) Sind $r, r' \in \mathbf{FS}(F, V)$ f-Strukturen mit $s_0, \ldots, s_n \sqsubseteq r \sqsubseteq r'$, dann gilt, daß auf der Menge der erreichbaren Knoten $\sim_r$ feiner ist als $\sim_{r'}$.[42]

(b) Sind $\sim, \sim'$ mit s_0 verträgliche Äquivalenzrelationen mit $s_0 \sim \cdots \sim s_n$ bzw. $s_0 \sim' \cdots \sim' s_n$, dann gilt $s_{\sim} \sqsubseteq s_{\sim'}$.

Beweis. Wie schon in den vorhergehenden Beweisen seinen s und t die Wurzeln der unifizierbaren Strukturen.

Ad (a). Wir betrachten zwei beliebige Knoten aus den f-Strukturen s und t und nehmen wieder o. B. d. A an, daß $x = s.\mathbf{f}$ und $y = t.\mathbf{g}$. Wir zeigen, daß mit $x \sim_r y$ auch $x \sim_{r'} y$ gilt.

Aus $x \sim_r y$ folgt nach (**) $r.\mathbf{f} = r.\mathbf{g}$. Wegen $r \sqsubseteq r'$ gilt dann auch $r'.\mathbf{f} = r'.\mathbf{g}$. Also folgt wieder nach (**) $x \sim_{r'} y$.

Ad (b). Auch in diesem Lemma muß zunächst eine Hilfsbehauptung gezeigt werden. Sie charakterisiert die Pfade, die bei der Konstruktion (*) entstehen und stellt eine Präzisierung von (43)(c) dar.

(c) $[x].\langle f_1, \ldots, f_n \rangle = [y]$ gilt genau dann, wenn es $x_0, \ldots, x_n, y_0, \ldots, y_n \in \mathbf{F}$ gibt mit $x \sim x_0, y \sim y_n, x_i \sim y_i$ für $i = 0, \ldots, n$ und $x_{i-1}.f_i = y_i$ für $i = 1, \ldots, n$.

Beweis. Induktion über n in beide Richtungen.

"Wenn, dann". Sei $n = 0$. Aus $x \sim x_0$, $y \sim y_0$ und $x_0 \sim y_0$ folgt $x \sim y$, also $[x] = [y]$. Damit ist trivialerweise auch $[x].\langle\rangle = [y]$.

Sei $n > 0$. Aus $x \sim x_0$, $x_i \sim y_i$ für $i = 0, \ldots, n-1$ und $x_{i-1}.f_i = y_i$ für $i = 1, \ldots, n-1$ folgt nach Voraussetzung $[x].\langle f_1, \ldots, f_{n-1} \rangle = [y_{n-1}]$. Aus $y_{n-1} \sim x_{n-1}$, $y \sim y_n$ und $x_{n-1}.f_n = y_n$ folgt nach (*) $[y_{n-1}].f_n = [y]$. Insgesamt gilt also $[x].\langle f_1, \ldots, f_n \rangle = [y]$.

"Genau, wenn". Sei $n = 0$, d. h. $[x] = [y]$. Mit $x = x_0 = y_0 = y$ folgt die Behauptung.

Sei $n > 0$ mit $[x].\langle f_1, \ldots, f_{n-1} \rangle = [z]$ und $[z].f_n = [y]$. Nach Voraussetzung gibt es $x_0, \ldots, x_{n-1}, y_0, \ldots, y_{n-1}$ mit $x \sim x_0$, $z \sim y_{n-1}$, $x_i \sim y_i$ für $i = 0, \ldots, n-1$ und $x_{i-1}.f_i = y_i$ für $i = 1, \ldots, n-1$. Nach (*) gibt es $z' \sim z$ und $y' \sim y$ mit $z'.f_n = y'$. Es gilt $y_{n-1} \sim z \sim z'$. Daher ist die Behauptung erfüllt, wenn $x_n := y_n := y'$ und $x_{n-1} := z'$ gesetzt wird.

Seien nun zwei Relationen $\sim$ und $\sim'$ gegeben, $\sim$ feiner als $\sim'$, die den Voraussetzungen von (b) genügen. Wir zeigen mit (c) und anhand von Definition (18), daß $s_{\sim} \sqsubseteq s_{\sim'}$.

Bedingung (18)(a): Sei $s_{\sim}.\mathbf{f}$ definiert, d. h. $[s].\mathbf{f} = [x]$ für irgendein x. Es folgt die rechte Seite von (c). Alle Äquivalenzen mit $\sim$ können durch $\sim'$ ersetzt werden, da $\sim'$ gröber ist als $\sim$. Es folgt die linke Seite von (c) mit den Äquivalenzklassen von $\sim'$. Also ist $s_{\sim'}.\mathbf{f}$ definiert.

Bedingung (18)(b): Sei $s_{\sim}.\mathbf{f} = v$ mit $v \in V$, d. h. $[s].\mathbf{f} = [v]$. Analog zu oben folgt $[s].\mathbf{f} = [v]$ auch für die Äquivalenzklassen von $\sim'$. Also ist $s_{\sim'}.\mathbf{f} = v$.

Bedingung (18)(b): Sei $s_{\sim}.\mathbf{f}_1 = s_{\sim}.\mathbf{f}_2$, d. h. $[s].\mathbf{f}_1 = [x] = [s].\mathbf{f}_2$ für irgendein x. Auch hier kann (c) angewendet und anschließend $\sim$ durch $\sim'$ ersetzt werden. Nochmalige Anwendung

[42] Auch hier ist Teil (a) eigentlich entbehrlich. Aus den in Fußnote 40 genannten Gründen ist er trotzdem aufgeführt.

von (c) in umgekehrter Richtung liefert $[s].\mathbf{f}_1=[x]=[s].\mathbf{f}_2$ für die Äquivalenzklassen von $\sim'$, also $s_{\sim'}.\mathbf{f}_1=s_{\sim'}.\mathbf{f}_2$.

Damit ist Teil (b) des Lemmas bewiesen.

Zusammenfassend läßt sich feststellen, daß offenbar subsumierte f-Strukturen in einer 1-zu-1-Beziehung zu entsprechenden verträglichen Partitionierungen der (erreichbaren) Knoten in Äquivalenzklassen stehen. Wenn wir die Unifizierbarkeit einer Menge von f-Strukturen prüfen wollen, so kann dies dadurch geschehen, daß wir systematisch die Konstruktion einer solchen Partitionierung versuchen. Ist eine Partitionierung möglich und haben wir darüber hinaus die feinster aller möglichen Partitionierungen gefunden, ist diese nach der Methode (*) von Satz (43) auch gleichzeitig isomorph zum Unifikat. Dieser Zusammenhang ist hier noch einmal formuliert als

(46) **Satz.** Gegeben sei eine verallgemeinerte f-Struktur $s_0\in\overline{\mathbf{FS}}(F,V)$ mit Knotenmenge $\mathbf{F}$ und $\sim$ eine zu s_0 verträgliche Äquivalenzrelation auf der Menge der von $s_1,\ldots,s_n\in\mathbf{F}$ erreichbaren Knoten aus $\mathbf{F}$ mit $s_1\sim\cdots\sim s_n$. Dann gilt: Ist $\sim$ die feinste aller Relationen mit diesen Eigenschaften, so gilt $s_1\sqcup\cdots\sqcup s_n=s_\sim$.

Beweis. Nach dem Beweis von Satz (43) gilt $s_1,\ldots,s_n\sqsubseteq s_\sim$. Sei $r\in\mathbf{FS}(F,V)$ eine weitere Struktur mit $s_1,\ldots,s_n\sqsubseteq r$. Nach dem Beweis von Satz (43) ist $\sim_r$ eine Relation mit den für $\sim$ geforderten Eigenschaften.

Nach Voraussetzung ist $\sim$ feiner als $\sim_r$. Nach Lemma (45)(b) gilt $s_\sim\sqsubseteq s_{\sim_r}$. Nach Lemma (44)(b) ist $s_{\sim_r}\sqsubseteq r$. Insgesamt folgt also $s_\sim\sqsubseteq r$. s ist also kleinste obere Schranke von $s_1,\ldots,s_n$, d. h. das gesuchte Unifikat.

Satz (46) gibt uns die formale Begründung für das schon angedeute Verfahren zur Berechnung des Unifikats mittels Konstruktion der feinstmöglichen verträgliche Partitionierung der Knotenmenge, bei der die Wurzeln der zu unifizierenden Strukturen in einer Partition liegen.

Der Algorithmus hierfür ist nun leicht aus Definition (42) und Satz (43) abzuleiten. Die feinstmögliche Äquivalenzrelation ist diejenige, in der die wenigsten Äquivalenzen gelten. Wir konstruieren daher systematisch die Äquivalenzklassen so, daß nur die "nötigen" Äquivalenzen gelten.

(47) **Algorithmus** zur Konstruktion der feinsten verträglichen Partitionierung der von $s_0,\ldots,s_n\in\mathbf{F}$ erreichbaren Knoten mit $s_0\sim\cdots\sim s_n$.

(a) Bilde die Klasse $C:=[s_1]:=\cdots:=[s_n]:=\{s_0,\ldots,s_n\}$.

(b) Für alle übrigen Knoten $x\in\mathbf{F}$ bilde eine Klasse $[x]:=\{x\}$.

(c) Wende auf C den Schritt (d) an.

(d) Gegeben sei eine Klasse $X\subset\mathbf{F}$. Für alle Paare $x,y\in X$ und alle $f\in F$ mit $x.f=x'$ und $y.f=y'$ vereinige die Klassen $[x']$ und $[y']$ und wende auf die neu entstandene Klassse Schritt (d) rekursiv an.

(e) Entsteht durch die Klassenbildung in Schritt (a) oder (d) eine Klasse mit zwei ungleichen atomaren Knoten oder einem atomaren Knoten und einem Knoten mit abgehender Kante, so brich das Verfahren ab; die Strukturen $s_0,\ldots,s_n$ sind dann nicht unifizierbar.

Der Algorithmus leitet von den Wurzeln der Strukturen ausgehend alle Äquivalenzen ab, die nach (42)(a) in einer verträglichen Relation zu gelten haben und nur solche. Wenn daher in Schritt (e) eine Verletzung der Verträglichkeit nach (42)(b) bzw. (c) entdeckt wird, kann der Algorithmus keinerlei *Backtracking* ausführen, d. h. es gibt keine verträgliche Relation der gesuchten Art.

Bei jeder Anwendung von Schritt (d) werden die Pfade, die von Knoten in den bearbeiteten Klassen ausgehen, kürzer. Da f-Strukturen nach unserer Definition keine Zyklen (beliebig lange Pfade) enthalten dürfen, ist die Terminierung des Algorithmus gesichert.

(47) stellt einen sog. *UNION-FIND*-Algorithmus dar, d. h. die Methode des Verfahrens beruht im wesentlichen auf dem der Identifizierung der Äquivalenzklasse $[x]$ zu einem Element x (*FIND*) und der destruktiven Vereinigung von Äquivalenzklassen (*UNION*).

4.3 Ein Netz zur Unifikation

Algorihmus (47) stellt eine Realisierung der Unifikationsoperation mit klassischen Mitteln dar. Im folgenden wird nun ein konnektionistischer Ansatz für das Problem beschrieben.

