Über Koordinaten Grammatiken zur Bildverarbeitung und Szenenanalyse

Der Technischen Fakultät der
Universität Erlangen-Nürnberg

zur Erlangung des Grades

DOKTOR-INGENIEUR

vorgelegt von

Eckart Michaelsen

Erlangen - 1998

Als Dissertation genehmigt von
der Technischen Fakultät der
Universität Erlangen-Nürnberg

Tag der Einreichung:	11. August 1997
Tag der Promotion:	28. Januar 1998
Dekan:	Prof. Dr.-Ing. G. Herold
Berichterstatter:	Prof. Dr.-Ing. H. Niemann
	Prof. Dr.-Ing. H. Kazmierczak

Vorwort

Die
vorliegende Arbeit entstand als Dissertation begleitend zu meiner Tätigkeit als wissenschaftlicher Mitarbeiter am **F**orschungsinstitut für **I**nformationsverarbeitung und **M**ustererkennung (FIM) in Ettlingen. Für die Ermöglichung der Arbeit und die Erlaubnis zur Verwendung des institutseigenen Maschinenparks bedanke ich mich besonders bei Herrn Prof. Dr. H. Kazmierczak und Herrn K. Lütjen. Herrn Prof. Dr. H. Kazmierczak danke ich darüber hinaus für die Übernahme des zweiten Gutachtens. Herrn Prof. Dr. H. Niemann danke ich für die detaillierte und konstruktive Kritik, für die Übernahme der Betreuung als 'Externer' an der Univ. Erlangen-Nürnberg und für das erste Gutachten.

Die Arbeit am FIM ist wesentlich Teamarbeit, und so wäre die vorliegende Dissertation kaum möglich gewesen ohne die laufende Zuarbeit, Kritik und Diskussion der Kollegen in der Gruppe. Besonders bedanke ich mich diesbezüglich wiederum bei Herrn K. Lütjen, Herrn Dr. U. Stilla, Herrn Dr. R. Geibel, Herrn K. Jurkiewicz und Herrn H. Füger aber auch bei allen anderen Mitarbeitern, die mir mit Kritik und Ratschlägen zur Seite gestanden sind.

Ettlingen, November 30, 2005 Eckart Michaelsen

Contents

List of Figures

List of Tables

Chapter 1

Einleitung

Automatisches Erkennen von 3D Objekten aus 2D Bilddaten gilt als schwieriges Problem (siehe z. B. [ULLM, CHEL]). Im Vergleich mit den perzeptiven Leistungen des Menschen sind die Leistungen beim Maschinensehen noch recht bescheiden. Gegenwärtig können nur für bestimmte begrenzte Aufgabenstellungen Teillösungen vorgeschlagen und implementiert werden. Einleitend wird daher zunächst in diesem Kapitel die Aufgabenstellung für die vorliegende Arbeit eingeschränkt, der veröffentlichte Stand der Forschung im entsprechenden Umfeld umrissen und vor diesem Hintergrund der eigene Beitrag dargestellt und abgegrenzt.

- **Aufgabenstellung.** Zunächst kann man unterscheiden in *kooperative* und *nicht kooperative* Aufnahmesituationen. In der industriellen Bildverarbeitung herrschen in der Regel kooperative Bedingungen. Man hat also z. B. die Möglichkeit, die Beleuchtung zu manipulieren, an den Objekten spezielle Markierungen anzubringen, die Objekte zu vereinzeln, um für Trennbarkeit vom Hintergrund Sorge zu tragen, und die Kameraparameter frei zu wählen.

  Von **nicht kooperativen Bedingungen** soll die Rede sein, wenn die Beleuchtungsparameter unbekannt sind, wenn unvorhersehbare Hintergrundobjekte und partielle Verdeckungen eine verläßliche Segmentation der Bilddaten verhindern und wenn die Kamerastandorte nicht frei gewählt werden können. Im sichtbaren Spektralbereich ist das in der Regel der Fall, wenn nicht unter kontrollierbaren Laborbedingungen sondern im Freien robuste Ergebnisse erzielt werden sollen. Bei hochaufgelösten Luftbildern kommt hinzu, daß Datenmengen anfallen, die das übliche Videostandbildformat bei weitem überschreiten. Man muß dann auch untersuchen, in welchem Zusammenhang der Rechenaufwand mit dem Umfang der Eingangsdaten steht.

Während für Nadiraufnahmen aus größeren Höhen eine Vernachlässigung der Szenentiefe oder eine $2\frac{1}{2}$D Modellierung als Näherung noch vernünftig erscheint, wird bei Schrägsichten oder aus niedriger Höhe die Berücksichtigung der echten 3D Effekte (variierende Größe des Erscheinungsbildes, verschiedene Ansichten, Perspektive, partielle Verdeckung, unvorhersehbare Hintergrundobjekte usw.) unvermeidbar. Mit modernen Mehrfachbildkameras wird der Bereich bis zum Horizont abgedeckt. Vorstellbar sind zukünftig **autonome Trägerplattformen**, die sehr tief an den interessierenden Objekten vorbeifliegen. Entsprechende Meßkampagnien sind sehr aufwendig. Bodenbasierte Aufnahmen sind aber für diese Aufnahmesituation eine gute Näherung. In der vorliegenden Arbeit wird die Detektion und Lokalisation von bestimmten Fahrzeugen unter diesen Bedingungen bearbeitet.

- **Stand der Technik und Forschung.** Automatisches Erkennen von 3D Objekten aus 2D Bilddaten ist für die Bereiche Robotersehen in der Industrie, Fernerkundung, autonome Fahrzeugnavigation und einige andere Anwendungen ein wichtiges Arbeitsgebiet. Beispielhaft werden im folgenden ein paar Arbeitsgruppen, die auf den besagten Gebieten tätig sind oder waren, aufgeführt:

  International haben sich mit der Thematik 3D Objekterkennung aus Stereobildpaaren im Hinblick auf **industrielle Anwendungen** eine Reihe von Autoren beschäftigt: Z. B. wurde in Groß Britannien das Programm IKBS[1]-VISION aufgelegt, das durch das SERC[2] finanziert wurde. Beteiligt waren die Univ. Sheffield, die Univ. Edinburgh, die Univ. Aberdeen, die Univ. of Sussex, IBM UK, GEC Research Limited usw. Publiziert sind eine Reihe von Arbeiten von Mayhew, Frisby, Pollard, Fisher und anderen. Ein Überblick findet sich in [MAYH].

  Ein anderes Beispiel sind eine ganze Reihe von Versuchen mit Stereoaufbauten mit zwei und drei Kameras und mit Laser Entfernungsmessung, die in Frankreich am INRIA[3] betrieben werden. Darunter fallen auch Versuche mit frei beweglichen autonomen Innenraumfahrzeugen [FAUG-93-1]. Einen Abriß über Aktivitäten und Veröffentlichungen, z. B. von Faugeras, Deriche, Ayache, Tombre und anderen an diesem Institut im Bereich Computersehen, findet sich in [FAUG-93-2].

  Im Bereich **Fernerkundung** hat das Programm RADIUS[4] in den USA und international wesentliche Akzente gesetzt. Es ist gemeinsam finanziert

[1] Industrial Knowledge Based Systems
[2] Science & Engineering Research Council
[3] Institut National de Recherche en Informatique et Automatique
[4] Research and Development of Image Understanding Systems

durch die ARPA[5] und die CIA[6]. 3D Modelle werden sowohl zur Verarbeitung von Einzelbildern als auch von Stereobildpaaren oder Bildfolgen verwendet. Die Problemlösungen werden mit Aufnahmen von Modellen und mit Luftbildern demonstriert. Im Gegensatz zu den Arbeiten der entsprechenden europäischen Gruppen werden auch Schrägsichten bearbeitet, die aber aus relativ großen Flughöhen und unter moderaten Neigungswinkeln aufgenommen sind. Führende Arbeiten sind veröffentlicht von Strat, Nevatia, Riseman, Mc. Keown und Rosenfeld. Einen Überblick findet man in [IU-94, GRUE]. Sehr viele Mitarbeiter aus einer ganzen Reihe von bekannten Universitäten, Forschungseinrichtungen und Firmen sind beteiligt (z. B. Carnegie-Mellon, Univ. Mass., Univ. of Maryland, Univ. of S. Cal., SRI International, Martin Marietta). Das auf RADIUS folgende Nachfolgeprojekt APGD[7] betont den Aspekt der praktischen Machbarkeit [STRA].

In den USA wird eine teilweise ähnliche Aufgabenstellung auch unter dem Kürzel IUE[8] bearbeitet. Hier findet sich z. B. die Stanford Univ. mit dem Projekt SISTO[9] [BINF]. Dieses Projekt bearbeitet Detektions- und Lokalisationsaufgaben, die der Aufgabenstellung der vorliegenden Arbeit sehr nahestehen.

In Deutschland wurden automatische und semiautomatische Verfahren des 3D modellgesteuerten Computersehens von der DFG[10] gefördert für Gebäudeextraktion aus Luftbilddaten zur 3D GIS[11] Gewinnung. Beispiele für Arbeitsgruppen hierzu sind Förstner (Univ. Bonn) auf dem Gebiet Gebäudeextraktion aus Luftbildern mit statistischen Verfahren [BRUN], Fritsch (Univ. Stuttgart) auf dem Gebiet der Integration unterschiedlicher Wissensquellen hierzu [HAAL] und Stilla (FIM[12] Ettlingen) mit Produktionsnetzen [STIL-97]. Für einen Überblick empfiehlt sich auch hier [GRUE].

In der Schweiz gibt es diesbezüglich das Projekt AMOBE[13] unter der Leitung von Grün und Kübler (ETH Zürich) [HENR, GRUE].

Die vorgestellten Arbeitsgruppen bearbeiten hauptsächlich Nadirluftaufnahmen. Bodenbasierte Aufnahmen dienen beim Aufbau von 3D Stadtmodellen meist nur als Textur zur Verwendung in 3D Graphiksystemen (siehe

[5] Advanced Research Projects Agency
[6] Central Intelligence Agency
[7] Automatic Population of Geospatial Databases
[8] Image Understanding Environment
[9] Spatial Organization and Hypothesis Generation
[10] Deutsche Forschungs Gemeinschaft
[11] Geo Informations System
[12] Forschungsinstitut für Informationsverarbeitung und Mustererkennung
[13] Automation of Digital Terrain Model Generation and Man-Made Object Extraction from Aerial Images

etwa [GRUB]). Ihre mustererkennungsmäßige Verarbeitung wurde vor allen Dingen im Rahmen von Systemen zur **autonomen Fahrzeugführung** untersucht. Beispiele für Projekte dieser Art sind NAVLAP (Carnegie Mellon, Kanade) [THOR] und PROMETEUS unter anderem mit den Arbeitsgruppen Nagel (Univ. Karlsruhe) [NAGE] und Dickmanns (BW Univ. München) [DICK]. Dabei liegt die Betonung meist mehr auf der Modellierung der Fahrbahn. Fahrzeuge werden in der Regel nicht als 3D Polyhedron modelliert (eine Ausnahme ist [KLUG]). An bestimmten Orten im Bild kann aufgrund der Fahrbahngeometrie aus einem Segment bestimmter Größe auf ein Fahrzeug geschlossen werden. Sehr wichtig ist bei all diesen Ansätzen die Echtzeitfähigkeit, weil meist die Einbeziehung in ein regeltechnisches System zur Fahrzeugsteuerung demonstriert werden soll [WETZ].