4.3.1 Das verwendete Modell

Die Unifikation ist von ihrer Definition her ein diskretes Problem: Zwei f-Strukturen unifizieren oder unifizieren nicht; ist Unifizierbarkeit gegeben, dann sind zwei Knoten in diesen Strukturen bezüglich des Unifikats äquivalent oder nicht äquivalent. Zwar sind auch Verallgemeinerungen der Unifikation denkbar, die Unifizierbarkeit als Werte über einem Kontinuum interpretieren (“fuzzy unification”), jedoch werden solche Ansätze hier zunächst nicht weiterverfolgt (siehe Abschnitt 4.4.2). Dies ist der Grund, warum das in den folgenden Abschnitten beschriebene Netz Units mit binärem Aktivationszustand verwendet, die über eine Schwellenfunktion aktiviert werden. Im folgenden wird nicht versucht, einen Überblick über alle möglichen Alternativen für formale Modellierung eines neuronalen Netzes zu geben. Stattdessen wird das hier verwendete Netz kurz vorgestellt und für andere Modelle auf [Rumelhart1986a] verwiesen.

Ein Netz der Größe N ist durch eine Matrix $\mathbf{w} \in \mathbb{R}^{N \times N}$ sowie einen Vektor $\mathbf{\Theta} \in \mathbb{R}^N$ bestimmt. Die Komponenten w_{ij} von $\mathbf{w}$ heißen *Gewichte*, die Komponenten Θ_i von $\mathbf{\Theta}$ *Schwellenwerte*. Der Zustand des Netzes zum Zeitpunkt t wird durch einen Vektor $\mathbf{a}(t) \in \mathbb{R}^N$ beschrieben. Aus den oben genannten Gründen sind die einzelnen Komponenten des Zustandvektors binär, d. h. in unserem Fall gilt stets $\mathbf{a}(t) \in \{0, 1\}^N$. Die i te Komponenten des Zustandsvektors $\mathbf{a}(t)$, $a_i(t)$, heißt die *Aktivation* der i ten *Unit*.

Das hier verwendete Netzmodell geht von einer diskreten Zeit aus, d. h. t nimmt nur ganzzahlige Werte an. Außerdem arbeitet das Netz *asynchron*, d. h. die Aktivationen der einzelnen Units werden nicht alle gleichzeitig neu berechnet, sondern zu jedem Zeitpunkt t wird immer nur eine Unit eine Aktualisierung ihres Zustandes vornehmen, während die anderen Units ihre Aktivation unverändert lassen. Wenn nicht ausdrücklich anders gesagt, werden dabei alle N Units mit gleicher Wahrscheinlichkeit berücksichtigt, so daß nach N Zeitschritten jede Unit im Durchschnitt ihre Aktivation einmal aktualisiert hat.

Die Vorschrift, nach der nun eine Unit i ihre Aktivation neu berechnet, lautet:

(48) $$a_i(t+1) = \left\{ \begin{array}{c} 1 \\ a_i(t) \\ 0 \end{array} \right\} \text{ falls } \sum_{j=1}^{N} w_{ji}\, a_j(t) - \Theta_i \left\{ \begin{array}{l} > 0 \\ = 0 \\ < 0 \end{array} \right.$$

Eine Unit i summiert also die Aktivationen aller anderen Units j und gewichtet dabei mit w_{ji}. Diese gewichtete Summe (die Eingabeaktivation) wird dann mit dem Unit-spezifischen Schwellenwert Θ_i verglichen. Übersteigt sie den Schwellenwert, wird die Unit aktiv; bleibt sie unter dem Schwellenwert, wird sie inaktiv, bei Gleichheit erfolgt keine Änderung der Aktivation.

In einer physischen Realisierungen von (48) würden zwischen den Units Verbindungen bestehen, über die diese sich ihre Aktivationen mitteilen. Daher interpretiert man das Gewicht w_{ji} als die Stärke des *Links* von Unit j nach Unit i. Ein $w_{ji} = 0$ bedeutet dann eine fehlende Verbindung, bei $w_{ji} > 0$ spricht man von einer *erregenden*, bei $w_{ji} < 0$ von einer *hemmenden* Verbindung. Die Schwellenwerte Θ_i realisieren eine Art Voreinstellung für die jeweilige Unit. Bei $\Theta_i > 0$ ist Unit i zunächst (ohne Eingabeaktivation) ausgeschaltet und wird erst aktiv, wenn die Eingabe diesen Wert überschreitet. Umgekehrt ist eine Unit mit $\Theta_i < 0$ ohne Eingabeaktivation angeschaltet und wird erst bei ausreichend hoher Eingabe über hemmende Links inaktiv.

Durch die Schwellenwerte lassen sich Randbedingungen (Eingaben), die das Netz von außen, d. h. aus seiner “Umgebung” empfängt, modellieren. Für jeden Eingabeparameter verwendet man eine Unit, die keinerlei Aktivation aus dem Netz selbst bekommt ($w_{ji} = 0$ für alle $j = 1, \ldots, N$). Ein $\Theta_i < 0$ kodiert eine “1” als Eingabe, $\Theta_i > 0$ entsprechend eine “0”. Dies ist gleichbedeutend mit der Aussage, daß eine solche *Eingabe-Unit* in ihrer Aktivation von außen “festgehalten” (*clamped*) wird.

Es sei noch erwähnt, daß man auf Schwellenwerte prinzipiell auch verzichten kann, wenn man stattdessen eine zusätzliche $(N+1)$te Unit annimmt, deren Aktivation immer 1 beträgt. Dann wird der Schwellenwert Θ_i ersetzt durch ein Gewicht $w_{(N+1),i} = -\Theta_i$.

Das hier verwendete Netzwerkmodell entspricht weitgehend den sog. Hopfield-Netzen [Hopfield1982a], die zusätzlich die Forderung einer symmetrischen Verbindungsstruktur ($w_{ji} = w_{ij}$) beinhalten.

Aus Gründen der Lesbarkeit werden die Units $1, \ldots, N$ im folgenden nicht durch Indizes, sondern symbolisch bezeichnet. Diese symbolischen Bezeichnungen erscheinen dabei in Klammern $\langle \cdots \rangle$ und werden sinngemäß anstatt der Indizes an Gewichten w, Aktivationen a, Schwellenwerten Θ usw. verwendet.

4.3.2 Unifikation durch Aktivationsausbreitung

4.3.2.1 Unifikation als “constraint satisfaction”

Eine wichtige Klasse von Problemen, die neuronale Netze typischerweise sehr gut lösen können, sind sog. *constraint satifaction*-Probleme. Dabei geht es darum, eine Reihe von Variablen so einzurichten, daß eine Menge von Bedingungen erfüllt sind. Durch die Bedingungen ergeben sich typischerweise eine Vielzahl von wechselseitigen Abhängigkeiten zwischen Variablen, so daß eine sequentielle Lösung durch inkrementelles Finden der Variablenbelegungen nur durch massives Backtracking erfolgen kann, wobei exponentielle Komplexitäten resultieren.

Gelingt es dagegen, die Variablen und ihre Abhängigkeiten in einem neuronalen Netz zu repräsentieren, so kann durch die Parallelität der Aktivationsausbreitung eine Lösung in linearer Zeit gefunden werden, weil das Netz in der Lage ist, alle Bedingungen simultan zu berücksichtigen. Gibt es keine Lösung, die alle Bedingungen erfüllt, dann finden solche Netze in der Regel Näherungslösungen die, wenn nicht optimal, doch zumindest brauchbar sind.

Eine Standardmethode, um neuronale Netze für derartige Probleme zu konstruieren, verwendet "lokale" Repräsentationen für die involvierten Variablen. Jeder Variable wird eine Unit zugewiesen, so daß deren Aktivation eine Hypothese über den Wert der Variable kodiert. Durch die Verbindungsstruktur wird erreicht, daß konsistente Hypothesen sich gegenseitig stützen, während sich widersprechende Variablenbelegungen um Aktivation konkurrieren. Die Gewichte der Verbindungen kodieren also die Menge der zu erfüllenden Bedingungen.

Satz (43) erlaubt es uns, auch die Unifikation von f-Strukturen als ein Problem von *constraint satisfaction* zu interpretieren. Die Variablen sind hier die Äquivalenzen bzw. Nichtäquivalenzen von Knoten der Strukturen, während die Bedingungen durch

(a) die Äquivalenz der Wurzeln,

(b) die Verträglichkeit mit der Kantenmenge und

(c) die Eigenschaften der Äquivalenz

gegeben sind. Wir wir gesehen haben, kommt Algorithmus (47) ohne Backtracking aus und hat deshalb eine Komplexität, die linear in der Anzahl der Koten (genauer: in der Anzahl der Knoten der kleinsten der zu unifizierenden Strukturen) ist. Wenn wir lediglich an der Unifikation einer festgelegten Menge von Strukturen interessiert sind, besteht der einzige Vorteil einer Lösung durch neuronale Netze in der Ausnutzung der möglichen Nebenläufigkeiten des Algorithmus. Diese besteht darin, daß mehrere zu unifizierende Pfade parallel verfolgt werden, wodurch die Zeitkomplexität linear in der *Tiefe* der beteiligten Strukturen wird.

Wir werden jedoch später sehen, daß sich erhebliche Vorteile ergeben, sobald in einer Menge von Strukturen zwischen *alternative* Unifikationen entschieden werden muß. Dies würde bei einer sequentiellen Implementierung zu Backtracking führen. Gerade das Problem der Entscheidung zwischen mehreren potentiellen Unifikationen tritt aber massiv auf, sobald die Generierung (oder auch Analyse) von Sätzen in Unifikationsgrammatiken betrachtet wird.

4.3.2.2 Repräsentation von Eingabe und Ausgabe

Voraussetzung für die Lösung eines Problems, nicht nur in neuronalen Netzen, ist eine geeignete Repräsentation von Ein- bzw. Ausgabe. Wenn man jedoch wie in konnektionistischen Ansätzen mit Absicht auf beliebig komplexe symbolische Darstellungen verzichtet, kommt der Frage der geeigneten Repräsentation eine Schlüsselrolle zu.

Die Eingabe für die Unifikation sind die zu unifizierenden Strukturen, wobei wie schon in der vorangegangenen Diskussion davon ausgegangen wird, daß mehrere Strukturen als ein DAG vorliegen, aus dem dann Teilstrukturen (f-Strukturen) durch ihre Wurzelknoten identifiziert werden. Dieser eine DAG wird durch die Menge seiner Kanten spezifiziert, wobei jede Kante als ein Tripel aus $\mathbf{F} \times F \times \mathbf{F}$ zu betrachten ist. Wir verwenden für diese Kanten eine lokale konnektionistische Repräsentation, d. h. wir ordnen jeder potentiellen Kante (x, f, y) die Unit $\langle x.f = y\rangle$ zu und interpretieren eine Aktivität $a_{\langle x.f = y\rangle} = 1$ als Existenz der Kante, eine Aktivität von 0 dagegen als Fehlen der Kante. Die Units, die auf diese Weise den gesamten Graphen repräsentieren, heißen *E[dge]-Units.*

Um das Resultat einer Unifikation (und Teilergebnisse während der Verarbeitung) repräsentieren zu können, muß die Paritionierung der Knotenmenge in Äquivalenzklassen dargestellt werden. Dies geschieht, indem die paarweisen Äquivalenzen zwischen Knoten lokal repräsentiert werden. Die Äquivalenz $x \sim y$ wird durch eine Unit $\langle x \sim y\rangle$ dargestellt, d. h. eine Aktivität von 1 zeigt an, daß x und y zu derselben Äquivalenzklasse gehören. Eine Aktivität von 0 dagegen bedeutet *nicht* unbedingt, daß $x \not\sim y$ gilt, sondern nur, daß über die Äquivalenz von x und y (noch) keine Aussage gemacht werden kann. Daher ist es notwendig, für die explizite Nicht-Äquivalenz von Knoten eine eigene Repräsentation vorzusehen, was analog durch entsprechende Units $\langle x \not\sim y\rangle$ geschieht. Die Units, die auf die geschilderte Weise insgesamt die Partitionierung der Knotenmenge repräsentieren, heißen im folgenden (weil sie letzlich das Unifikat repräsentieren) *U-* bzw. *NU-Units*.

4.3.2.3 Verbindungsstruktur und Konsistenz

Natürlich muß sichergestellt werden, daß die Menge der U- und NU-Units in ihrer Gesamtheit eine Äquivalenzrelation darstellt. Die Reflexivität braucht nicht repräsentiert zu werden, da die Eigenschaft $x \sim x$ für alle Knoten x invariant für alle möglichen Fälle gilt. Ein Unit, die diesen Sachverhalt repräsentieren würde, hätte also eine konstante Aktivation von 1. Ihre Wirkung auf andere Units des Netzes kann daher fest in die Verbindungsstruktur eingebaut werden. Aus diesem Grund kann auf U-Units der Form $\langle x \sim x\rangle$ ebenso wie auf die entsprechenden NU-Units verzichtet werden.

Die Symmetrie von ~ muß zwar repräsentiert werden, jedoch kann dies implizit ("intrinsisch") geschehen, indem die Units $\langle x \sim y\rangle$ und $\langle y \sim x\rangle$ identifiziert werden, d. h. die beiden symmetischen U-Units werden durch eine einzige neue U-Unit ersetzt und alle betroffenden Links auf diese eine Unit "umgelenkt". Die neue Unit nennen wir der Einfachheit halber wieder $\langle x \sim y\rangle$. Entsprechendes gilt auch hier für die NU-Units.

Während Reflexivität und Symmetrie der Äquivalenzrelation also nicht (explizit) durch die Verbindungsstruktur gesichert werden muß, verlangt die Transitivität eine explizite ("extrinsische") Repräsentation.[43] Die gleichzeitige 1-Aktivität von $\langle x \sim y\rangle$ und $\langle y \sim z\rangle$ muß eine 1-Aktivität von $\langle x \sim z\rangle$ nach sich ziehen. Entsprechend muß $\langle x \sim y\rangle$ zusammen mit $\langle y \not\sim z\rangle$ eine Aktivierung von $\langle x \not\sim z\rangle$ bewirken. In beiden Fällen geschieht dies durch zusätzliche Units, die Konjunktionen realisieren, *T[ransitivitäts]-Units* genannt. Die T-Unit $\langle x \sim y \sim z\rangle$ bekommt je einen Links von $\langle x \sim y\rangle$ und $\langle y \sim z\rangle$ und einen Schwellenwert, der ihr nur dann eine Aktivation von 1 erlaubt, wenn diese beiden U-Units gleichzeitig 1-Aktivation haben. Wird $\langle x \sim y \sim z\rangle$ auf diese Weise aktiv, so aktiviert sie ihrerseits über einen weiteren Link $\langle x \sim z\rangle$. Die Links, die zu

[43] Die unterschiedliche Behandlung der drei Eigenschaften entspricht deren unterschiedliche logische Struktur. Die Reflexivität stellt sich dar als eine einfache Formel mit konstantem Wahrheitswert; die Symmetrie ist eine Äquivalenz und die Transitivität eine Implikation.

den T-Units führen und von ihnen abgehen, heißen *T-Links*.

Durch die Einführung der NU-Units ist ein weiteres Konsistenz-Problem geschaffen worden. Es muß nämlich gewährleistet sein, daß sich entsprechende U- und NU-Units nicht gleichzeitig 1-Aktivation haben. Solche sich ausschließenden Aktivationen erzeugt man typischerweise durch wechselseitige Hemmung der Units, d. h. durch symmetrische negative Gewichte zwischen den Units. Wie sich in der weiteren Diskussion zeigen wird, haben jedoch hier die NU-Units eine stärkere “Aussagekraft”, d. h. die Aktivation einer U-Unit beruht auf schwächerer Evidenz als die der entsprechenden NU-Unit.

Man kann sich das an einer U-Unit klarmachen, die für die Äquivalenz der Wurzeln zweier zu unifizierender Strukturen steht. Die “positive Evidenz” für die Knotenäquivalenz kommt in diesem Fall aus der Umgebung, die die Unifikation “versucht”, indem die entsprechende U-Unit aktiviert wird. Falls jedoch “negative Evidenz” vorhanden ist, d. h. wenn die Strukturen inkompatibel sind, so wird die NU-Unit aktiviert. In einem solchen Fall muß die NU-Unit überwiegen, da die zwei Strukturen ja nicht unifizieren *müssen*.

Aus diesem Grund werden U- und NU-Units asymmetrisch verbunden, so daß eine 1-Aktivität der NU-Unit die Aktivität der entsprechenden U-Unit in jedem Fall auf 0 bringt. Dies wird durch eine negatives Gewicht erreicht, daß betragsmäßig sehr groß gegenüber allen positiven Gewichten an der U-Unit ist. Die Verbindungen von NU- zu U-Units heißen *NU-Links*.

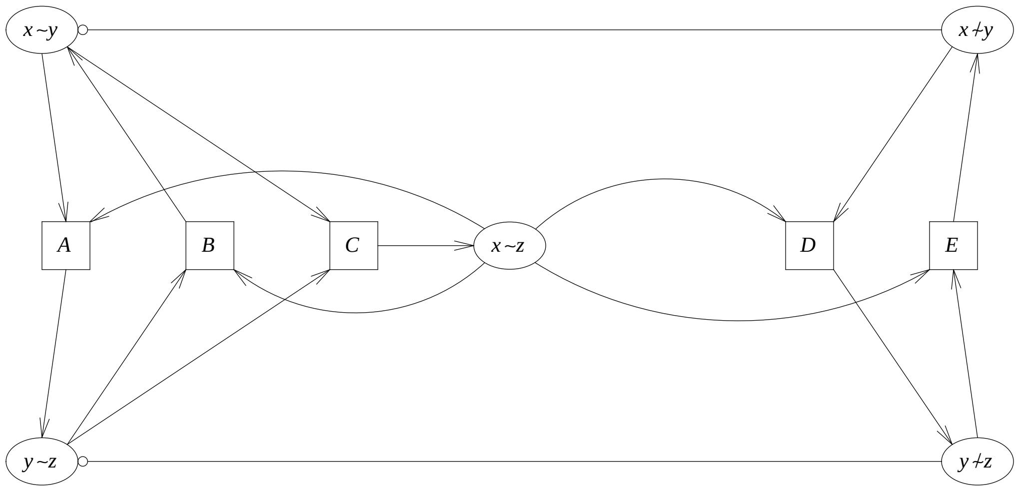
Die Units und Links, die der konsistenten Repräsentation der Äquivalenzrelation dienen, sind in Abb. (49) zusammengefaßt. Insgesamt stellen die beschriebenen Maßnahmen sicher, daß das Netz sich nur dann in einem Gleichgewichtszustand befindet, wenn die Gesamtheit der U- und NU-Units in konsistenter Form eine Äquivalenzrelation repräsentiert. Ein *Gleichgewichtszustand* ist dann gegeben, wenn die Aktivation aller Units stationär ist, d. h. jede Unit erhält bei einer Neuberechnung ihrer Aktivation den Wert, den sie bereits innehat. Umgekehrt kann jede beliebige Äquivalenzrelation in Form eines Gleichgewichtszustandes durch das Netz repräsentiert werden.

4.3.2.4 Verbindungsstruktur und Verträglichkeit

Die Verbindungsstruktur wird nun so erweitert, daß die repräsentierten Äquivalenzen nicht nur in sich konsistent, sondern auch verträglich zu den vorhandenen Kanten sind. Man kann dabei systematisch nach Definition (42) vorgehen.

Bedingung (42)(a) bedeutet, daß sich Äquivalenzen in den Graphen von oben nach unten fortpflanzen, und zwar über Paare von Kanten, die mit gleichen Merkmalen versehen sind. Es handelt sich hier um eine Art erweiterte Transitivität, die durch die Kantenstruktur induziert wird. Die Umsetzung dieser Bedingung erfolgt ähnlich wie bei der Transitivität der Äquivalenzrelation durch konjunktive Verbindungen zwischen den beteiligten Units. Für jedes Paar von E-Units $\langle x.f = x' \rangle$ und $\langle y.f = y' \rangle$ wird eine zusätzliche Unit $\langle x.f = x' {\sim} y.f = y' \rangle$ benötigt, die Links von den beiden E-Units und der U-Unit $\langle x \sim y \rangle$ bekommt. Der Schwellenwert dieser sog. *TD-Unit* wird so gewählt, daß nur bei 1-Aktivität aller drei angeschlossenen Units die TD-Unit ebenfalls eine 1-Aktivität erreicht. Diese gibt sie dann mit einem weiteren Link an die zweite U-Unit $\langle x' {\sim} y' \rangle$ weiter. Die vier Links, die auf diese Weise die Äquivalenzen von “oben nach unten” übertragen, werden *TD-Links* genannt.

(49)



Repräsentation von Äquivalenzen und Nicht-Äquivalenzen. Die Units A, B, C, D und E sind T-Units, die nur dann eine Aktivation von 1 erhalten, wenn beide eingehenden T-Links Eingabe liefern. Die rechteckige Form soll andeuten, daß diese Units ein Konjunktion realisieren, während die übrigen (durch Ellipsen dargestellten) Units disjunktiv aktiviert werden.

Die in kleinen Kreisen endenden Linien stellen hemmende Verbindungen dar, hier die NU-Links. Die negativen Gewichte an diesen Links sind so groß, daß eine U-Unit $\langle x \sim y \rangle$ in jedem Fall deaktiviert wird, sobald die entsprechende NU-Unit $\langle x \nsim y \rangle$ aktiv wird.

Man beachte die unterschiedliche Struktur in der linken und rechten Hälfte des Bildes. Die Verschaltung der drei U-Units untereinander ist "punktsymmetrisch", die der U-Unit mit den beiden NU-Units dagegen nur "achsensymmetrisch".

Nicht eingezeichnet sind noch je zwei T-Units, die die Implikationen $x \sim y \wedge y \nsim z \Rightarrow x \nsim z$ und $x \nsim y \wedge y \sim z \Rightarrow x \nsim z$ sowie deren Umkehrungen realisieren.

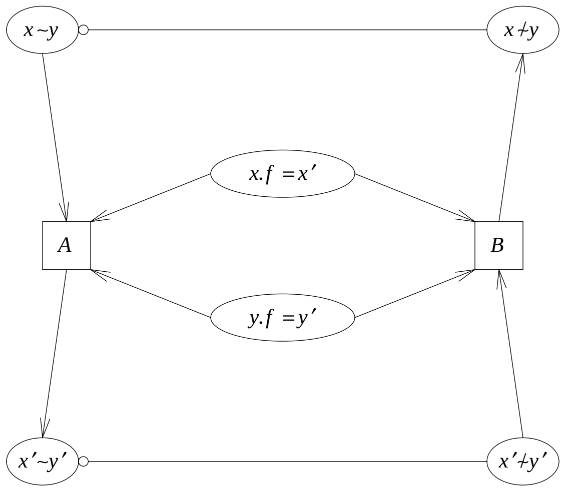
Auf ganz analoge Weise müssen sich Nicht-Äquivalenzen durch den Graphen von "unten nach oben" fortpflanzen. Dies ergibt sich aus (42)(a) durch Kontraposition. Somit aktivieren $\langle x.f = x' \rangle$, $\langle y.f = y' \rangle$ und $\langle x' \nsim y' \rangle$ über die *BU-Unit* $\langle x.f = x' \nsim y.f = y' \rangle$ die U-Unit $\langle x \nsim y \rangle$. Die dabei verwendeten vier Links heißen entsprechend *BU-Links*.