Nicht nur Fahrzeugführung wurde untersucht, sondern auch Verkehrsüberwachung. Bei dieser Aufgabenstellung werden mit ruhender Kamera Bildfolgen aus Schrägsicht aufgenommen. Aus der Arbeitsgruppe Nagel (Univ. Karlsruhe) findet sich hier mit [KOLL] ein Verfahren, das Aspekte von Polyhedronmodellen von Fahrzeugen direkt mit den Grauwertgradienten vergleicht.

In der Arbeitsgruppe Niemann (Univ. Erlangen) wird auch 3D modellbasiertes Verfolgen in Bildfolgen mit bewegter Kamera bearbeitet [DENZ]. Darüberhinaus gibt es von dort ein statistisches Verfahren zur Erkennung von 3D Polyhedrons auf Einzelbildern [HORN].

- **Modellbasierte Ansätze zur Erkennung von 3D Objekten aus 2D Bilddaten.** Der vorgestellte Lösungsansatz ist modellbasiert. Daher folgt ein kurzer Überblick über die hierfür vorgeschlagenen Strukturen und Methoden. Verfahren der Mustererkennung können unterteilt werden in *statistische* und *nicht statistische* Ansätze [NIEM-83].

  In letzter Zeit gewinnt die **statistische Sichtweise** für die vorgestellte Aufgabe an Bedeutung. Einige Beispiele für solche Ansätze werden im folgenden gegeben: Hornegger faßt die extrahierten Merkmale als geometrische Zufallsvariablen auf [HORN]. So können Ausfälle oder Einsetzungen von Segmenten - etwa durch Beleuchtungseinflüsse - kompensiert werden. Verdeckungen werden in das statistische Modell eingebaut. Auch das Nichtwissen bzgl. der Tiefe, das durch die Projektion entsteht, wird modelliert. Zum Einsatz kommt insbesondere der EM[14] Algorithmus. Damit werden alle Werkzeuge bereitgestellt, um den Bayes Klassifikator auch für die vorgestellte Aufgabe anwendbar zu machen. Das bedeutet, daß auch die Modelle gelernt werden können anhand von Beispieldaten. Sie bestehen aus parametrisierten Dichtefunktionen.

[14] Expectation Maximization

Andere Beispiele sind aus dem Bereich Fernerkundung (Arbeitsgruppe Förstner, Univ. Bonn) bekannt. Auch hier werden parametrische Dichten verwendet. Verfolgt wird ein geschichteter Ansatz, um Beobachtungen mit Modellwissen zu verrechnen [BRUN, SEST]. Die Arbeitsgruppe von der Univ. Stanford verwendet Bayes Netze [BINF]. Diese werden automatisch aus einem VSCP[15] Modell, das der Benutzer zur Verfügung stellt, erzeugt. Die Betonung liegt hier auf einer gezielten Steuerung schon der Vorverarbeitung und Segmentierung durch das Modell.

Viele Autoren schlagen wegen des hohen Aufwandes bei der Modellierung und bei der Erfassung der statistischen Kenngrößen vor, für die vorgestellte Aufgabe **nicht statistische** Ansätze zu verwenden, und nehmen dabei einen gewissen Vericht auf Quantifizierbarkeit in Kauf. Das grundlegende Zuordnungsproblem zwischen gemessenen Bildmerkmalen und Objektmerkmalen aus dem Modellwissen tritt so deutlicher hervor [GRIM, ULLM, HARA, CHEN, FAUG-93-1]. Betonung liegt auf der Fragestellung als Problem der kombinatorischen Optimierung. Dabei gelten die geometrischen Gegebenheiten (Projektion, Perspektive, Verdeckung usw.) als Randbedingungen, die ausgenutzt werden können, um die Suche zu beschneiden und so durchführbar zu halten.

Die in der Mustererkennung übliche Unterscheidung in Ansätze mit globalen Merkmalen und Ansätze mit lokalen Merkmalen kann hier nur schwer durchgeführt werden. In den Verfahren werden meist globale Merkmale und Hinweise zur Aufstellung von Hypothesen über die Klasse, Lage und Orientierung genutzt, die dann mit lokalen Merkmalen durch Zuordnung und Fehlerquadratminimierung verifiziert werden. Man kann jedoch die Ansätze untergliedern in solche, die eine hinreichend große Anzahl von unterschiedlichen *Aspekten* zum Vergleich verwenden, und solche, die den Vergleich direkt mit dem *3D Modell* anstellen.

Aspektbasierte Verfahren können in der Fernerkundung [KORT] oder für den Bereich Robotersehen [BUN-89] verwendet werden. Diesen Verfahren ist gemein, daß ein Katalog von Aspekten des Objekts geführt wird, der im Verfahrensablauf zu Fallunterscheidungen führt. Die Aspekte eines Polyhedron Modells werden als Aspektgraph dargestellt. Jeder Aspekt ist ein Knoten darin. An den Kanten stehen die sogenannten visuellen Ereignisse, das sind Spezialperspektiven, bei denen z. B. eine Fläche als Linie abgebildet wird. Aspektgraphen können heute automatisch erzeugt werden [FISC]. Normalerweise wird mit extrahierten Liniensegmenten verglichen. Eine Ausnahme diesbezüglich ist der Ansatz von Kollnig [KOLL] mit Grauwertgradienten.

Bei Ansätzen mit einem direkten **3D Modellvergleich** werden

[15] Volume Surface Curve Point

hauptsächlich Tiefenbilder verwendet. In [FAUG-93-1] werden z. B. eine Reihe von Möglichkeiten vorgeschlagen, solche Tiefenbilder zu gewinnen. Bevorzugt werden drei Kameras und ein Laserentfernungsmesser verwendet. Andere Gruppen verwenden Streifenlichtprojektoren zur Tiefenbildgewinnung, z. B. [KREB]. Beispiele für Arbeiten, die die Entfernung der einzelnen Merkmale (Punkte oder Linien) von den Aufnahmestandorten mit Hilfe von Stereoberechnungen ermitteln, ohne ein komplettes Tiefenbild zu erzeugen, sind [MAYH, HAAL, STIL-97, MICH-96].

- **Syntaktische Verfahren in der Mustererkennung.**

  Da die vorliegende Arbeit Begriffe aus der syntaktischen Mustererkennung verwendet, wird der Stand der Dinge auf diesem Felde kurz umrissen. Seit den 60er Jahren wurden Ansätze entwickelt, die Beschreibung und Verarbeitung von Bildern mit formalen Grammatiken und Parsern durchzuführen. Dabei standen zunächst vergleichsweise einfache Strukturen (vereinzelte Chromosomen, Spuren auf Blasenkammerbildern usw.) im Mittelpunkt des Interesses [ROSE-79, FU-84, SHAW-69, GONZ]. Heute beschäftigen sich die Autoren auf diesem Felde (z. B. Wang, Bunke, Sanfeliu und Tombre) überwiegend mit komplexen Anwendungen aus dem Bereich der Dokumentenanalyse (Maschinenbauzeichnungen, Schaltpläne, Noten usw.) [BAIR, TOMB]. Alle zwei Jahre trifft sich die Arbeitsgruppe SSPR[16] der IAPR[17]. In den entsprechenden Tagungsbänden gewinnt man einen Überblick, z. B. für 1996 in [PERN].

  Für
  Koordinaten Grammatiken sind nur theoretische Analysen veröffentlicht (in letzter Zeit [ROSE-89-1, NAKA-89, NAKA-95]). Einzige Ausnahmen sind eine kurz gefaßte Vorabveröffentlichung von einigen Teilen aus der vorliegenden Arbeit in [MICH-96] und einige Bemerkungen in dieser Richtung in [STIL-97]. Im Zusammenhang mit graphischen Eingabeschnittstellen wurde in Gestalt der *constraint multiset grammars* eine verwandte Struktur entwickelt [MARI]. Der Ansatz beruht auf logischen Grammatiken also auf speziellen PROLOG Programmen.

- **Beitrag der vorliegenden Arbeit.** Die seit etwa Mitte der achtziger Jahre im FIM in Ettlingen verfolgten praktischen Ansätze zur Bildverarbeitung mit Produktionsnetzen, wie sie etwa in [LUE-86-1, LUE-86-2, FUE-90-2, STIL-91, STIL-95-2] beschrieben sind, werden theoretisch vertieft durch *Koordinaten Grammatiken*. Die Definition folgt dabei zunächst [ANDS-68-2, MILG]. Sie wird in passender Form präzisiert und erweitert bzw. modifiziert. Unterschiede zu anderen syntaktischen Strukturen, wie z. B. der *attributierten Grammatik* nach [TSAI, KNUT] oder den *Feld-Grammatiken* nach [KIRS, CLOW] werden verdeutlicht.

  Die Tragfähigkeit dieses Ansatzes für die vorgestellte 3D modellbasierte Detektions- und Lokalisationsaufgabe unter nicht kooperativen Bedingungen wird anhand einer Beispielgrammatik praktisch demonstriert. Dabei werden auch die Grenzen, Schwierigkeiten und Beschränkungen erläutert.

  Neu ist hier also einerseits die Verwendung eines syntaktischen Ansatzes für die modellbasierte Erkennung von 3D Objekten und andererseits die

[16] Syntactic and Structural Pattern Recognition
[17] International Association for Pattern Recognition

Demonstration an schwierigen bodenbasierten, hochaufgelösten Außenaufnahmen aus dem sichtbaren Spektralbereich, die einen Vorbeiflug in extrem niedriger Höhe simulieren. Dabei wird ein strikter 'bottom up' Ansatz verfolgt.

Durch die theoretische Analyse können neue Erkenntnisse gesammelt werden in Bezug auf die Systematik, Semantik und Rechenkomplexität der Produktionsnetze. Z. B. wird demonstriert, daß die verwendete anhäufende Ablaufmethode nicht äquivalent ist mit dem Ersetzungssystem, das die Semantik eines Produktionsnetzes beschreibt. Es wird deutlich, warum man trotz dieses theoretischen Resultats an der anhäufenden Ablaufmethode festhalten sollte.

Ein anderes wichtiges Resultat ist die hohe Rechenkomplexität im 'worst case' bei kontrollierbarem Aufwand in der Praxis. Es ergeben sich hier wichtige neue Anhaltspunkte und Faustregeln für die praktische Arbeit mit Produktionsnetzen. Neue Perspektiven für die Abschätzung von zu erwartenden erforderlichen Rechenleistungen für ein gegebenes Produktionsnetz aus einer Statistik der Eingangsdaten heraus werden eröffnet. Das ermöglicht eine Reihe von neuen Ansatzmöglichkeiten für die zukünftige Bewertung und Entwicklungsstrategie auf dem Gebiet der Produktionsnetze.

- **Abgrenzung der vorliegenden Arbeit.** In dieser Arbeit wird eine präzise Erkennung von Objekten anhand von sehr spezifischen Modellen angestrebt. Dies unterscheidet sie von vielen Arbeiten aus dem Gebiet der autonomen Fahrzeugführung, wie etwa [WETZ]. Dazu braucht man höher aufgelöste Daten. Man kann auch nicht von Hinweisen - wie etwa Straßen - ausgehen, die den Bereich und in Abhängikeit davon Größe und Erscheinungsbild einschränken. Es gibt aber dennoch Gemeinsamkeiten mit den Arbeiten aus der Fahrzeugführung. So ist im Gegensatz zu Bildfolgen von einer ruhenden Überwachungskamera aus, die bewegte Objekte aufnimmt (wie etwa in [KOLL]), die Kamera bewegt und das Objekt ruhend zu denken.