TD- und BU-Links sind in Abb. (50) zusammengefaßt. Anschaulich gesprochen fließt die Information über Knotenäquivalenzen in Form von Aktivation an den U-Units von den Wurzeln der Strukturen zu den terminalen (atomaren) Knoten. Antiparallel dazu fließt die Information über Nicht-Äquivalenz von Knoten von den terminalen Knoten aufwärts zu den Wurzeln, indem die NU-Units aktiv werden.

Die Initialzündung für den Aktivationsfluß von oben nach unten geht von den Paaren von Wurzeln aus, deren Unifizierung vom "Anwender" beabsichtigt ist. Da allerdings die NU-Units ihre zugehörigen U-Units immer überstimmen können, bleiben die von außen aktivierten Wurzeläquivalenzen nur aktiv, wenn sich keine gegenteilige Information ergibt.

Wo aber entsteht die "Initialzündung" für die Ausbreitung der Nicht-Äquivalenzen? Genau dafür sind die Verträglichkeitsbedingungen (b) und (c) verantwortlich. (42)(b) besagt, daß verschiedene atomare Knoten in jedem Fall in unterschiedlichen Äquivalenzklassen liegen müssen. Dies bedeutet, daß die NU-Units der Form $\langle u \nsim v \rangle$ mit $u, v \in V$ immer 1-Aktivation und ihre Pendants $\langle u \sim v \rangle$ (qua NU-Link) immer 0-Aktivation haben müssen.

(50)



Top-Down- und Bottom-Up-Fortpflanzung von Äquivalenzen und Nicht-Äquivalenzen. Die Units A und B realisieren jeweils die Konjunktion ihrer drei Eingangs-Links. Sind die beiden E-Units aktiv, kann Aktivation auf der linken Seite von oben nach unten, auf der rechten Seite von unten nach oben fließen, allerdings nicht gleichzeitig: Die hemmenden NU-Links stoppen die Aktivation auf der linken Seite, sobald die rechte Seite aktiv wird.

Wie bereits diskutiert kann aber auf Units mit konstanter Aktivation grundsätzlich verzichtet werden. Units mit konstanter 0-Aktivität können ersatzlos entfallen, da sie keinerlei Aktivation zu anderen Units beitragen. Units mit konstanter 1-Aktivität können entfallen, wenn die Schwellenwerte der Units, die von ihnen Aktivation empfangen, um die entsprechenden Gewichte vermindert werden.

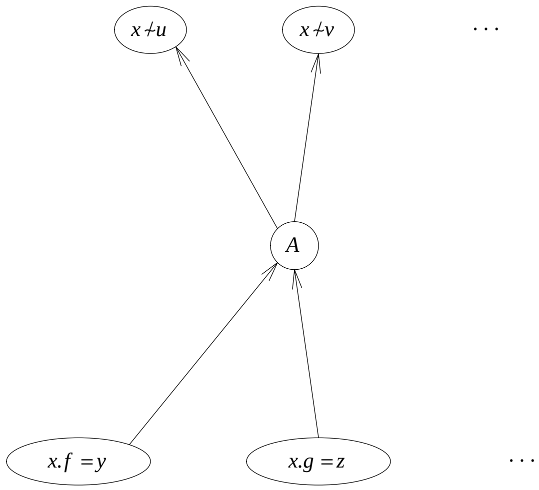
Schließlich verlangt Bedingung (42)(c), daß ein atomarer Knoten nicht gemeinsam mit einem Knoten, von dem Kanten abgehen, in einer Klasse vorkommt. Die Bedingung kann dadurch umgesetzt werden, daß E-Units der Form $\langle x.f = y \rangle$ alle NU-Units $\langle x \not\sim v \rangle$, $v \in V$, aktivieren. Um dabei nicht jede E-Unit mit jeder NU-Unit verbinden zu müssen, kann man das Verbindungsschema in Abb. (51) verwenden. Alle E-Units, die Kanten von demselben nichtatomaren Knoten repräsentieren, aktivieren disjunktiv eine sog. *NA-Unit*, die ihrerseits sämtliche NU-Units aktiviert, die für die Nicht-Äquivalenz mit einem atomaren Knoten stehen. Die Verbindungen von und zu der NA-Unit werden *NA-Links* genannt.

Auf diese Weise erhält man ein Netz, das nur noch solche Gleichgewichtszustände besitzt, die verträgliche Äquivalenzrelationen repräsentieren.

4.3.2.5 Größe des Netzes

Tabelle (52) faßt zusammen, wieviele Units und Links das Netz benötigt, um f-Strukturen mit einer bestimmten Anzahl von Knoten und Kanten zu repräsentieren und zu verarbeiten. Dabei bedeutet n_N die Anzahl der Knoten und n_V die Anzahl der (in n_N enthaltenen) atomaren Knoten, d. h. $n_V \le |V|$. $n_F \le |F|$ ist die Anzahl der Merkmale, die vorkommen, und n_E ist die Anzahl der Kanten, die in allen f-Strukturen insgesamt (potentiell) vorkommen. Damit sind auch Kanten gemeint, die im Rahmen einer Anwendung nicht immer vorhanden sind, die also je nach Bedarf von außen “eingeschaltet” werden können. Solche potentiellen Kanten gibt es maximal

(51)



Ableitung von Nicht-Äquivalenz mit atomaren Knoten. A ist eine NA-Unit, die sicherstellt, daß Knoten, von denen Kanten abgehen, nicht äquivalent zu einem atomaren Knoten sind. Sie arbeitet disjunktiv, d. h. wird aktiv, sobald einer der eingehenden NA-Links Aktivation liefert.

(52)

Objekttyp	*Aufwand*
E-Units	n_E
U-Units NU-Units NU-Links	$\approx \frac{1}{2} n_N^2$
T-Units	$\approx \frac{3}{2} n_N^3$
T-Links	$\approx \frac{9}{2} n_N^3$
TD-Units BU-Units	$\approx \frac{n_E^2}{n_F}$
TD-Links BU-Links	$\approx 4 \frac{n_E^2}{n_F}$
NA-Units	$n_N - n_V$
NA-Links	$(n_N - n_V) n_V + n_F$

Aufwand für Units und Links. Für U-Units, T-Units und T-Links sind nur die Terme höchster Ordnung angegeben. Bei TD-Units und TD-Links kann nur ein Durchschnittswert angegeben werden, da die genaue Anzahl von den konkret vorhandenen Kanten abhängt. (Gesucht ist hier die Anzahl von Kantenpaaren mit übereinstimmendem Merkmal.)

$(n_N - n_V) n_F n_N$, von denen allerdings in einer konkreten Anwendung immer nur ein sehr geringer Anteil wirklich benötigt wird. Dies liegt auch daran, daß die meisten dieser Kantenkombinationen keine wohldefinierten verallgemeinerten f-Strukturen ergeben. Die Kantenzahl einer konkreten f-Struktur mit n nicht-atomaren Knoten kann $n\,|F|$ nicht überschreiten, da von jedem

Knoten maximal pro Merkmal eine Kante ausgehen kann.

Wie zu vermuten schlagen die Units und Links, die die Transitivität der repräsentierten Relation garantieren, mit dem höchsten Aufwand zu Buche; T-Units und T-Links wachsen mit $O(n_N^3)$. In Abschnitt 4.3.3.3 wird besprochen, wie man den Aufwand für Units und Links im Fall einer konkreten Anwendung durch eine Vorverarbeitung des Netzes erheblich reduzieren kann.

4.3.2.6 Anwendung des Netzes

Für eine konkrete Anwendung muß man ausreichend viele E-Units vorsehen, um alle vorkommenden f-Strukturen bei Bedarf speichern zu können. Man setzt dann die E-Units, die für vorhandene Kanten stehen, auf 1 und alle anderen auf 0. E-Units sind reine Eingabe-Units, d. h. ihr Zustand verändert sich im Lauf der Verarbeitung nicht. E-Units haben lediglich die Funktion, "Schleusen" für die *bottom-up* bzw. *top-down* fließende Aktivation zu öffnen (vgl. Abb. (50)).

Die eigentliche Verarbeitung findet durch Wechselwirkung der U- und NU-Units statt. Diese und alle anderen Units werden mit 0 initialisiert. Bevor das Netz nun gestartet wird, müssen alle beabsichtigten Unifikationen durch Aktivation der entsprechenden U-Units von außen (oder durch entsprechendes Senken des Schwellenwertes) aktiviert werden. Wie beschrieben, können sich nun Äquivalenzen und Nicht-Äquivalenzen simultan im Netz ausbreiten. Ist eine Unifikation möglich, wird sich nach einer Zeit, die proportional zur Tiefe der bearbeiteten f-Strukturen ist, ein Gleichgewichtszustand einstellen, in dem die Gesamtheit der U-Units in Form einer Äquivalenzrelation nach Satz (43) das Unifikat repräsentiert.[44] Das Gelingen der Unifikation ist daran zu erkennen, daß im Endzustand die anfangs aktivierten Äquivalenzen bestehen bleiben.

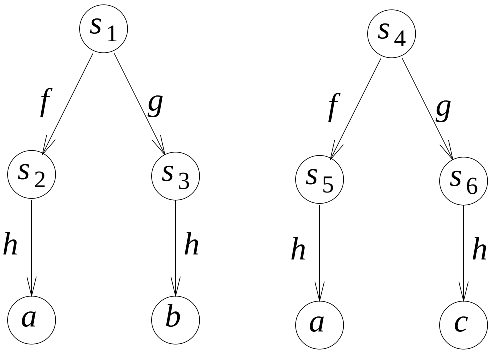
Die Initialisierung aller Units mit 0 stellt sicher, daß im Falle einer erfolgreichen Unifikation nur diejenigen Äquivalenzen abgeleitet werden, die sich aus der Äquivalenz der zu unifizierenden Wurzeln ergeben. Auf diese Weise ist (völlig analog zu Algorithmus (47)) gewährleistet, daß die feinste aller möglichen Äquivalenrelation erzeugt wird.

Nun sollte das Netz nicht nur erfolgreiche Unifikation anzeigen, sondern auch zuverlässig feststellen, wann die verarbeiteten f-Strukturen nicht unifizierbar sind. Eine analytische Untersuchung des Netzes in diesem Fall ist schwierig, weshalb man hier vor allem auf Erfahrungen aus Simulationsexperimenten angewiesen ist.

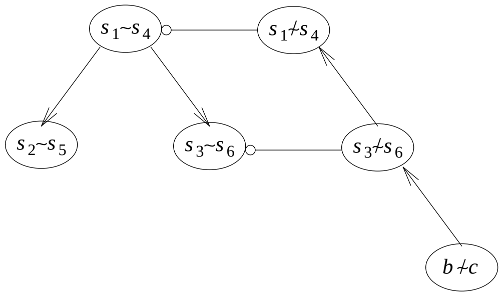
In den meisten Fällen bildete sich bei nicht unifizierbaren Strukturen ebenfalls ein Gleichgewichtszustand aus, in dem die anfangs aktivierten U-Units durch die entsprechenden NU-Units deaktiviert worden sind. Dieser Fall tritt mit Sicherheit dann ein, wenn baumartige f-Strukturen verarbeitet werden, die aufgrund atomarer Knoten inkompatibel sind, wie in Beispiel (53). Problematisch kann das Verhalten des Netzes jedoch werden, wenn sich Inkompatibilität indirekt durch Transitivität ergibt, wie in Beispiel (54). In (54) bewirkt eine Rückkoppelung über T-Links und T-Units, daß ein Teil des Netzes in Schwingung gerät.

[44] Dafür muß die Aktivation einmal die Strukturen von der Wurzel bis zu den atomaren Knoten (und umgekehrt) durchqueren. Aufgrund des probabilistischen Charakters der Aktivationsausbreitung läßt sich jeoch keine exakte Aussage über die dafür benötigte Zeit machen.

(53) (a)



(b)



Inkompatible f-Strukturen, die in einen Gleichgewichtszustand führen. (a) stellt zwei f-Strukturen dar, die aufgrund eines inkompatiblen Merkmals g nicht unifizieren können. (b) zeigt einen Ausschnitt aus dem dazugehörigen Netz. TD- und BU-Links sind ohne die vermittelnden TD- bzw. BU-Units dargestellt. Die Information über Nicht-Äquivalenz breitet sich von den Endknoten direkt zu den Wurzeln aus und schafft einen Gleichgewichtszustand, da die entsprechenden Äquivalenzen (U-Units) permanent unterdrückt werden. In diesem Endzustand sind alle Units deaktiviert außer $\langle s_0 \not\sim s_4 \rangle$ und $\langle s_3 \not\sim s_6 \rangle$. (Die Unit $\langle b \not\sim c \rangle$ is nur “virtuell” vorhanden, da sie konstant aktiviert ist. Diese Aktivation ist in den Schwellenwert der angeschlossenen BU-Unit einbezogen.)

Dadurch wird dann auch die NU-Unit an der Wurzel abwechselnd an- oder ausgeschaltet. Trotzdem kann man auch hier an der Wurzel erkennen, daß die Unifikation nicht gelungen ist.

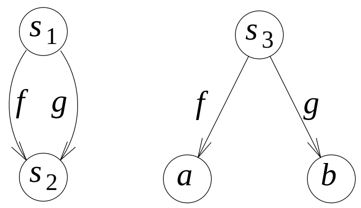
4.3.2.7 Suchen von Unifikationen

Wie man sieht, gelangt das Netz nicht in jedem Fall in einen Gleichgewichtszustand. Dies liegt daran, daß die Verbindungsstruktur einerseits Zyklen enthält und andererseits nicht symmetrisch ist. Man kann zeigen, daß Netze, deren Gewichte symmetrisch sind (darunter fallen auch die klassischen Hopfield-Netze), eine *Energiefunktion*

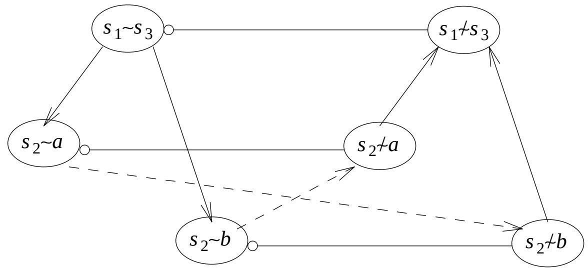
(55)
$$E(t) = -\sum_{i,j=1}^{N} w_{ij}\, a_i(t) a_j(t) + \sum_{i=1}^{N} \Theta_i\, a_i(t)$$

minimieren und daher selbst bei zyklischen Verbindungen immer gegen einen Gleichgewichtszustand konvergieren [Almeida1987a]. Die Bezeichnung “Energiefunktion” ergibt sich aus der Analogie zu verschiedenen physikalischen Systemen. Im Zusammenhang mit constraint-satisfaction-Problemen könnte man E auch (mit umgekehrten Vorzeichen) als *Gütefunktion* betrachten, die mißt, wie gut die Aktivationen a_i mit den Zwängen der Verbindungsstruktur in Einklang stehen.

(54) (a)



(b)



Inkompatible f-Strukturen, die zu Schwingungen führen. Die f-Strukturen in (a) sind aufgrund der Transitivität der Äquivalenz inkompatibel: s_2 ist mit a bzw. mit b unifizierbar, aber nicht mit beiden gleichzeitig. (b) zeigt, warum das Netz kein Gleichgewicht erreicht. (Die gestrichelt eingezeichneten Verbindungen werden indirekt über T-Links und T-Units hergestellt.) $\langle s_1 \sim s_3 \rangle$ deaktiviert sich selbst über zwei Zyklen, die über $\langle s_1 \not\sim s_3 \rangle$ führen. Solange $\langle s_1 \sim s_3 \rangle$ von außen aktiviert wird, entsteht eine Schwingung.

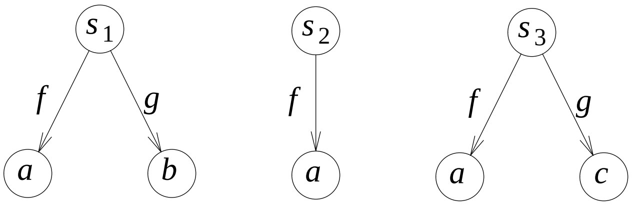
Obwohl also aufgrund der asymmetrischen Gewichte das hier betrachtete Netz zumindest nicht die Funktion E minimiert, ist es doch in der Lage, auch bei nicht unifizierbaren Strukturen Äquivalenzklassen mit “maximaler Güte” zu finden. Dieser Fall tritt typischerweise dann ein, wenn von mehreren zu unifizierenden f-Strukturen zwar einige untereinander, aber nicht alle zusammen unifizierbar sind. Das Netz muß sich dann für eine der möglichen Paarungen entscheiden, wie durch Beispiel (56) illustriert wird. Solche Entscheidungen zwischen mehreren sich ausschließenden Unifikationen werden zufällig getroffen. Sie hängen davon ab, welche der konkurrierenden (sich gegenseitig hemmenden) Units zuerst ihre Aktivität neu berechnen.

Man kann das Netz also nicht nur benutzen, um eine vorgegebene Unifikation zu *prüfen*, sondern auch, um in einer Menge von f-Strukturen mögliche Unifikationen zu *suchen*. Man geht dabei so vor, daß alle möglichen Paarungen von Wurzeln aktiviert werden. Das Netz deaktiviert diejenigen U-Units, die unmöglichen (inkompatiblen) Paarungen entsprechen. Dabei kann es sein, daß das Netz Entscheidungen treffen muß zwischen mehreren alternativen, aber sich ausschließenden Paarungen.

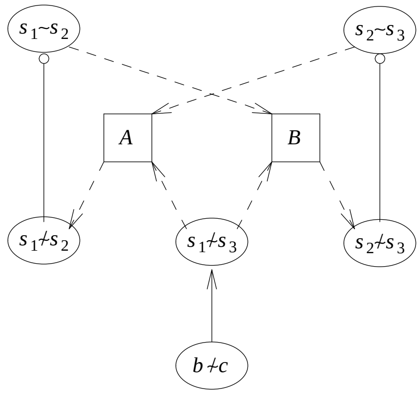
Die Simulationen haben gezeigt, daß solche Entscheidungen manchmal erst nach einer Phase der “unschlüssigen Suchens” zustande kamen, in der verschiedene Teilstrukturen unifiziert wurden, ohne daß eine globaler Konsens gefunden wurde. Manchmal wurde eine mögliche Unifikation nicht gefunden, weil sich Teile des Netzes gegenseitig stabilisierten, und damit die Abhängigkeit von der Aktivation durch die Umgebung unterlaufen konnten. Dieses Phänomen und eine (partielle) Abhilfe wird nun im einzelnen besprochen.

Während der Unifikation mehrerer f-Strukturen tritt relativ häufig der Fall ein, daß vorübergehend drei U-Units gleichzeitig aktiv werden, die sich über T-Links und T-Units

(56) (a)



(b)



Konkurrierende Unifikationen. Die drei f-Stukturen in (a) können nicht gleichzeitig unifizieren, sondern die beiden äußeren nur paarweise mit der mittleren. (b) zeigt einen Ausschnitt aus dem entsprechenden Netz. $\langle s_1{\sim}s_2\rangle$ und $\langle s_2{\sim}s_3\rangle$ hemmen sich gegenseitig über T-Links und die T-Units A und B. Wenn jedoch eine der beiden U-Units einmal aktiv und die andere inaktiv ist, entsteht ein Gleichgewicht, in dem die aktive U-Unit dauerhaft die andere unterdrücken kann. Zum Beispiel gewinnt $\langle s_1{\sim}s_2\rangle$ die Oberhand, wenn in der asynchronen Simulation B und anschließend $\langle s_2{\not\sim}s_3\rangle$ ihre Aktivation vor A und $\langle s_1{\not\sim}s_2\rangle$ berechnen.

gegenseitig stützen (vgl. Abb. (49)). Die Aktivation von drei Units $\langle x{\sim}y\rangle$, $\langle y{\sim}z\rangle$ und $\langle x{\sim}z\rangle$ kann sich gegenseitig erhalten, selbst wenn die Quelle, aus der sie ursprünglich ihre Aktivation erhielten, nicht mehr aktiv ist. Solche Gruppen von sich selbst stabilisierenden Units werden als *stabile Koalitionen* [Feldman1982a] bezeichnet und können die gesamte Operation des Netzes verfälschen, da unmotivierte Äquivalenzen wiederum unmotivierte Nicht-Äquivalenzen nach sich ziehen usf. Die Korrektheit der Verarbeitung des Netzes beruht aber darauf, daß jede Äquivalenz sich auf die Äquivalenz der zu unifizierenden Wurzeln zurückführen läßt und jede Nicht-Äquivalenz auf eine Inkompatibilität von atomaren Werten zurückgeht.

Wie können solche Koalitionen verhindert werden? Wenn aufgrund der Transitivität zwei U-Units die Aktivierung einer dritten U-Unit verursachen, muß verhindert werden, daß diese dritte Unit nun ihrerseits die beiden ursprünglichen Units qua Transitivität aktiviert, da sonst die Rückkoppelung perfekt ist. Die Rückkoppelung wird unterdrückt, indem die T-Units, die zwischen denselben drei U-Units vermitteln, sich gegenseitig so stark hemmen, daß immer nur eine von ihnen aktiv sein kann. Diese eine T-Unit reicht immer aus, um die Transitivität zu vermitteln, verhindert aber gleichzeitig, daß ein andere T-Unit aktiv wird und so eine Rückkoppelung herstellt.

Im Fall von Abb. (49) werden die Units A, B und C paarweise mit symmetrischen, stark hemmenden Links versehen. Ebenfalls wechselseitig hemmend verbunden werden D und E, da hier ein ähnliches Verhalten auftritt, wenn $\langle x \sim z \rangle$ konstant aktiv ist. Durch diese zusätzlichen hemmenden Verbindungen konnten die vorher beobachteten stabilen Koalitionen völlig eliminiert werden.[45]

4.3.3 Verarbeitung von Unifikationsgrammatiken

4.3.3.1 Grammatikregeln als verallgemeinerte f-Strukturen

Um das oben beschriebene Netz auf die Generierung aus Unifikationsgrammatiken anwenden zu können, muß die Erzeugung einer c- bzw. f-Struktur als eine Folge von Unifikationen dargestellt werden. In Abschnitt 3.2.2 wurde gezeigt, wie man c- und f-Strukturen inkrementell aus Grammatikregeln aufbauen kann, wobei die f-Struktur durch die destruktive Unifikation von Teilstrukturen entsteht, die aus den einzelnen Regeln abgeleitet werden. Wir werden dieses Verfahren jetzt auf die Generierung anwenden.