  Es wird davon ausgegangen, daß das Objekt nicht auffällig ist und mögliche globale Merkmale kaum beeinflußt. Statt dessen wird mit [MARR] ein reiner 'bottom up' Ansatz verfolgt. Damit steht diese Arbeit im Gegensatz zu Standardansätzen zur modellbasierten 3D Objekterkennung, wie etwa [ULLM, BINF, HAAL].

  Das vorgestellte Verfahren verwendet kein automatisiertes Lernen oder adaptieren. Es steht damit im Gegensatz zu Verfahren, die das verwendete Modell anhand von Beispieldaten erlernen wie etwa in [HORN]. Es werden vielmehr sowohl das Modell (als 3D Polyhedron mit Teil-von Struktur) als auch Teile der Vorgehensweise beim Erkennen vom Anwender vorgegeben. Es wird davon ausgegangen, daß hierfür genügend Zeit und Personal zur

Verfügung stehen. Ebenfalls im Gegensatz zu [HORN] liegt ein nicht statistischer Ansatz vor.

- **Gliederung der vorliegenden Arbeit.** Kap. 2 definiert die grundlegenden Begriffe. Die im wesentlichen auf Rosenfeld zurückgehenden Hauptresultate der Theorie werden vorgestellt. Mit den Parsern, die zu solchen Strukturen gehören, beschäftigt sich das Kapitel 3. Das Kap. 4 stellt dann einige Möglichkeiten zur Implementierung entsprechender Verfahren vor. Im Kapitel 6 wird an einem Transporter (VW-Bus Doppelkabiner) in natürlichem Gelände demonstriert, wie solche Verfahren, z. B. zur modellgesteuerten Detektion von bestimmten Fahrzeugen in einer 3D Szene, eingesetzt werden können. Das beinhaltet Themen der Bildverarbeitung und der Szenenanalyse. Unumgänglich ist bei Entwicklung und Einsatz solcher Verfahren eine Unterstützung des Wissenserwerbs und der Wissenserklärung. Dabei sollte auch eine statistische Untersuchung der zu erwartenden Eingangsdaten durchgeführt werden, um die voraussichtliche Rechenkomplexität abschätzen zu können. Das Kapitel 5 beschäftigt sich mit diesen für die Praxis sehr wichtigen Fragen. Im Kapitel 7 wird der Ansatz mit einigen anderen Veröffentlichungen vergleichend diskutiert und bewertet. Den Abschluß bildet mit Kapitel 8 eine kurze Zusammenfassung.

Chapter 2

Grundlagen und Begriffe

In [MARR] werden drei Ebenen eines Problems der Informatik unterschieden: 1. Die theoretischen Grundlagen, 2. die Repräsentation der Daten und die verwendeten Algorithmen und 3. die Implementierung in Hardware.

Auf der ersten Ebene verwendet man heute üblicherweise Begriffe aus der Automatentheorie, aus der Theorie der Berechenbarkeit und Entscheidbarkeit oder der Theorie der rekursiven Funktionen. Oft sind auch stochastische Modellierungen erforderlich. Man versucht eine gestellte Aufgabe solange zu präzisieren, bis sich herausstellt, ob sie äquivalent ist mit einer der bekannten klassischen Problemklassen. Dann kann beantwortet werden, ob es einen Algorithmus geben kann, der die Aufgabe löst. Auf dieser Ebene erscheinen viele Aufgabenstellungen aus dem Gebiet des Maschinensehens schlecht gestellt. Man kann nicht aus einer zweidimensionalen Strahlungsdichtenverteilung, die in einem Projektionszentrum ankommt, auf die Lage und Art der Objekte schließen, die diese Strahlung emittiert oder reflektiert haben. Das Urbild einer oder mehrerer Projektionen ist eine sehr große Menge von möglichen Szenen. Diese Mehrdeutigkeit kann reduziert werden, wenn Wissen über mögliche oder doch wahrscheinliche Szenen hinzugenommen wird. Dieser Ansatz zur theoretischen Fundierung des Maschinensehens (und der Wahrnehmungspsychologie) gewinnt immer mehr an Bedeutung. Man findet z. B. in [BENN] die maß- und wahrscheinlichkeitstheoretische Modellierung eines 'observers' in seiner 'Welt'. Danach kann derselbe Schluß von denselben Daten auf dieselbe Szene, der in der einen 'Welt' fast sicher falsch ist, in der anderen fast sicher korrekt sein, wenn nur das Maß auf den vorkommenden und zu unterscheidenden Szenen anders definiert ist.

Leider sind diese Maße in vielen praktischen Anwendungen schwer zu bestimmen. Die Frage, wieviel und welches Wissen über die Szene von Fall zu Fall notwendig ist, um den Schluß von den beobachteten Daten auf die zu unterscheidenden Szenen hinreichend sicher zu machen, kann oft nur heuristisch angegangen werden. Man probiert ein Verfahren an hinreichend repräsentativen Daten aus, und,

wenn es zuviele Fehler produziert, fügt man zusätzliches Wissen über die Szene hinzu. Wenn also eine Theorie des Maschinensehens von praktischem Wert sein soll, so muß sie es ermöglichen, unkompliziert Erweiterungen und Modifikationen an den durch sie zu präzisierenden Verfahren vorzunehmen.

Andererseits erscheint ein 'blindes' Probieren mit Evolutionsstrategien oder einfachen Lernverfahren unangebracht, weil man durch Introspektion der eigenen Wahrnehmung oder andere Quellen einige Anschauung insbesondere über die geometrische, topologische und strukturierte Natur der Objekte in der Szene und ihre Abbildung in den Daten gewinnen kann. Syntaktische Strukturen bieten sich bei der Umsetzung solch anschaulichen Wissens in zu testende Verfahren an, weil sie der menschlichen Formulierung von Wissen durch Sprache nachgebildet sind. Man kann damit mit kleinem, endlichem Vokabular und wenigen Bildungsregeln sehr große, und sogar auch unendliche Mengen von Szenen unterscheiden. Das Maschinensehen verlangt aber nach etwas anderen syntaktischen Strukturen als die Sprachanalyse.

Dieses Kapitel beschäftigt sich mit der nach Marr ersten Ebene, also der theoretischen. Es wird einen formaler Sprachapparat konstruiert, der es erlaubt, gewisse, auf dem Gebiet der Szenenanalyse und des Maschinensehens wichtige, logische und geometrische Constraints so zu präzisieren, daß sie sowohl implementierbar werden als auch erklärbar für den Benutzer bleiben. Abschnitt 2.1 liefert die formale Definition der Koordinaten Grammatik. Der resultierende Reduktionsbegriff unterscheidet sich von dem normaler, generischer Grammatiken erheblich. Gegenüber den in der Literatur über Koordinaten Grammatiken verwendeten Definitionen erscheint hier noch eine gewisse Präzisierung angebracht. Dies geschieht in Abschnitt 2.2. Damit sind die Grundlagen gelegt, um in Abschnitt 2.3 die Hauptresultate der Theorie der Koordinaten Grammatiken vorzustellen.

2.1 Koordinaten Grammatiken

D. L. Milgram and A. Rosenfeld definieren in [MILG] den Begriff der *graphical rewriting grammar* (GRG), um Verfahren der strukurorientierten, symbolverarbeitenden Mustererkennung auf eine logische Grundlage zu stellen, bzw. die automatentheoretische Mächtigkeit solcher Ansätze zu klären. Sie orientieren sich dabei an einem entsprechenden Verfahren von Anderson [ANDS-68-1, ANDS-68-2]. Bei Rosenfeld heißt solch ein Produktionssystem (etwa in [ROSE-89-1]) auch *coordinate grammar*. Ein ähnlicher Formalismus wird nun hier als *Koordinaten Grammatik* (kurz KG) eingeführt. Er arbeitet auf einer Menge von Symbolen, denen die Koordinaten eines Attributvektors[1] zugeordnet

[1] Es ist in der KI allgemein üblich, meist numerische Werte, die den Instanzen eines Symbols zugewiesen werden sollen, als 'Attribute' zu bezeichnen. Der an sich also naheliegende Name

werden. Analog zu den normalen, generischen Grammatiken gibt es Produktionen. Sie werden aber andersherum geschrieben, da man sich mehr für die reduzierende Richtung interessiert. Im übrigen wird nicht auf Tupeln von Symbolen gearbeitet, sondern auf *Mengen von Instanzen*. Der Unterschied liegt darin, daß nicht die Konkatenation der Symbole in einem Wort das entscheidende Kriterium für die Anwendbarkeit einer Ersetzungsregel bildet, sondern vielmehr die Konfiguration der zugehörigen Koordinatenwerte im Attributraum.

2.1.1 Definition der Koordinaten Grammatik

Eine **Koordinaten Grammatik** kurz **KG** wird definiert als ein 6-Tupel $G = (T, N, D, n, P, g)$, wobei die einzelnen Symbole für folgende Strukturen stehen:

- T ist eine endliche Menge von Symbolen. Sie heißen **Terminale**.
- N ist eine endliche Menge von Symbolen, wobei $T \cap N = \emptyset$ gilt. Sie heißen daher **Nichtterminale**. $V \stackrel{def}{=} T \cup N$ heißt das **Alphabet** der KG.
- D ist ein n-Tupel $(D_1, ..., D_n)$ von Mengen. Man sagt dazu auch **Attributbereich**.
- n ist die Zahl der Komponenten [2] des Attributraumes also eine natürliche Zahl.
- P ist eine endliche Menge von **Produktionen** der Form $p = (\Lambda, \Sigma, \pi, \phi)$, wobei dahinter wiederum folgende Struktur steht:
  - Λ, die **linke Seite**, ist ein j-Tupel von Symbolen für eine natürliche Zahl
    $j \geq 1$. Mindestens eines der Symbole muß ein Nichtterminal sein.
  - Σ, die **rechte Seite**, ist ein k-Tupel von Symbolen für eine natürliche Zahl $k \geq 1$.
  - π ist ein k-stelliges Prädikat. Der Definitionsbereich von π ist D^k.
  - ϕ ist ein j-Tupel von k-stelligen Funktionen

$$\phi : D^k \longrightarrow D^j.$$

'attributierte Grammatik' ist aber anderweitig belegt (siehe Abschnitt 7.3.2 oder [KNUT]), und kann daher hier nicht Verwendung finden.

[2] der Begriff 'Dimension' wird hier vermieden, da er später im Abschnitt 5.1.3 noch anderweitig belegt wird.

- $g \in N$ ist ein einzelnes, spezielles nichtterminales Symbol, das **Startsymbol**.