Die Idee des Verfahrens ist die, daß jede Regel einen Teil der c- bzw. f-Struktur beinhaltet, die letztlich den fertigen Satz ausmacht. Bei der Anwendung einer Regel werden diese Strukturfragmente zusammengesetzt, wobei im Falle der c-Struktur lediglich Unterbäume aneinander gefügt werden, während f-Strukturfragmente durch destruktive Unifikation verknüpft werden.

Anhand der Beispielgrammatik aus Abschnitt 3.3.3 soll dieses Verfahren nun erläutert werden. Abb. (57) zeigt die c- bzw. f-Struktur-Fragmente, die aus den Regeln der Grammatik abgeleitet wurden. Die c-Strukturfragmente erhält man einfach, indem man der linken Seite einer Regel einen c-Strukturknoten zuordnet und für jede Variable in der rechten Seite einen Sohnknoten erzeugt. Die f-Strukturfragmente erhält man aus den Gleichungen zu einer Regel, indem man die Gleichungen als Pfadspezifikationen für eine f-Struktur interpretiert. Die kleinste verallgemeinerte f-Struktur, die den Gleichungen genügt, ist das gesuchte Fragment.

Der Satz

Peter liebt Maria

entsteht mitsamt seiner f-Struktur, wenn folgende destruktive Unifikationen ausgeführt werden: $s_2 \sim s_{11}$, $s_3 \sim s_6$, $s_7 \sim s_{17}$ und $s_8 \sim s_{14}$.

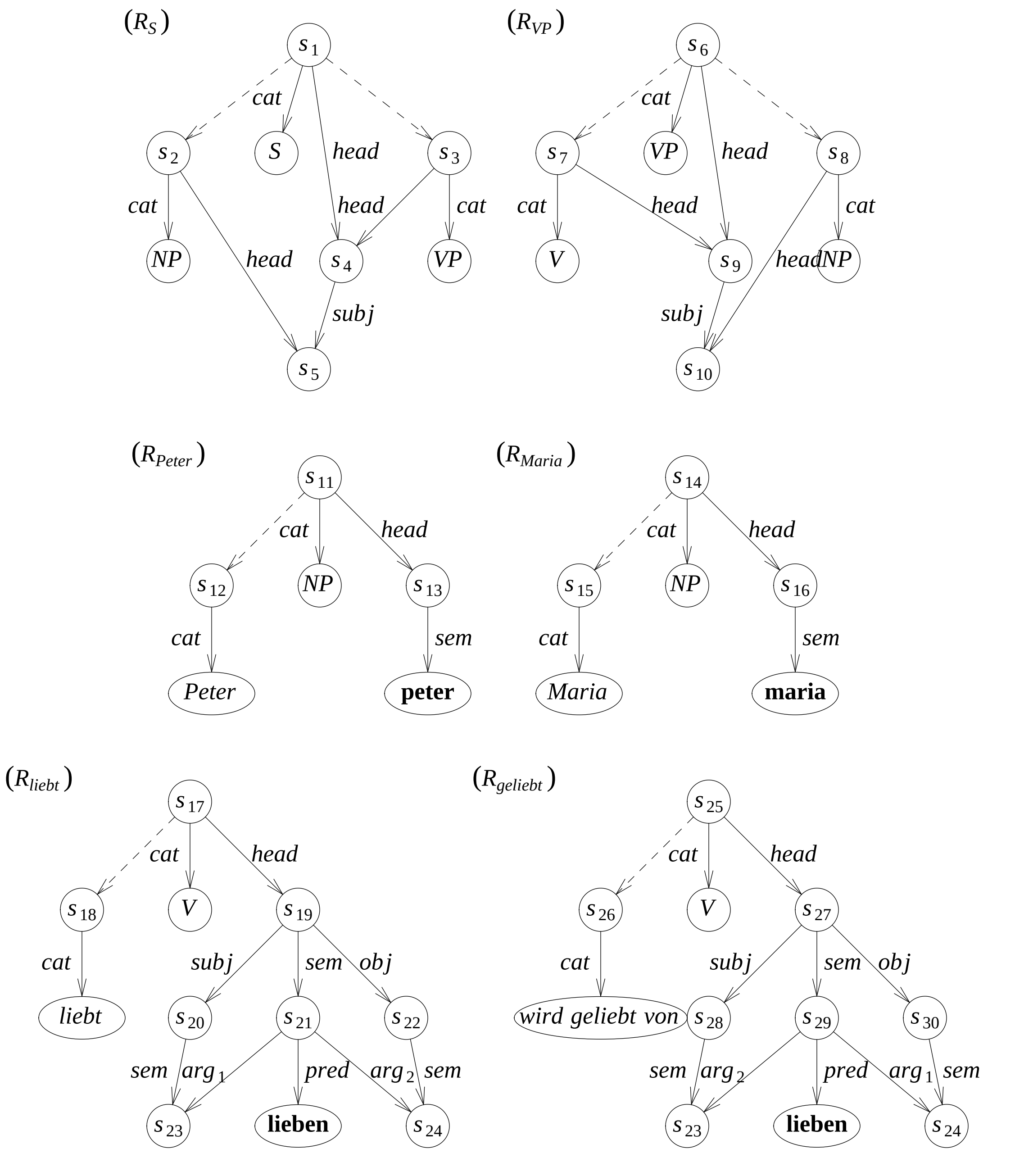
4.3.3.2 Generierung im Netz

Der erste Schritt für das Generieren aus einer Unifikationsgrammatik mithilfe des vorgestellten Netzes ist das Übersetzen der Grammatik in eine Menge von f-Strukturfragmenten wie oben beschrieben. Da die Generierung der f-Struktur auf einer *destruktiven* Verknüpfung der Strukturfragmente beruht, muß für jede Regelanwendung eine neue Kopie des Fragments verwendet werden. Dies bedeutet, das die Größe des Netzes nicht nur mit der verwendeten Grammatik, sondern auch mit der maximalen Tiefe der zu generierenden Strukturen wächst. Allerdings können sich mehrere Kopien einer Regel die verwendeten E-Units teilen, da diese für alle Kopien identisch sind und nur als Eingabe-Units fungieren.

---

[45] Allerdings sind zumindest theoretisch immer noch Koalitionen von fünf und mehr U-Units denkbar, jedoch traten diese spontan während keiner der Simulationen auf.

(57)



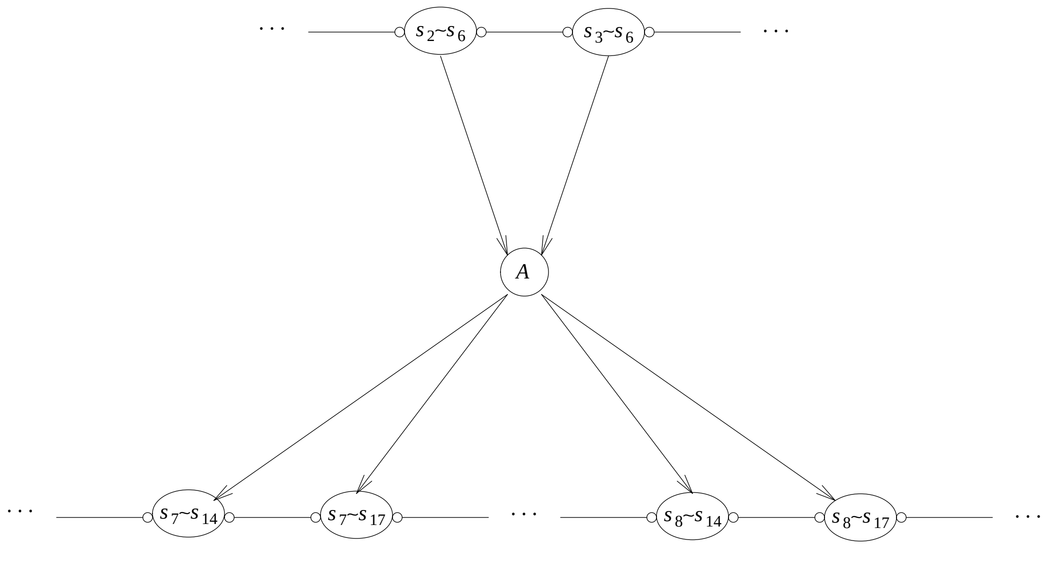
c- und f-Strukturfragmente entsprechend den Regeln der Beispielgrammatik. Teile der c-Struktur sind gestrichelt, Teile der f-Struktur durchgehend gezeichnet. Entsprechende Knoten aus beiden Strukturen wurden identifiziert.

Neben der eigentlichen Grammatik muß auch noch die Startregel der augmentierten Grammatik in das Netz aufgenommen werden, damit eine Satzspezifikation in Form einer f-Struktur (vgl. Satz (32)) eingegeben werden kann. Für diese Startregel sollten so viele E-Units vorgesehen werden, daß alle sinnvollen Spezifikationen durch aktivieren bzw. deaktivieren von Kanten eingegeben werden können. Tatsächlich ist das f-Strukturfragment aus der Startregel der augmentierten Grammatik das einzige Fragment, das sich kurzfristig (von einer Generierung zur nächsten) ändert, während alle übrigen Fragmente – und damit die sie darstellenden E-Units – konstant bleiben.

Die c-Struktur geht durch eine Erweiterung der Verbindungsstruktur in das Netz ein. Durch diese Erweiterung soll erreicht werden, daß die erfolgreiche Unifikation eines Fragments (entsprechend der erfolgreichen Anwendung einer Regel) die Unifikation weiterer Fragmente (entsprechend den rechten Seiten dieser Regel) auslöst. In jedem f-Strukturfragment gibt es einen Knoten, der der linken Seite der Regel entspricht (sog. *L-Knoten*) und mehrere Knoten, die den Variablen der rechten Seite entsprechen (*R-Knoten*). Sei nun eine Regel gegeben mit x_0 als L-Knoten und $x_1, \ldots, x_n$ als R-Knoten. Sobald x_0 mit irgend einem anderen R-Knoten y unifiziert hat, muß das Netz versuchen, auch $x_1, \ldots, x_n$ zu unifizieren. Dies geschieht über Links, mit denen $\langle x_0 \sim y \rangle$ alle U-Units $\langle x_i \sim z \rangle$ aktiviert, wobei $i = 1, \ldots, n$ und z ein beliebiger L-Knoten ist.

Darüberhinaus muß sich das Netz bei der Unifikation eines R-Knotens für einen L-Knoten entscheiden. Dies wird durch hemmende Verbindungen zwischen konkurrierenden U-Units erreicht. Umgekehrt kann ein L-Knoten nicht mit mehreren R-Knoten unifizieren, weil dies einer mehrfachen Verwendung des zugehörigen f-Strukturfragments entsprechen würde. Auch diese Entscheidung wird durch wechselseitig hemmende Verbindungen erzwungen. Die zusätzlichen Links, die sich aus der c-Struktur ergeben, sind in Abb. (58) dargestellt, und zwar anhand eines Ausschnitts aus dem Netzes, das aus der Beispielgrammatik erzeugt wurde (vgl. (57)).