In der Bildverarbeitung und Szenenanalyse hat der Attributbereich z. B. Komponenten der folgenden Gestalt:

- $D_i = \{a \in Z^{d_i}; min_{i,1} \leq a_1 \leq max_{i,1},\ ...,\ min_{i,d_i} \leq a_{d_i} \leq max_{i,d_i}\}$ also begrenzte d_i-dimensionale Intervalle von ganzen Zahlen, insbesondere eindimensionale Intervalle, Pixelkoordinaten in Bildern, Bildfolgen mit zusätzlicher Koordinate t, Voxelräume usw.;
- $D_i = \{a \in Z\ mod\ (max_{i,1} + 1) \times ... \times Z\ mod\ (max_{i,d_i} + 1)\}$ also Intervalle aus Z^{d_i} mit in sich geschlossener, torischer Topologie, insbesondere Orientierungen in Grad oder Millirad;
- $D_i = \{(a \in Z^{d_i}; a_1^2 + ... + {a_{d_i}}^2 \simeq L\}$ mit einer natürlichen Zahl L, also eine ganzzahlige Annäherung an Mannigfaltigkeiten der Dimension $d_i - 1$ mit der Topologie einer Kugel zur Darstellung etwa von Richtungs- und Normalvektoren im Szenenraum.

Auch andere Topologien und Geometrien, wie etwa die projektive Ebene, sind sinnvoll, wenn dies die Anwendung erfordert. Symbolische Attribute wie 'Farbe' mit Werten wie 'blau' sind ebenfalls möglich. Man vermeidet aus technischen nicht aus theoretischen Gründen Gleitkommadarstellungen der Koordinaten. Alle Attribute D_i sind in der Praxis endlich. Deswegen heißt eine solche Grammatik auch **endlich attributiert**. Für theoretische Untersuchungen nimmt man dagegen

- $D_i = Z^{d_i}$ also das unbegrenzte, d_i-dimensionale, ganzzahlige Gitter.

Eine solche Grammatik heißt dann **nicht endlich attributiert**.

2.1.2 Konfiguration und Produktion

- Unter einer **Instanz**[3] I wird ein Paar verstanden, bestehend aus einem Symbol aus V zusammen mit den Koordinaten eines Attributvektors $a =$

[3]Der Begriff *Instanz* wird in der KI häufig im Zusammenhang mit dem Begriff *Konzept* verwendet (vergl. Abschnitt 7.4 oder etwa [NIEM-90]). Ein Konzept zu Objekten in Bildern beinhaltet Attribute wie z. B. Ort, Ausrichtung und Länge, mit denen die Objekte beschrieben werden, und Verfahren, mit denen die Werte dieser Attribute bestimmt werden. Ein aus einem konkreten Bild so erzeugtes Objekt heißt dann Instanz dieses Konzeptes. Es hat konkrete Attributwerte und wird im Rahmen des zugehörigen Konzeptes beschrieben. Insofern erscheint die Bezeichnung 'Instanz' für ein Symbol einer KG zusammen mit einem konkreten, aus den Eingangsdaten bestimmten Attributvektor gerechtfertigt. Den Rahmen definiert bei der KG der Attributbereich. Auf eine definitive Verwendung des Begriffs *Konzept* kann verzichtet werden.

$(a_1, ..., a_n) \in D$. Mit $s(I)$ wird der Symbolteil angesprochen und mit $a(I)$ der Attributvektor. Die Menge aller möglichen Instanzen, also

$$\mathcal{U} \stackrel{def}{=} V \times D \ ,$$

heißt das **Universum** zu einer solchen KG.

- Wenn $\Gamma \in V^m$ eine Symbolkette (ein Wort der Länge m) ist, dann ist ein m-Tupel κ von Instanzen aus $\mathcal{U}$ eine **Konfiguration** von Γ, genau dann, wenn $\Gamma_i = s(\kappa_i)$ für alle $i = 1...m$ gilt. Mit $a(\kappa)$ wird das m-Tupel der Attributvektoren bezeichnet.

- Wenn $p = (\Lambda, \Sigma, \pi, \phi) \in P$ eine Produktion in der KG ist, dann ist κ_a eine **p-Ausgangskonfiguration**, genau dann, wenn κ_a eine Konfiguration der rechten Seite Σ ist, und $a(\kappa_a)$ das Prädikat π erfüllt. Vermöge der Funktion ϕ ist zu solch einer p-Ausgangskonfiguration ein j-Tupel von Koordinaten im Attributraum bestimmt, welches zusammen mit der linken Seite Λ die zu κ_a gehörige **p-Zielkonfiguration** $\kappa_z = (\Lambda, \phi(a(\kappa_a)))$ ergibt. Die natürliche Richtung einer KG Produktion ist also nicht wie bei den Stringgrammatiken generisch (von links nach rechts) sondern reduzierend (von rechts nach links).

  Für die Menge der Ausgangskonfigurationen zu einer gegebenen Produktion p wird die folgende Schreibweise eingeführt:

$$\mathcal{A}_p \stackrel{def}{=} \{\kappa_a;\ s(\kappa_a) = \Sigma\ \wedge\ \pi(a(\kappa_a))\}$$

  Und für die Zielkonfigurationen setzt man entsprechend:

$$\mathcal{Z}_p \stackrel{def}{=} \{\kappa_z;\ s(\kappa_z) = \Lambda\ \wedge\ \exists \kappa_a \in \mathcal{A}_p :\ a(\kappa_z) = \phi(a(\kappa_a))\}$$

  In Anlehnung an die Konventionen in der mathematischen Symbolik zum Funktionsbegriff wird die Produktion dann wie folgt dargestellt:

$$\Sigma \stackrel{\pi}{\underset{\phi}{\longrightarrow}} \Lambda$$

  Das Überführen einer Ausgangskonfiguration in die Zielkonfiguration schreibt sich als:

$$\kappa_a \stackrel{\pi}{\underset{\phi}{\longmapsto}} \kappa_z$$

  Dabei ermöglicht es die Platzierung des Prädikates über dem Pfeil und der Funktion unter dem Pfeil, auch zusammengesetzte Prädikate und vor allen Dingen die immerhin n-dimensionalen Funktionen übersichtlich darzustellen, wie es das folgende Beispiel zeigt.

2.1.3 Ein Beispiel zur Veranschaulichung

Die folgende Produktion arbeitet auf einem quadratischen Bild mit hundertfünfzig Pixeln Kantenlänge. Definiert sind die Symbole *Winkel* und *Linie*. Als Attribute gibt es drei Endpunktkoordinaten im Bild. Es werden zwei *Linien* zu einem *Winkel* zusammengesetzt. Als Prädikat π dient die Nachbarschaft *adj* zweier Endpunkte, als Funktion ϕ im wesentlichen die Schnittpunktbildung int_{2d}

der zugehörigen Geraden. Die genaue Definition des Prädikats und der Funktion findet sich im Anhang B. Es ergibt sich die folgende Schreibweise.

$$(Linie, Linie) \xrightarrow[\begin{array}{l} P_1 = P_{2,2} \\ P_2 = int_{2d}(P_{1,1}, P_{1,2}, P_{2,1}, P_{2,2}) \\ P_3 = P_{1,2} \end{array}]{adj(P_{1,1}, P_{2,1})} (Winkel)$$

Die Wirkungsweise dieser Produktion kann man sich, wie in Abbildung 2.1 geschehen, an einem Beispiel von passenden Ausgangs- und Zielkonfigurationen klarmachen, indem man, der geometrischen Natur der Sache entsprechend, eine graphische Darstellung der Symbole vornimmt.

In diesem Falle findet die folgende Ersetzung statt:

$$\kappa_a = \begin{pmatrix} Linie & Linie \\ \begin{pmatrix} 100 \\ 115 \end{pmatrix} & \begin{pmatrix} 117 \\ 72 \end{pmatrix} \\ \begin{pmatrix} 20 \\ 115 \end{pmatrix} & \begin{pmatrix} 60 \\ 19 \end{pmatrix} \\ \begin{pmatrix} - \\ - \end{pmatrix} & \begin{pmatrix} - \\ - \end{pmatrix} \end{pmatrix} \longmapsto \kappa_z = \begin{pmatrix} Winkel \\ \begin{pmatrix} 60 \\ 19 \end{pmatrix} \\ \begin{pmatrix} 132 \\ 115 \end{pmatrix} \\ \begin{pmatrix} 20 \\ 115 \end{pmatrix} \end{pmatrix}$$

Es ist an dieser Stelle eine Warnung angebracht: Die Veranschaulichung einer Produktion wie in Abbildung 2.1 durch graphische Ausgabe beispielhafter Ausgangs- und Zielkonfigurationen birgt, wie jede Veranschaulichung, entscheidende Gefahren in sich. Die Gestaltwahrnehmung des Menschen sieht hier Dinge, die nicht gemeint sind, mit, ohne daß sich der Betrachter dessen bewußt wird. Insbesondere die Art, wie unsere Wahrnehmung generalisiert, gruppiert und Symmetrien erkennt und verwendet, ist dem Betrachter auch durch gezielte Introspektion nur schwer zugänglich. Daher läßt sich durch Aufzeichnen solcher Beispiele kaum eine Produktion definieren, die dann auch das tut, was der Anwender 'im Sinn' hatte. Das bleibt vielmehr ein iterativer Prozeß, auf den das Kapitel 5 detaillierter eingeht.

Insbesondere muß beim Erstellen einer Produktion auf **Wohldefiniertheit** geachtet werden. Es ist nachzuweisen, daß die Funktion ϕ für alle Attributwerte, die das Prädikat π zuläßt, definiert ist, und ihr Wert in die entsprechenden Attributintervalle fällt. In der Beispielproduktion ist beides verletzt. Erstens würde für Konfigurationen mit parallelen Linien die Schnittpunktbildung auf eine Division durch Null führen, und zweitens muß der Schnittpunkt, wenn er denn existiert, nicht unbedingt innerhalb der Bildgrenzen liegen. Man könnte in diesem

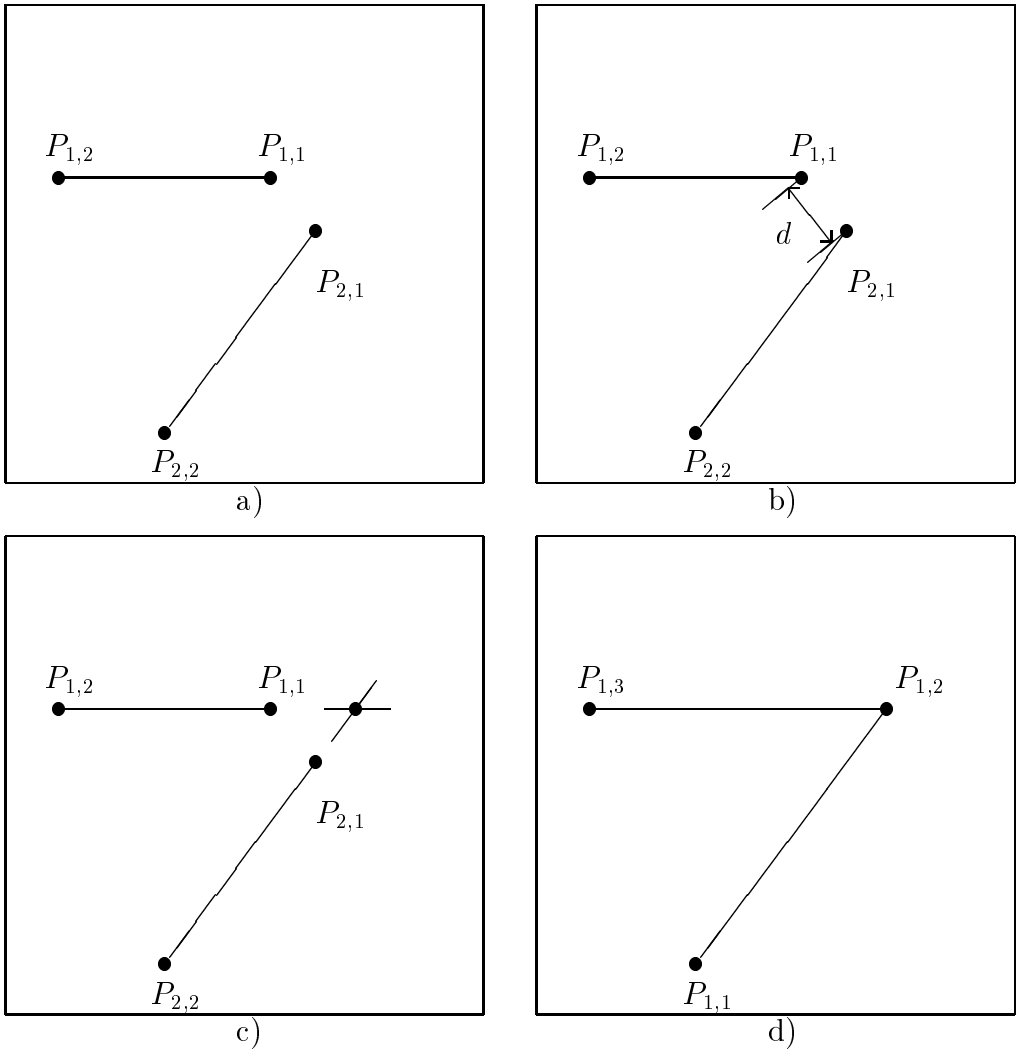


Figure 2.1: **Graphische Veranschaulichung einer Produktion**
a) Ausgangskonfiguration aus zwei Instanzen des Symbols *Linie*
b) Das Prädikat $adj(P_{1,1}, P_{2,1})$ ist erfüllt, denn der Abstand zwischen diesen Punkten d ist kleiner d_{max}
c) Die Funktion $int_{2d}(P_{1,1}, P_{1,2}, P_{2,1}, P_{2,2})$ berechnet den Schnittpunkt
d) Die neue Konfiguration besteht aus einer Instanz des Symbols *Winkel* mit den dargestellten Attributwerten

Beispiel auf die Idee kommen, die offenbar recht 'instabile' Schnittpunktbildung durch den Mittelwert der benachbarten Punkte $P_{1,1}$ und $P_{2,1}$ zu ersetzen. Mittelwerte sind immer wohldefiniert. Allerdings 'verbiegt' das den Winkel, den man eigentlich meint, auf unschöne Weise, und es werden Konfigurationen als Ausgangskonfiguration für einen Winkel zugelassen, die nicht wirklich gemeint waren (z. B. dieselbe Linie in beiden Rollen).

Offensichtlich ist es wichtig, für das Vorliegen einer Ausgangskonfiguration für einen 'Winkel', nicht nur die Nachbarschaft zweier Endpunktkoordinaten zu verlangen, sondern auch unterschiedliche Orientierung der beiden Linien. Natürlich ließe sich ein entsprechendes Prädikat auf den vier Bildkoordinatenattributen der beiden Linien definieren. Einfacher und für den Benutzer übersichtlicher ist es, die Orientierung direkt als Attribut einer 'Linie' zu definieren. Rosenfeld selbst schlägt z. B. in [ROSE-89-2] schon Koordinaten Grammatiken mit solchen Winkelattributen vor. Parallelität ist dann einfach Nachbarschaft in diesem Teilattribut, und die geforderte Antiparallelität ist ein Mindestabstand darin. Dabei ist die Topologie des Orientierungsattributes torisch. Das entsprechende Prädikat wird hier mit *apa* bezeichnet. Durch dieses zusätzliche Prädikat ist nun der Schnittpunkt sicher definiert. Zur Wohldefiniertheit muß er noch im Bild liegen, was ebenfalls konjunktiv zum Prädikat hinzugefügt werden kann:

$$(Linie, Linie) \xrightarrow[\substack{P_1 = P_{2,2} \\ P_2 = int_{2d}(P_{1,1}, P_{1,2}, P_{2,1}, P_{2,2}) \\ P_3 = P_{1,2}}]{\substack{adj(P_{1,1}, P_{2,1}) \\ \wedge apa(O_1, O_2) \\ \wedge int_{2d}(P_{1,1}, P_{1,2}, P_{2,1}, P_{2,2}) \in D_2}} (Winkel)$$

Auch das trifft noch nicht die gewünschte Semantik. Abbildung 2.2 zeigt eine Konfiguration, die wie eine korrekte Ausgangskonfiguration für diese Produktion aussieht. Es sind aber die falschen Punkte benachbart. Deswegen liegt hier keine Ausgangskonfiguration für diese Produktion vor. Auf solche Probleme, die ihre Ursache in der Wahl der Attribute haben, wird in Abschnitt 5.2.3 noch detailliert einzugehen sein.

2.2 Reduktion

Wie das Wort 'Grammatik' schon suggeriert, befindet man sich hier im Gebiet der syntaktischen Verfahren. Im Gegensatz zu eher statistischen - also entscheidungstheoretischen - Ansätzen, stehen nicht nur die numerischen Berechnungen

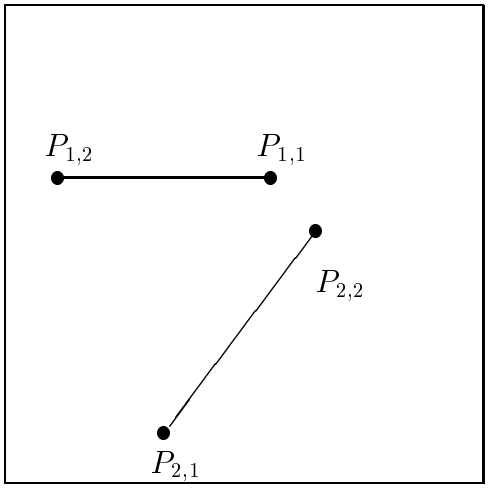


Figure 2.2: **Gegenbeispiel: Keine Ausgangskonfiguration**

im Vordergrund, sondern auch logische Strukturen, die durch Operationen mit Symbolen, Mengen von Symbolen, Ketten von Symbolen, Herleitungswege usw. zu beschreiben sind, damit sie nachvollziehbar und anschaulich bleiben. Abschnitt 2.2.1 zitiert den Ansatz zur Definition für einen Reduktionsbegriff für Koordinaten Grammatiken aus der Literatur. Abschnitt 2.2.2 präzisiert diesen Ansatz zu einer mathematisch korrekten Definition. Der praktische Gebrauch dieses Begriffs wird im Abschnitt 2.2.3 veranschaulicht.

2.2.1 Definition in der Literatur

Anderson, Milgram, Rosenfeld usw. verwenden folgenden Reduktionsbegriff (zitiert aus [MILG] Seite 189):

- "A set S_{i+1} of symbols and associated n-tuples of coordinates is said to directly reduce into another such set S_i if there exists a production $(\Lambda, \Sigma, \pi, \phi)$, for which Σ is a subset to S_{i+1}; its coordinates satisfy π; the coordinates of the symbols of Λ are obtained from those in Σ by applying the functions in ϕ; and $S_{i+1} - \Sigma \cup \Lambda = S_i$."

Dabei steht 'coordinates' für Attribute. Den Begriff 'Instanz' verwenden Milgram und Rosenfeld nicht. Statt dessen sind bei ihnen die Attributwerte zu den Symbolen 'assoziiert'. Einen Begriff für Konfigurationen gibt es nicht. Auch bei Milgram und Rosenfeld sind Λ und Σ natürlich *Tupel* von Symbolen. S_{i+1}

hingegen ist eine *Teilmenge* des Universums der Symbole mit ihren assoziierten Koordinaten. Insofern ist ein Ausdruck wie 'Σ is a subset to S_{i+1}' einfach nicht korrekt. Man kann also zwar diesen Text nicht als Definition im mathematischen Sinne verstehen, sollte aber trotzdem versuchen herauszubekommen, was eigentlich gemeint ist: Ein k-Tupel von Symbolen ordnet jedem Index zwischen 1 und k ein Symbol zu. Der Bildbereich dieser Zuordnung ist eine Menge von Symbolen, die natürlich Untermenge einer anderen Symbolmenge sein kann. Normalerweise meint man diesen Bildbereich, wenn man ein Tupel als Teilmenge auffassen will. Das läuft auf eine Vernachlässigung der Reihenfolge hinaus.

In diesem Falle sind allerdings Koordinaten assoziiert, die π erfüllen sollen. Die Symbole sind mit ihren Koordinaten zu betrachten, eben als Instanzen. Offenbar ist Milgrams und Rosenfelds Intention nicht die Konfiguration - also ein Tupel von Instanzen - sondern deren Bildbereich - eine Instanzenmenge zu betrachten. Sie folgen darin Anderson. Folgte man dieser Betrachtungsweise konsequent, so müßte man das Prädikat und die Funktion nicht auf den k-Tupeln aus D definieren, sondern auf der Menge der k-elementigen Untermengen von D. Das heißt der Wahrheitswert des Prädikates darf nicht von der Aufzählungsreihenfolge der Konfiguration abhängen. Bei vielen Prädikaten - etwa Nachbarschaft, Parallelität usw. - ist das der Fall. An dieser Stelle bringt es aber unnötige Einschränkungen in der Allgemeinheit,[4] so daß diese Arbeit dem hier nicht folgt, sondern im nächsten Abschnitt eine brauchbare und hinreichend exakte Definition auf der Basis des oben eingeführten Tupel-Konfigurationsbegriffs liefert, die ansonsten mit den Intentionen von Anderson, Milgram und Rosenfeld übereinstimmen dürfte.

2.2.2 Exakte Definition des Reduktionsbegriffs einer KG

Für den Bildbereich eines Tupels λ wird im folgenden $\mathcal{B}(\lambda)$ geschrieben. Sind dann S und $S' \subseteq \mathcal{U}$ Mengen von Instanzen, so definiert man:

$$S \longrightarrow S'$$

genau dann wenn

$$\exists p = (\Lambda, \Sigma, \pi, \phi) \in P \; \exists \kappa_a \in \mathcal{A}_p \text{ mit } p: \; \kappa_a \overset{\pi}{\underset{\phi}{\mapsto}} \kappa_z$$

$$\text{so daß } \mathcal{B}(\kappa_a) \subset S \text{ und } \; S' = S \backslash \mathcal{B}(\kappa_a) \; \cup \; \mathcal{B}(\kappa_z)$$

Man sagt dazu: 'S' **erzeugt direkt** S.' Oder es heißt: 'S **reduziert direkt** zu S'.' Hier wird die Intention explizit, daß eine solche Koordinaten Grammatik

[4] Die Abschnitte 5.2.3 und 5.2.5 gehen auf dieses Thema nocheinmal genauer ein

nicht auf Zeichen*ketten* arbeitet sondern auf *Mengen* von Instanzen. S' entsteht, indem man das Bild der Ausgangskonfiguration in S ersetzt durch das Bild der Zielkonfiguration. Will man hervorheben, durch welche Produktion die Reduktion geschieht, so kann man $S \xrightarrow{p} S'$ notieren. Im nächsten Schritt wird die transitive Hülle definiert durch

$$S_0 \xrightarrow{*} S_r$$

genau dann, wenn

$$\exists S_0, ..., S_r \subseteq \mathcal{U} \ : \ \ \forall i = 1...r : \ \ S_{i-1} \longrightarrow S_i$$

und sagt dazu 'S_r **erzeugt** S_0' oder auch 'S_0 **reduziert** zu S_r'. [5] Damit hat man einen syntaktischen Herleitungsbegriff.