(58)



Durch die c-Struktur induzierte Links. Die symmetrisch hemmenden Links erzwingen, daß ein L-Knoten (hier s_6) mit nur einem R-Knoten unifiziert bzw. daß ein R-Knoten (s_7, s_8) nur mit einem L-Knoten unifiziert. Die übrigen Links leiten sich aus Regel (R_{VP}) ab und sorgen dafür, daß die R-Knoten der Regel unifiziert werden, sobald der L-Knoten unifiziert wurde. Unit A dient wieder dazu, die Anzahl der benötigten Links zu reduzieren.

Insgesamt bewirken diese neu geschaffenen Verbindungen, daß das Netz versucht, von der Wurzel der c-Struktur ausgehend f-Strukturfragmente zu unifizieren, bis terminale Regeln erreicht sind. Man kann die Generierung daher anstoßen, indem man die Unifikation der rechten Seite der Startregel mit allen L-Knoten in Gang setzt (auf die übliche Weise durch externe Aktivierung der entsprechenden U-Units).

4.3.3.3 Optimierung des Netzes

Mit einer einfachen Methode läßt sich ein beträchtlicher Teil der Units und Links des Netzes eliminieren und die Verarbeitung des Netzes optimieren. Diese Optimierung beruht darauf, daß eine spontane Aktivationsausbreitung ohne Eingabespezifikation und externer Aktivierung bereits eine Vorverarbeitung der verwendeten Grammatik erlaubt.

Zwar kann noch nicht festgestellt werden, welche Knoten mit welchen unifizieren (äquivalent sind), aber inkompatible atomare Werte sorgen dafür, daß bereits ein Großteil der *unmöglichen* Unifikationen (Nicht-Äquivalenzen) berechnet werden kann. Zum Beispiel wird so festgestellt, daß die meisten der Unifikationen zwischen L-Knoten und R-Knoten aufgrund inkompatibler *cat*-Merkmale nicht in Frage kommen. Man erhält so eine Reihe von NU-Units, die konstante 1-Aktivation haben, und entsprechende U-Units mit konstanter 0-Aktivation. Wie bereits besprochen, können Units mit konstanter Aktivation prinzipiell aus einem Netz eliminiert werden, ebenso wie alle Links, die von diesen Units ausgehen oder bei diesen enden.

Auch können alle E-Units (bis auf die zur Eingabe einer Spezifikation benötigten) durch “festverdrahtete” TD- bzw. BU-Links ersetzt werden, wobei auch die zugehörigen TD- und BU-Units entfallen (vgl. Abb. (50)).

4.3.3.4 Erfahrungen und Bewertung

Die Größe der mit Simulationen untersuchten Grammatiken war stark eingeschränkt, da sich schon mit wenigen Regeln erhebliche Knoten- und Kantenmengen ergeben; der maximalen Anzahl von Units und Links waren relativ enge Grenzen gesetzt (insgesamt ca. 4000 Objekte). Es war also keinesfalls möglich, realistische Grammatiken zu testen; die Beispielgrammatik entspricht in etwa dem äußersten dessen, was noch realisierbar war. Obwohl die Simulationen also keine signifikanten Aussagen erlauben, lassen sich doch einige theoretische Überlegungen zum Verhalten des Netzes anstellen.

Das Netz in der beschriebenen Form muß Entscheidungen fällen, wenn mehrere Regeln zur Auswahl stehen. Sind mehrere Regeln nach der Unifizierbarkeit ihrer f-Stukturen anwendbar, werden solche Entscheidungen dem in der asynchronen Simulation enthaltenen Zufall überlassen.

Ist die Eingabespezifikation ausreichend, um diese Entscheidungen zu kanalisieren, erzeugt das Netz in jedem Fall einen Satz, der der Spezifikation entspricht, sofern dies nach der Grammatik möglich ist.

In der kleinen Beispielgrammatik werden beliebige Spezifikationen korrekt in entsprechende Regelanwendungen umgesetzt. Dies liegt daran, daß die betrachtete Grammatik keine Sackgassen erzeugt. Problematisch wird die Generierung, wenn z. B. eine zweite Regel für die Expansion einer Verbalphrase vorgesehen ist:

$$(R_{VP}{}') \qquad VP \rightarrow V \\ \qquad\qquad VP.head = V.head$$

Mit dieser Regeln kann das Netz in eine Sackgasse laufen, wenn es bei der Unifikation des VP-Knoten auf der rechten Seite von (R_S) zufällig die falsche Entscheidung trifft.

Verschiedene Lösungsansätze sind hierfür denkbar. Der einfachste besteht darin, die Grammatik so zu erweitern, daß anhand der Spezifikation entschieden werden kann, welche Regel jeweils anzuwenden ist. Wenn obige Regel erweitert wird zu

$$(R_{VP}{}') \qquad \begin{aligned} VP &\to V \\ VP.head &= V.head \\ VP.head.obj &= undef \end{aligned}$$

wird die Sackgasse vermieden. Diese Methode stellt jedoch eine zusätzliche Forderung an die Grammatik und die verwendeten Spezifikationen, die nur bedingt zu erfüllen ist. Sie verlangt, daß in der Spezifikation bereits die Entscheidungen des Generierungsprozesses vorausgesehen werden müssen, was im allgemeinen nicht gewährleistet ist.

Ein zweiter Ansatz besteht darin, das Netz simultan nach allen möglichen Expansionen suchen zu lassen. Dafür muß der Zwang, daß ein L-Knoten immer nur mit einem R-Knoten unifizieren kann, aufgehoben werden. Dies kann durch Weglassen der entsprechenden hemmenden Verbindungen erreicht werden. Diese Methode ist vor allem deshalb unbefriedigend, weil dadurch die Größe des Netzes exponentiell mit der Tiefe der generierten c-Strukturen erhöht wird.

Ein dritter Ansatz bestünde darin, ein Backtracking der Generierung durch eine externe Ablaufsteuerung zu realisieren. Man würde damit von dem Versuch abweichen, den gesamten Generierungsprozeß einheitlich in Form eines neuronalen Netzes abzuwickeln. Allerdings wurden z. B. in [Touretzky1985a] Verfahren entwickelt, mit dem eine allgemeine Ablaufsteuerung konnektionistisch implementiert werden kann. Ein solches System würde dann das Unifikationsnetz als eine Art Unterprogramm aufrufen. Das Unbefriedigende an diesem Ansatz ist die Tatsache, daß man nicht mehr allein mit der lokalen Interaktion von Units (einer der grundlegenden Charakteristika des konnektionistischen Prinzips) auskommt, sondern globale Überwachungsmechanismen mit traditionellen oder konnektionistischen Mitteln einführt.

Betrachtet man die Generierung eines Satzes insgesamt als ein constraint-satisfaction-Problem, bei dem die Bedingungen durch die Grammatik und die Spezifikation gegeben sind, dann stellen Sackgassen lokale Maxima einer Gütefunktion dar, entsprechend den lokalen Minima der Energiefunktion (55). Mit der Anwendung einer Regel, die in eine Sackgasse führt, gelangt man zwar auf der E-Hyperebene im Zustandsraum des Netzes noch eine Stück weit “bergab”, hat damit aber den Weg in globales Minimum (das einer vollständigen f-Struktur entspricht) verlassen.

Erfüllt ein Netz gewisse Voraussetzungen, zu denen in erster Line wieder die Symmetrie der Gewichte gehört, kann man solche lokalen Minima durch stochastische Methoden umgehen. Dies wird in sog. Boltzmann-Maschinen beim *simulated annealing* gemacht [Hinton1986a]. Dieser Ansatz würde allerdings voraussetzen, daß es gelingt, Unifikation durch symmetrische Netze zu implementieren. Vermutlich kann dabei die Unifikation nicht in ihrer hier verwendeten, diskreten Form beibehalten werden. Diese Perspektive wird im nächsten Abschnitt ausführlicher diskutiert werden.

4.4 Zusammenfassung und Ausblick

Ausgangspunkt für die Arbeit war die Frage, inwieweit sich traditionelle Grammatikformalismen mit dem konnektionistischen Verarbeitungsparadigma vereinbaren lassen. Als eine der elegantesten und in der Sprachgenerierung populärsten Grammatikmodelle wurde dabei der unifikationsbasierte Formalismus betrachtet.

Ein zentrales Problem – auch unabhängig von der Anwendung in der Sprachgenerierung – war dabei die algebraische Operation der Unifikation. Basierend auf einer Charakterisierung der Unifikation als constraint-satisfaction-Problem wurde ein Netz entwickelt, das die Operation bei maximaler Parallelisierung zuverlässig realisiert.

Durch Zurückführung des Generierungsprozesses auf die Unifikation von f-Strukturen war es möglich, dieses Netz mit geringfügigen Erweiterungen auch zur Produktion von (einfachen) Sätzen einzusetzen. Dabei ergaben sich allerdings Einschränkungen, die dieses Verfahren nicht als voll befriedigende Lösung erscheinen lassen.

Die Frage nach der Vereinbarkeit von Konnektionismus und Unifikationsgrammatiken bleibt damit (wie zu erwarten) weiterhin offen. Abschließend sollen einige Richtungen aufgezeigt werden, in die der hier beschriebene Ansatz weiterentwickelt werden kann.

4.4.1 Lernbarkeit

Obwohl es ein berechtigtes Anliegen ist, neuronale Netze mit bestimmten Eigenschaften am Reißbrett zu entwerfen, erhöht es die Plausibilität eines Netzes als kognitivem Modell, wenn das Verhalten des Netzes über einen *Lernmechanismus* aus externen Stimuli abgeleitet werden kann. Ob die Lernmechanismen und Stimuli selbst wiederum plausibel sind oder nur ein künstliches Mittel zum automatischen Generierung der Netzstruktur darstellen, ist dann eine zweite Frage.

Zumindest für die Unifikation sind Trainingsstimuli denkbar, die in Verbindung mit gängigen Lernverfahren (*Backpropagation* [Rumelhart1986b, Almeida1987a]; Boltzmann-Maschinen [Hinton1986a]) die geschilderte Verbindungsstruktur entwickeln.

4.4.2 "Fuzzy" Unifikation

In der vorliegenden Arbeit wurde versucht, die Unifikation exakt so zu implementieren, wie sie in gängigen Formalismen als diskrete algebraische Operation verwendet wird. Hinter der formalen Operation steht aber eine Intuition, die diesen diskreten Charakter nicht unbedingt sinnvoll erscheinen läßt. f-Strukturen können auch "mehr oder weniger" unifizieren, je nachdem, wie kompatibel die in ihnen enthaltenen Informationen sind. Es müßte also möglich sein, die Unifikation so zu verallgemeinern, daß Unifizierbarkeit als ein Kontinuum definiert ist. In Analogie zu einer entsprechenden Mengentheorie könnte man dieses Konzept als *fuzzy unification* bezeichnen.

Analog dazu gibt es auch bei natürlichsprachlichen Sätzen zu gängigen Grammatikkonzepten die Alternative die Zulässigkeit oder Adäquatheit (bezüglich einer Grammatik und einer Spezifikation) als Werte auf einer kontinuierlichen Skala aufzufassen. Es erscheint auch natürlicher, für die Generierung von Sätzen keine optimalen Ergebnisse sondern akzeptable Kompromisse zu erwarten. Diese Art von Problemlösung ist aber für konnektionistische

Modelle wesentlicher typischer als das Treffen von “Alles-oder-Nichts”-Entscheidungen.