Die Definition der Koordinaten Grammatik in Abschnitt 2.1.1 folgt insofern nicht ganz der in [MILG] angedeuteten Definition, als analog zu klassischen Zeichenkettengrammatiken verlangt wird, daß die linke Seite einer Produktion Λ mindestens ein Nichtterminal enthalten muß. Damit ist klar, daß aus einer Menge, die nur noch terminale Instanzen enthält, nichts weiter erzeugt werden kann. Definiert man die Menge aller terminalen Instanzen

$$\mathcal{T} \stackrel{def}{=} \{I \in \mathcal{U};\ s(I) \in T\}$$

so ergibt sich somit analog zu den Zeichenkettengrammatiken die zu einer solchen KG gehörige **Sprache** als:

$$\mathcal{L}_{pure} \stackrel{def}{=} \{S \subseteq \mathcal{T};\ \ \exists I \in \mathcal{U} \ : \ s(I) = g \ \wedge \ S \xrightarrow{*} \{I\} \ \}$$

Die Markierung *pure* dient dabei der Kennzeichnung als 'ungestörte' Sprache. In der Bildverarbeitung und Szenenanalyse auf Daten aus der natürlichen Umwelt mit syntaktischen Verfahren kann man nicht von einer Konfiguration ausgehen, die lediglich eine Instanz des Startsymbols enthält. Meist befinden sich außer dem gesuchten Objekt noch andere Dinge im Bild oder der Szene. Solche irrelevante Störinformation macht meist sogar den weit überwiegenden Teil der Daten aus. Sie sollten ignoriert werden und nicht etwa zur Rückweisung der Szene führen. Dafür wird in der vorliegenden Arbeit der Begriff der **gestörten Sprache** für KGs definiert:

$$\mathcal{L}_{noisy} \stackrel{def}{=} \{S \subseteq \mathcal{T};\ \ \exists S_0 \subseteq \mathcal{U}\ \exists I \in S_0 : \ \ s(I) = g \ \wedge \ S \xrightarrow{*} S_0\}$$

Natürlich gilt $\mathcal{L}_{pure} \subseteq \mathcal{L}_{noisy}$.

Eingangsdaten zugehöriger Parser sind damit Mengen von terminalen Instanzen, und alle Zielkonfigurationen von allen Reduktionen enthalten mindestens eine

[5] Man schreibt $\xrightarrow{*}$, wenn man auch $r = 0$ zuläßt, und $\xrightarrow{+}$, wenn $r \geq 1$ verlangt wird.

nichtterminale Instanz. Milgram und Rosenfeld verlangen das in [MILG] nicht. Damit kann auch eine Konfiguration von Terminalen auf der linken Seite einer Produktion stehen, also reduziert werden. Da sie auch nicht verlangen, daß das Startsymbol g ein Nichtterminal sein soll, gibt es eigentlich keinen Grund mehr, überhaupt zwischen Terminalen und Nichtterminalen zu unterscheiden, wie sie es immer noch tun. Es ist anzunehmen, daß diese Modifikationen gegenüber den normalen, syntaktischen Begriffen dadurch motiviert sind, daß in [MILG] dann eine Generalisierung der Koordinaten Grammatik vorgenommen wird, die das Prädikat π und die Funktion ϕ durch eine einzelne Relation ersetzt, in der die Koordinaten der Attributvektoren von Ausgangs- und Zielkonfiguration zueinander stehen. Damit sei die Definition symmetrisch und es erübrige sich, zwischen Generierung und Reduktion zu unterscheiden.

Die vorliegende Arbeit folgt dem nicht aus folgenden Gründen: Aus der Menge S wird das Bild der Ausgangskonfiguration entfernt. Dann wird das Bild der Zielkonfiguration hinzugefügt. Dabei muß man darauf achten, daß Instanzen aus κ_z durchaus schon in $S \backslash \mathcal{B}(\kappa_a)$ sein können. Das kommt in der Praxis (etwa in dem in Kapitel 6 beschriebenen Verfahren) sogar oft vor. Die Definition ist eben nicht symmetrisch. Darüberhinaus soll auf die Begriffsbildung der Sprache $\mathcal{L}$ zu einer KG nicht verzichtet werden. Die Begriffspaare Parser/Grammatik und Reduktion/Generierung behalten ihren Sinn. Ziel ist es zum einen, Parser zu finden, die entscheiden, ob eine gegebene terminale Instanzenmenge $T_{given} \subset \mathcal{T}$ Element der Sprache $\mathcal{L}_{noisy}$ ist (Detektionsaufgabe), und zum anderen solche Verfahren, die alle Instanzen des Startsymbols g, die sich daraus reduzieren lassen, konstruieren (Lokalisationsaufgabe).

Man kann die Produktionen einer KG auch **kumulativ** (also anhäufend) verwenden. Dabei wird darauf verzichtet, die Ausgangskonfiguration aus der Menge zu entfernen. Hierfür wird die folgende Schreibweise eingeführt:

$$S \longrightarrow_k S'$$

genau dann, wenn

$$\exists p = (\Lambda, \Sigma, \pi, \phi) \in P \; \exists \kappa_a \in \mathcal{A}_p \text{ mit } p: \; \kappa_a \overset{\pi}{\underset{\phi}{\mapsto}} \kappa_z$$

$$\text{so daß } \mathcal{B}(\kappa_a) \subset S \text{ und } S' = S \cup \mathcal{B}(\kappa_z)$$

und sagt 'S kulminiert direkt zu S''. Die transitive Hülle $\overset{*}{\longrightarrow}_k$ wird analog zur Reduktion definiert. Aus $S \overset{*}{\longrightarrow}_k S'$ folgt, wie man der Definition leicht entnimmt, $S \subset S'$. Man kann also nicht analog zur Reduktion einen Sprachbegriff bilden, der alle Teilmengen von $\mathcal{T}$ umfaßt, die zu einer Instanz des Startsymbols g kulminieren. Definiert wird aber analog zu $\mathcal{L}_{noisy}$ die kumulative Sprache einer

KG als

$$\mathcal{K} \stackrel{def}{=} \{S \subseteq \mathcal{T};\ \ \exists S_0 \subseteq \mathcal{U}\ \exists I \in S_0 :\ \ s(I) = g\ \wedge\ S \stackrel{*}{\longrightarrow}_k S_0\}.$$

Da man aus jeder Reduktion leicht eine Kulmination konstruieren kann, indem man die Ausgangskonfigurationen eben nicht entfernt, gilt $\mathcal{L}_{noisy} \subseteq \mathcal{K}$. Die Umkehrung gilt im allgemeinen nicht. Es kann sein, daß in einer Kulmination eine Instanz für mehrere Ausgangskonfigurationen verwendet wird.

Bisher sind ausschließlich Produktionen zugelassen, deren Bedingungsprädikat π nur auf der Ausgangskonfiguration definiert ist. Damit sind gewisse Konsistenzprüfungen, wie sie in der Praxis der Szeneninterpretation durch KI Verfahren häufig naheliegend und notwendig sind, nicht möglich. Es kann z. B. notwendig werden, zu verlangen, daß eine Instanz eines Fahrzeugtyps nur dort erzeugt werden darf, wo nichts anderes - kein anderes Fahrzeug, kein Haus, kein Baum usw. - ist. Produktionen, die ein zusätzliches **globales Konsistenz Bedingungsprädikat** der Gestalt

$$\psi\ =\ \neg\exists I \in S :\ s(I) \in \psi_V \subseteq V\ \wedge\ \psi_A(a(I), a(\kappa_a))$$

enthalten, werden zugelassen. Entsprechend heißen Koordinaten Grammatiken, die solche Produktionen enthalten, 'KG mit globaler Konsistenz Bedingung'. Für die Anwendung einer solchen Produktion muß also die Menge aller Instanzen in der Ausgangsmenge S, deren Symbolteil in ψ_V enthalten ist, und deren Attributteil zusammen mit dem Attributteil der Ausgangskonfiguration κ_a das Prädikat ψ_A erfüllt, leer sein. Für die in Kapitel 4 beschriebenen Implementierungsmöglichkeiten bedeuten solche Abfragen - insbesondere, wenn sie lokal im Attributraum bleiben - keinen wesentlich erhöhten Aufwand. Trotzdem sind sie zu vermeiden. Man handelt sich damit Probleme ein, die in der KI unter den Begriffen 'truth maintainance' und 'non monotonic reasoning' diskutiert werden. Der resultierende Reduktionsbegriff und damit auch der Sprachgriff muß für solche Systeme anders gefaßt werden, damit sinnvolle Bedeutungen möglich sind. Durch später, z. B. durch andere Reduktionen, erzeugte Instanzen kann eine Reduktion *im nachhinein* inkonsistent werden. Eine Kette direkter Reduktionen $S_0 \rightarrow ... \rightarrow S_n$ darf nur dann als Reduktion zusammengefaßt werden, wenn im i-ten Schritt ψ_i bezüglich aller S_j mit $0 \leq j \leq n$ gilt. Insgesamt ist es besonders für KGs mit globaler Konsistenz Bedingung ratsam, die Kumulationen statt der Reduktionen zu betrachten, da hier die 'Zwischenergebnisse' für die Konsistenzprüfung erhalten bleiben, und nicht erst wieder erzeugt werden müssen.

2.2.3 Beispiel einer Reduktion mit einer einfachen KG

Die folgende KG setzt Polygonzüge zusammen, die drei gerade Linien umfassen, die ein konvexes Viereck bilden, dem eine Seite fehlt:

$$
\begin{aligned}
T &= \{Linie\} \\
N &= \{L\_Linie,\ Winkel,\ O\_Viereck\} \\
D &= \{(P_1, P_2, P_3, P_4, O);\ P_i \in \{0, ... 179\}^2,\ O \in \{0, ... 179\}\} \\
n &= 5 \\
P &= \{p_1,\ p_2,\ p_3\} \\
g &= O\_Viereck
\end{aligned}
$$

Die Definition der Produktionen lautet:

$$
p_1 : (Linie, Linie) \xrightarrow[\substack{P_1 = P_{1,1} \\ P_2 = P_{2,2} \\ O = atan(P_{1,1}, P_{2,2})}]{\substack{col(P_{1,1}, P_{1,2}, P_{2,1}, P_{2,2}) \\ \wedge ovl(P_{1,1}, P_{1,2}, P_{2,1}, P_{2,2})}} (L\_Linie)
$$

$$
p_2 : (L\_Linie, Linie) \xrightarrow[\substack{P_1 = P_{1,1} \\ P_2 = int_{2d}(P_{1,1}, P_{1,2}, P_{2,1}, P_{2,2}) \\ P_3 = P_{2,2} \\ O = O_1}]{\substack{adj(P_{1,2}, P_{2,1}) \\ \wedge apa(O_1, O_2) \\ \wedge int_{2d}(P_{1,1}, P_{1,2}, P_{2,1}, P_{2,2}) \in D_2}} (Winkel)
$$

$$
\begin{array}{rl}
 & \begin{array}{l} adj(P_{1,2}, P_{2,1}) \\ \wedge apa(O_1, O_2) \\ \wedge spin(P_{1,1}, P_{1,2}, P_{2,2}) = spin(P_{2,1}, P_{2,2}, P_{2,3}) \\ \wedge int_{2d}(P_{1,1}, P_{1,2}, P_{2,1}, P_{2,2}) \in D_2 \end{array} \\
p_3 : (Linie, Winkel) & \longrightarrow (O\_Viereck) \\
 & \begin{array}{l} P_1 = P_{1,1} \\ P_2 = int_{2d}(P_{1,1}, P_{1,2}, P_{2,1}, P_{2,2}) \\ P_3 = P_{2,2} \\ P_4 = P_{2,3} \\ O = O_1 \end{array}
\end{array}
$$

p_1 faßt zwei überlappende (ovl), kollineare (col), Linien Instanzen zu einer verlängerten Linie zusammen. p_2 entspricht im wesentlichen der winkelbildenden Produktion aus dem Abschnitt 2.1.3. p_3 setzt eine Instanz des Startsymbols aus einer Linie und einem Winkel in ähnlicher Weise zusammen, wenn die Dreiecke $P_{1,1}P_{1,2}P_{2,2}$ und $P_{2,1}P_{2,2}P_{2,3}$ denselben Drehsinn aufweisen (also keine 'Z Lage' vorliegt). Die genaue Definition der Prädikate und Funktionen findet sich im Anhang B.