4.4.3 Zeitliche Sequentialisierung

Obwohl Sprachverarbeitung ohne Zweifel parallele Verarbeitung voraussetzt, enthält gerade Sprache eine inhärent sequentielle Komponente, die durch den sequentiellen Charakter des akustischen Mediums bestimmt wird. Die sequentielle Verarbeitung in neuronalen Netzen ist z. Z. noch relativ wenig untersucht. Gerade diese Fähigkeit wird aber wohl bei einer realistischen Realisierung von Sprachgenerierung eine wesentliche Rolle spielen. Es ist nicht anzunehmen, daß – wie in dem hier vorgestellten Netz – ein längerer Satz vollig simultan erzeugt wird. Plausibler ist es, die zeitliche Struktur soweit wie möglich auszunutzen, um möglichst wenig parallele Verarbeitungkapazität zu binden.

Im Falle des vorgestellten Netzes wäre es vorstellbar, den Aktivationsfluß so zu kanalisieren, daß “von links nach rechts” unifiziert wird. Ist man mit der Unifikation der Grammatikregeln am linken Ende zu den terminalen Knoten gelangt, müssen die bis dahin gefundenen Unifikationen “eingefroren” werden, was einer Äußerung der Wörter entspricht, die damit nicht mehr rückgängig gemacht werden können. Allerdings ist noch unklar, wie ein derartiges sequentielles Verhalten ohne globale Kontrollinstanz realisiert werden kann.

Bibliographie

[Almeida1987a]
Luis B. Almeida: “A learning rule for asynchronous perceptrons with feedback in a combinatorial environment”. In: Maureen Caudill, Charles Butler (Hg.): *Proceedings of the 1st IEEE International Conference on Neural Networks*. San Diego, Calif., Juni 1987, S. II-609–II-618.

[Appelt1983a]
Douglas E. Appelt: “TELEGRAM: A Grammar Formalism for Language Production”. In: *Proceedings of the 8th International Joint Conference on Artificial Intelligence*. Karlsruhe, August 1983, S. 595–599.

[Appelt1985a]
Douglas E. Appelt: *Planning English Sentences*. Cambridge: Cambridge University Press, 1985.

[Ballard1986a]
Dana H. Ballard: “Parallel Logical Inference and Energy Minimization”. In: *Proceedings of the 5th National Conference on Artificial Intelligence*. Philadelphia, Pa., August 1986, S. 203–208.

[Bock1982a]
J. Kathryn Bock: “Toward a Cognitive Psychology of Syntax: Information Processing Contributions to Sentence Formulation”. *Psychological Review* **89**(1), 1982, 1–47.

[Bock1985a]
J. Kathryn Bock: “Coordinating Words and Syntax in Speech Plans”. In: A. Ellis (Hg.): *Progress in the Psychology of Language* (Vol. 3). London: Erlbaum, 1985.

[Charniak1987a]
Eugene Charniak, Eugene Santos: “A Connectionist Context-Free Parser Which is not Context-Free, But Then It is not Really Connectionist Either”. In: *Proceedings of the 9th Annual Conference of the Cognitive Science Society*. Seattle, Wash., Juli 1987, S. 70–77.

[Davey1979a]

Anthony Davey: *Discourse Production*. Edinburgh: Edinburgh University Press, 1979.

[Dell1985a]

Gary S. Dell: “Positive Feedback in Hierarchical Connectionist Models: Applications to Language Production”. *Cognitive Science* **9**, 1985, 3–23.

[Eisele1986a]

Andreas Eisele, Jochen Dörre: “A Lexical Functional Grammar System in Prolog”. In: *Proceedings of the 11th International Conference on Computational Linguistics*. Universität Bonn, Bonn, August 1986, S. 551–553.

[Erben1987a]

Ayse Erben: “Ein natürlichsprachliches Dialogsystem mit Syntax-Semantik-Interaktion basierend auf der LFG-Theorie”. *KI*(4), 1987, 4–9.

[Feldman1982a]

Jerome A. Feldman, Dana H. Ballard: “Connectionist Models and Their Properties”. *Cognitive Science* **6**, 1982, 205–254.

[Gazdar1985a]

Gerald Gazdar, E. Klein, G. K. Pullum, I. A. Sag: *Generalized Phrase Structure Grammar*. Cambridge, Mass.: Harvard University Press, 1985.

[Goldman1975a]

Neil M. Goldman: “Conceptual Generation”. In: Roger C. Schank (Hg.): *Conceptual Information Processing*. Amsterdam: North-Holland, 1975, S. 289–371.

[Halliday1976a]

M. A. K. Halliday: *System and Function in Language*. London: Oxford University Press, 1976.

[Hinton1986a]

Geoffrey E. Hinton, Terrence J. Sejnowski: “Learning and Relearning in Boltzmann Machines”. In: David E. Rumelhart, James L. McClelland (Hg.): *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations*. Cambridge, Mass.: MIT Press, 1986, S. 282–317.

[Hopfield1982a]

J. J. Hopfield: “Neural networks and physical systems with emergent collective computational abilities”. *Proceedings of the National Academy of Sciences USA* **79**, April 1982, 2554–2558.

[Hovy1984a]

Eduard H. Hovy, Roger C. Schank: “Language Generation by Computer”. In: B. G. Bara, G. Guida (Hg.): *Computational Models of Natural Language Processing*. Amsterdam: North-Holland, 1984, S. 165–193.

[Hovy1985a]

Eduard H. Hovy: “Integrating Text Planning and Production in Generation”. In: *Proceedings of the 9th International Joint Conference on Artificial Intelligence*. Los Angeles, Calif., August 1985, S. 848–851.

[Jefferson1974a]
G. Jefferson: “Error correction as an interactional resource”. *Language in society* **3**, 1974, 181–199.

[Kalita1987a]
Jugal Kalita, Lokendra Shastri: “Generation of Simple Sentences in English Using the Connectionist Model of Computation”. In: *Proceedings of the 9th Annual Conference of the Cognitive Science Society*. Seattle, Wash., Juli 1987, S. 555–565.

[Kaplan1982a]
Ronald M. Kaplan, Joan Bresnan: “Lexical Functional Grammar: A Formal System for Grammatical Representation”. In: Joan Bresnan (Hg.): *The Mental Representation of Grammatical Relations*. Cambridge, Mass.: MIT Press, 1982, S. 173–281.

[Kay1980a]
Martin Kay: “Algorithm Schemata and Data Strutures in Syntactic Processing”. Report CSL-80-12. XEROX Palo Alto Research Center, Oktober 1980.

[Kay1984a]
Martin Kay: “Functional Unification Grammar: A formalism for machine translation”. In: *Proceedings of the 10th International Conference on Computational Linguistics*. Stanford, Calif., Juli 1984, S. 75–78.

[Mann1983a]
William C. Mann: “Overview of the Penman Text Generation System”. In: *Proceedings of the 3rd National Conference on Artificial Intelligence*. Washington, D. C., August 1983, S. 261–265.

[McClelland1987a]
James L. McClelland: “The Case for Interactionism in Language Processing”. Technical Report ONR-87-1. Carnegie-Mellon University, Pittsburgh, Pa., 1987.

[McDonald1987a]
David D. McDonald: “Natural-Language Generation”. In: Stuart C. Shapiro, David Eckroth (Hg.): *Encyclopedia of Artificial Intelligence* (Vol. 1). New York, 1987, S. 642–655.

[McKeown1983a]
Kathleen R. McKeown: “Recursion in Text and its Use in Generation”. In: *Proceedings of the 3rd National Conference on Artificial Intelligence*. Washington, D. C., August 1983, S. 270–273.

[McKeown1985a]
Kathleen R. McKeown: *Text Generation. Using discourse strategies and focus constraints to generate natural language text*. Cambridge: Cambridge University Press, 1985.

[McKeown1985b]
Kathleen R. McKeown: “Discourse Strategies for Generating Natural Language Text”. *Artificial Intelligence* **27**, 1985, 1–41.

[Radford1981a]
Andrew Radford: *Transformational Syntax. A student’s guide to Chomsky’s Extended Standard Theory*. Cambridge: Cambridge University Press, 1981.

[Rath1975a]
R. Rath: “Korrektur und Anakoluth im gesprochenen Deutsch”. *Linguistische Berichte* **37**, 1975, 1–12.

[Rumelhart1986a]
David E. Rumelhart, Geoffrey E. Hinton, James L. McClelland: “A General Framework for Parallel Distributed Processing”. In: David E. Rumelhart, James L. McClelland (Hg.): *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations*. Cambridge, Mass.: MIT Press, 1986, S. 45–76.

[Rumelhart1986b]
David E. Rumelhart, Geoffrey E. Hinton, Ronald J. Williams: “Learning Internal Representations by Error Propagation”. In: David E. Rumelhart, James L. McClelland (Hg.): *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations*. Cambridge, Mass.: MIT Press, 1986, S. 318–362.

[Sacerdoti1977a]
Earl D. Sacerdoti: *A Structure for Plans and Behaviour*. Amsterdam: North-Holland, 1977.

[Schank1975a]
Roger C. Schank (Hg.): *Conceptual Information Processing*. Amsterdam: North-Holland, 1975.

[Searle1969a]
J. R. Searle: *Speech Acts*. London-New York: Cambridge University Press, 1969.

[Shieber1983a]
S. M. Shieber, H. Uszkoreit, F. C. N. Pereira, J. J. Robinson: “The Formalism and Implementation of PATR-II”. In: B. Grosz, M. Stickel (Hg.): *Research on Interactive Acquisition and Use of Knowledge*. SRI Final Report 1894. Artificial Intelligence Center, SRI International, Menlo Park, Calif., 1983.

[Shieber1986a]
Stuart M. Shieber: “An Introduction to Unification-Based Approaches to Grammar”. CSLI Lecture Note Series. Center for Study of Language and Information, Stanford, Calif., 1986.

[Sidner1979a]
C. L. Sidner: “Towards a computational theory of anaphora comprehension in English discourse”. Ph.D. Dissertation. Massachusetts Institute of Technology, Cambridge, Mass., 1979.

[Stolcke1986a]
A. Stolcke: “Korrekturen und Wiederholungen im gesprochenen Deutsch”. Arbeit zum Hauptseminar *Syntax der gesprochenen Sprache*. Institut für Deutsche Philologie, Universität München, München, Sommersemester 1986.

[Taylor1984a]
T. J. Taylor: “Editing rules and understanding: the case against sentence-based syntax”. *Language and Communication* **4**, 1984, 105–127.

[Touretzky1985a]
David S. Touretzky, Geoffrey E. Hinton: “Symbols Among the Neurons: Details of a

Connectionist Inference Architecture”. In: *Proceedings of the 9th International Joint Conference on Artificial Intelligence*. Los Angeles, Calif., August 1985, S. 238–243.

[Touretzky1986a]
David S. Touretzky: “BoltzCONS: Reconciling Connectionism with the Recursive Nature of Stacks and Trees”. In: *Proceedings of the 8th Annual Conference of the Cognitive Science Society*. Amherst, Mass., August 1986, S. 522–530.

[Weizenbaum1966a]
Joseph Weizenbaum: “ELIZA”. *Communications of the ACM* **9**, 1966, 36–45.

[Wilensky1983a]
Robert Wilensky: *Planning and Understanding*. Cambridge, Mass.: Addison-Wesley, 1983.

[Winograd1972a]
Terry Winograd: *Understanding Natural Language*. New York: Academic Press, 1972.

[Woods1970a]
W. A. Woods: “Transition network grammars for natural language analysis”. *Communications of the ACM* **13**(10), 1970, 591–606.

Inhalt