Man kann nun den Reduktionsbegriff verdeutlichen, indem man zu dieser Grammatik das Beispiel einer Menge $T_{given} \in \mathcal{T}$ angibt und $T_{given} \in \mathcal{L}_{noisy}$ durch Reduktion zeigt: Sei also

$$
T_{given} = S_1 = \left\{ \begin{array}{cccccc}
Linie & Linie & Linie & Linie & Linie & Linie \\
\begin{pmatrix} 30 \\ 30 \end{pmatrix} & \begin{pmatrix} 40 \\ 80 \end{pmatrix} & \begin{pmatrix} 102 \\ 156 \end{pmatrix} & \begin{pmatrix} 70 \\ 136 \end{pmatrix} & \begin{pmatrix} 120 \\ 80 \end{pmatrix} & \begin{pmatrix} 150 \\ 125 \end{pmatrix} \\
\begin{pmatrix} 50 \\ 10 \end{pmatrix} & \begin{pmatrix} 80 \\ 40 \end{pmatrix} & \begin{pmatrix} 70 \\ 113 \end{pmatrix} & \begin{pmatrix} 37 \\ 102 \end{pmatrix} & \begin{pmatrix} 170 \\ 80 \end{pmatrix} & \begin{pmatrix} 110 \\ 165 \end{pmatrix} \\
\begin{pmatrix} - \\ - \end{pmatrix} & \begin{pmatrix} - \\ - \end{pmatrix} & \begin{pmatrix} - \\ - \end{pmatrix} & \begin{pmatrix} - \\ - \end{pmatrix} & \begin{pmatrix} - \\ - \end{pmatrix} & \begin{pmatrix} - \\ - \end{pmatrix} \\
\begin{pmatrix} - \\ - \end{pmatrix} & \begin{pmatrix} - \\ - \end{pmatrix} & \begin{pmatrix} - \\ - \end{pmatrix} & \begin{pmatrix} - \\ - \end{pmatrix} & \begin{pmatrix} - \\ - \end{pmatrix} & \begin{pmatrix} - \\ - \end{pmatrix} \\
135^o & 135^o & 53^o & 53^o & 0^o & 135^o
\end{array} \right\}.
$$

Darin bilden die vierte und die dritte Linieninstanz eine Ausgangskonfiguration für die 'linienverlängernde' Produktion p_1. Somit gibt es die Reduktion $S_1 \xrightarrow{p_1} S_2$ mit

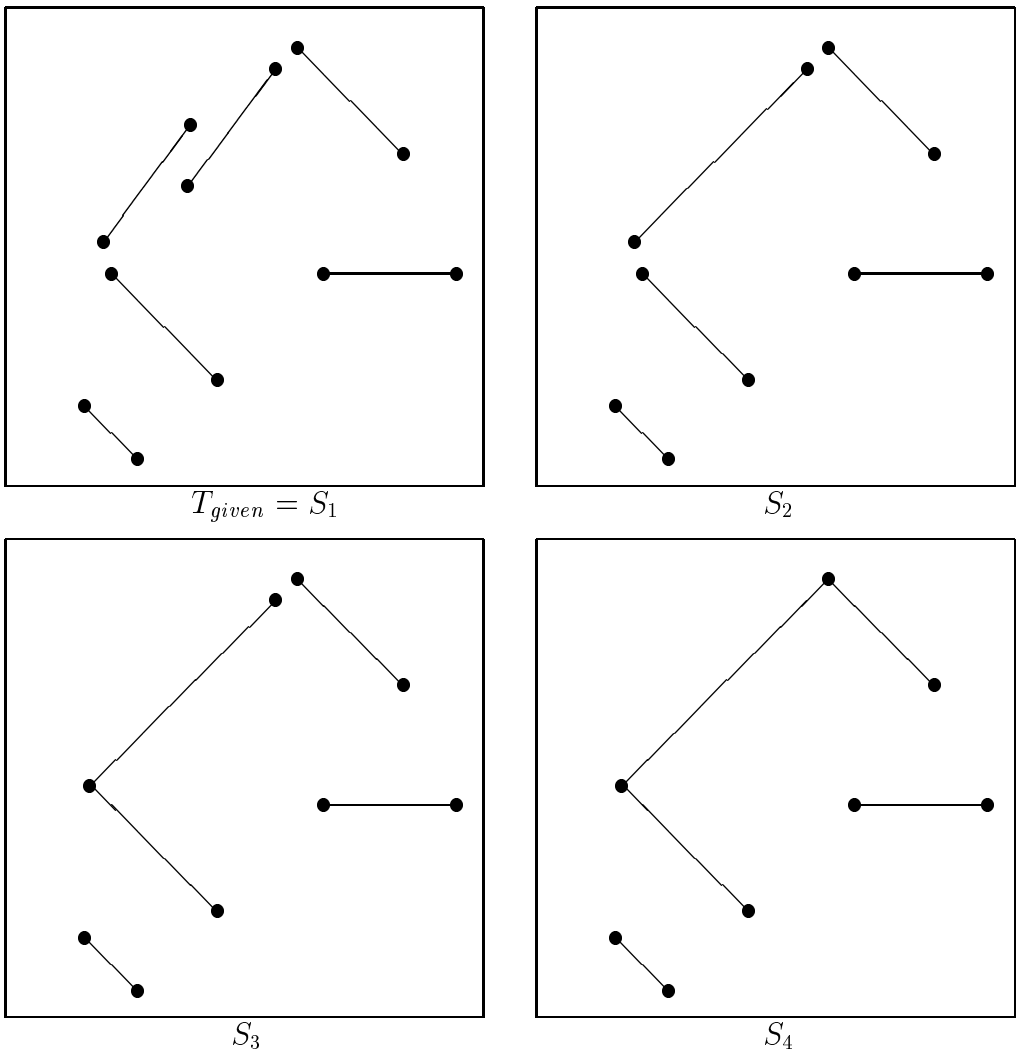


Figure 2.3: **Veranschaulichung einer Reduktion mit einer einfachen KG**
$T_{given} = S_1 \longrightarrow S_2 \longrightarrow S_3 \longrightarrow S_4 \in \mathcal{L}_{noisy}$

$$S_2 = \left\{ \begin{array}{ccccc} Linie & Linie & Linie & Linie & L\_Linie \\ \begin{pmatrix} 30 \\ 30 \end{pmatrix} & \begin{pmatrix} 40 \\ 80 \end{pmatrix} & \begin{pmatrix} 120 \\ 80 \end{pmatrix} & \begin{pmatrix} 150 \\ 125 \end{pmatrix} & \begin{pmatrix} 102 \\ 156 \end{pmatrix} \\ \begin{pmatrix} 50 \\ 10 \end{pmatrix} & \begin{pmatrix} 80 \\ 40 \end{pmatrix} & \begin{pmatrix} 170 \\ 80 \end{pmatrix} & \begin{pmatrix} 110 \\ 165 \end{pmatrix} & \begin{pmatrix} 37 \\ 102 \end{pmatrix} \\ \begin{pmatrix} - \\ - \end{pmatrix} & \begin{pmatrix} - \\ - \end{pmatrix} & \begin{pmatrix} - \\ - \end{pmatrix} & \begin{pmatrix} - \\ - \end{pmatrix} & \begin{pmatrix} - \\ - \end{pmatrix} \\ \begin{pmatrix} - \\ - \end{pmatrix} & \begin{pmatrix} - \\ - \end{pmatrix} & \begin{pmatrix} - \\ - \end{pmatrix} & \begin{pmatrix} - \\ - \end{pmatrix} & \begin{pmatrix} - \\ - \end{pmatrix} \\ 135^o & 135^o & 0^o & 135^o & 45^o \end{array} \right\}.$$

Die L_Linie Instanz bildet hierin zusammen mit der zweiten Linie eine Ausgangskonfiguration für die winkelbildende Produktion p_2, wodurch die Reduktion $S_2 \xrightarrow{p_2} S_3$ möglich wird mit

$$S_3 = \left\{ \begin{array}{cccc} Linie & Linie & Linie & Winkel \\ \begin{pmatrix} 30 \\ 30 \end{pmatrix} & \begin{pmatrix} 120 \\ 80 \end{pmatrix} & \begin{pmatrix} 150 \\ 125 \end{pmatrix} & \begin{pmatrix} 102 \\ 156 \end{pmatrix} \\ \begin{pmatrix} 50 \\ 10 \end{pmatrix} & \begin{pmatrix} 170 \\ 80 \end{pmatrix} & \begin{pmatrix} 110 \\ 165 \end{pmatrix} & \begin{pmatrix} 32 \\ 97 \end{pmatrix} \\ \begin{pmatrix} - \\ - \end{pmatrix} & \begin{pmatrix} - \\ - \end{pmatrix} & \begin{pmatrix} - \\ - \end{pmatrix} & \begin{pmatrix} 80 \\ 40 \end{pmatrix} \\ \begin{pmatrix} - \\ - \end{pmatrix} & \begin{pmatrix} - \\ - \end{pmatrix} & \begin{pmatrix} - \\ - \end{pmatrix} & \begin{pmatrix} - \\ - \end{pmatrix} \\ 135^o & 0^o & 135^o & 45^o \end{array} \right\}.$$

Darin wiederum bildet die Winkel Instanz mit der letzten Linien Instanz zusammen eine Ausgangskonfiguration für die auf das Startsymbol führende Produktion p_3. Es ergibt sich die Reduktion $S_3 \xrightarrow{p_3} S_4$ mit

$$S_4 = \left\{ \begin{array}{ccc} Linie & Linie & O\_Viereck \\ \begin{pmatrix} 30 \\ 30 \end{pmatrix} & \begin{pmatrix} 120 \\ 80 \end{pmatrix} & \begin{pmatrix} 150 \\ 125 \end{pmatrix} \\ \begin{pmatrix} 50 \\ 10 \end{pmatrix} & \begin{pmatrix} 170 \\ 80 \end{pmatrix} & \begin{pmatrix} 110 \\ 165 \end{pmatrix} \\ \begin{pmatrix} - \\ - \end{pmatrix} & \begin{pmatrix} - \\ - \end{pmatrix} & \begin{pmatrix} 32 \\ 97 \end{pmatrix} \\ \begin{pmatrix} - \\ - \end{pmatrix} & \begin{pmatrix} - \\ - \end{pmatrix} & \begin{pmatrix} 80 \\ 40 \end{pmatrix} \\ 135^o & 0^o & 135^o \end{array} \right\}$$

und damit insgesamt die Reduktion $T_{given} \xrightarrow{*} S_4$.

In Abbildung 2.3 ist diese Reduktion verdeutlicht. Weil S_4 eine Instanz des Startsymbols enthält, gilt $T_{given} \in \mathcal{L}_{noisy}$ und natürlich auch $T_{given} \in \mathcal{K}$. Es bleiben aber zwei 'störende' Linien Instanzen, die nicht reduziert werden können. Somit ist T_{given} nicht in $\mathcal{L}_{pure}$ enthalten.

2.3 Hierarchie der Koordinaten Grammatiken

Milgram und Rosenfeld schlagen in [MILG] vor, das Prädikat π und die Funktion ϕ einer Produktion - die sie als eine Relation zusammenfassen - stark einzuschränken, um zu einer Hierarchie in der algorithmischen Mächtigkeit analog zur Chomsky Hierarchie der String Grammatiken zu kommen.[6] Die vorliegende Arbeit folgt dem nicht, denn damit würde man die Aussagekraft und somit die Brauchbarkeit der KGs in der Praxis unnötig einschränken.

Es erscheint dagegen sinnvoll, nach dem Vorbild der Chomsky Hierarchie zunächst von der allgemein rekursiven Mächtigkeit der allgemeinen Grammatiken herunterzusteigen auf die primitiv rekursive Mächtigkeit der kontextsensitiven Grammatiken. Bei den KGs finden sich hier die monotonen Grammatiken. Damit ist man in einem Bereich, der sinnvoll algorithmisch angegangen werden kann (was im Kapitel 3 geschieht). Um praktisch nutzbare Verfahren implementieren zu können, ist es dann erforderlich, weiter hinunterzusteigen zu den reduktionsbaumbildenden KGs, die als Analogon zu den kontextfreien Stringgrammatiken gesehen werden können, und zu den nicht rekursiven KGs. Diese spezielleren Klassen schränken aber die Ausdrucksfähigkeit der zugehörigen Sprachen ein. Man kann eine gewisse Beschleunigung erreichen, indem man zu den binärbaumbildenden KGs übergeht. Das entspricht in etwa dem Übergang zu den Normalformen bei den Stringgrammatiken, bedeutet also keine Einschränkung in der Ausdruckskraft. Im Abschnitt 3.2 wird für den Rechenaufwand zum Parsen von binärbaumbildenden KGs eine obere Schranke bestimmt. Diese ist größer als polynomial in der Kardinalität der Eingangsdaten. Mit der in Abschnitt 2.3.3 durchgeführten Reduktion eines NP-vollständigen Problems auf eine reduktionsbaumbildende KG würde aus einer polynomialen Parsemethode für solche KGs NP=P folgen. Das ist nicht zu erwarten. Es werden mit den nicht rekursiven KGs und den endlich attributierten KGs aber auch Klassen von KGs gegeben, die im Aufwand polynomial bleiben. Um Faustregeln für die Konstruktion von praktisch nutzbaren Grammatiken und zugehöriger Hardware aufzustellen, braucht man noch einige wichtige Spezialbegriffe. Die Definitionen der einzelnen Stufen der Hierarchie finden sich im Abschnitt 2.3.1. Im Abschnitt 2.3.2 wird ein wichtiges diesbezügliches Resultat von Milgram und Rosenfeld diskutiert. Der Abschnitt

[6] Die entsprechende Stelle ist im Abschnitt 2.3.2 zitiert.

2.3.3 geht auf Querverbindungen ein, die sich zu einigen der im Abschnitt 7.3 skizzierten Strukturen durch Einbettung aufzeigen lassen. Es wird eine Übersicht über die Stellung der KG Hierarchie bzgl. einiger Standardstrukturen der theoretischen Informatik gegeben.

2.3.1 Klassen von Koordinaten Grammatiken und ihre Eigenschaften

1. **Monotone KGs**

   - Defintion: Eine Produktion $p = (\Lambda, \Sigma, \pi, \phi) \in P$ heißt **isometrisch** genau dann, wenn das Wort Σ gleichlang ist wie das Wort Λ.
   - Defintion: Eine Produktion $p = (\Lambda, \Sigma, \pi, \phi) \in P$ heißt **monoton** genau dann, wenn das Wort Λ kürzer oder höchstens gleichlang ist wie das Wort Σ.
   - Defintion: Eine Produktion $p = (\Lambda, \Sigma, \pi, \phi) \in P$ heißt **streng monoton** genau dann, wenn das Wort Λ echt kürzer ist als das Wort Σ.

   - Satz: Sei G eine KG mit ausschließlich streng monotonen Produktionen und S eine endliche Menge von terminalen Instanzen des zugehörigen Vokabulars. Dann kann eine Kette von reduzierenden Mengen $S = S_1, ..., S_r$ mit $S_i \longrightarrow S_{i+1}$ höchstens $|S| - 1$ Glieder haben. Der Beweis geht offensichtlich durch Induktion über die stets kleiner werdende Anzahl der Elemente in den Mengen.
   - Bemerkung: Mit der so beschränkten Länge der möglichen Reduktionen ergibt sich auch eine Einschränkung der *Anzahl der möglichen Reduktionen*. Wenn k_i die Länge der rechten Seite der $i-$ten Produktionen in P ist, so ergeben sich höchstens $|S|^{k_i}$ Möglichkeiten für eine direkte Reduktion mit dieser Produktion, insgesamt also $\sum_{i=1}^{|P|} |S|^{k_i}$ für die erste direkte Reduktion. Die Gesamtzahl der Reduktionen ist mithin beschränkt durch

     $$\prod_{j=0}^{|S|-1} \sum_{i=1}^{|P|} (|S| - j)^{k_i}.$$

     Damit ist eine obere Schranke für die Anzahl der nötigen algorithmischen Schritte zur Entscheidung der Wortprobleme $S \stackrel{?}{\in} \mathcal{L}_{pure}$ und $S \stackrel{?}{\in} \mathcal{L}_{noise}$ gefunden, die nur von $|S|$, $|P|$ und den k_i abhängt. Die streng monotonen KGs entsprechen also in der Chomsky Hierarchie höchstens dem Typ 1 (kontextsensitive String Grammatiken), in der

Theorie der rekursiven Funktionen höchstens den primitiv rekursiven Funktionen.

Anders liegt der Fall bezüglich der kumulativen Sprache $\mathcal{K}$. Hier werden die Mengen mit jeder Kulmination auf jeden Fall größer. Solch ein Argument ist dann also nicht möglich.

- Bemerkung: In der Praxis ist die obige Abschätzung der Anzahl der möglichen Reduktionen sehr grob. Gäbe es wirklich auch nur annähernd so viele, so wäre die Grammatik untauglich für die Szenenanalyse.
- Verallgemeinerung: Es kann ebenfalls eine obere Schranke für monotone Grammatiken angegeben werden, wenn damit keine rekursiven Zyklen aus lauter isometrischen Produktionen möglich sind.

2. **Reduktionsbaumbildende KGs**

- Definition: Eine Produktion $p = (\Lambda, \Sigma, \pi, \phi) \in P$ heißt **reduktionsbaumbildende** Produktion, wenn die linke Seite nur eine Instanz enthält, also $|\Lambda| = 1$. Es wird dafür die Abkürzung RBB-Produktion eingeführt. KGs die nur solche Produktionen enthalten, heißen entsprechend RBB-KGs.
- Satz: RBB-KGs sind monoton. Das liegt natürlich daran, daß die Ausgangskonfigurationen per Definitionem mindestens eine Instanz enthalten.
- Bemerkung: Der Name weist darauf hin, daß die Reduktion eines Nichtterminals in einer solchen KG als Baum kodiert werden kann.[7] Eine Kante führt jeweils von dem einen Nichtterminal in der Zielkonfiguration zu den Instanzen der Ausgangskonfiguration. Strenggenommen kann es vorkommen, daß im Verlaufe einer Reduktion mit einer RBB-KG eine Instanz, die schon einmal herausgenommen wurde, wieder erzeugt wird. Dann ist der Reduktionsgraph natürlich kein Baum mehr, sondern enthält einen Zyklus. Insbesondere ist der Reduktionsgraph der Teilreduktion vom Herausnehmen bis zum Wiedereinsetzen der kritischen Instanz zyklisch. Ohne Einschränkung der Allgemeinheit kann angenommen werden, daß in diesem zyklischen Teilgraph kein noch kleinerer Zyklus enthalten ist. Dann ist dieser Teilgraph baumförmig bis auf diesen einen Zyklus. Er führt nur dazu, daß die Terminale, die die Blätter seines nichtzyklischen Teils bilden, aus der Ausgangsmenge entfernt werden. Bzgl. $\mathcal{L}_{noisy}$ ergibt sich kein Unterschied. Die Reduktion kann ersetzt werden durch eine Teilreduktion, die zur selben Instanz führt, indem man den zyklischen Teil

[7] Auf Reduktionsgraphen wird im Abschnitt 5.1.1 noch näher eingegangen.

entfernt. Es gilt also nicht, daß jede Reduktion baumförmig ist, aber die minimale ist es immer.

- Beispiel für eine monotone nicht-RBB-Produktion: In der Praxis benötigt man Produktionen, die mehr als eine Instanz in der Zielkonfiguration enthalten, dann, wenn man Kontext - im umgangssprachlichen Sinn - einbeziehen will. Ein Beispiel findet sich in [FUE-90-1, STIL-91]. Dort wird zur Siedlungsanalyse in Luftbildern eine Produktion konstruiert, die aus einer Instanz des Symbols *block_row* und einer Instanz des Symbols *road* eine Instanz des Symbols *house_row* bildet, wenn die *road* Instanz und die *block_row* Instanz hinreichend benachbart sind und parallel verlaufen. Damit kommt zum Ausdruck, daß man das Wissen, daß Häuser an Straßen stehen, nutzen möchte, um sie als solche zu erkennen. In einem reduzierenden Produktionssystem, wie der KG, müßte man dafür natürlich schreiben

$$(block\_row, road) \underset{copy}{\overset{\pi}{\longrightarrow}} (house\_row, road),$$

  denn die Straße steht zur Häuserreihe nicht in einer Teil-von Relation, sondern dient als Kontext. Im Prädikat π werden dabei Nachbarschaft und Parallelität geprüft. Die zugehörige Funktion ist trivial, sie kopiert die Attributwerte aus der Ausgangskonfiguration in die Zielkonfiguration. Wichtig ist, daß die *road* Instanz durch diese Produktion nicht aus der aktuellen Menge entfernt wird. Sie kann also wieder verwendet werden, z. B. um auf der anderen Straßenseite ebenfalls eine *block_row* zu einer *house_row* zu reduzieren. Reduktionsgraphen von KGs mit solchen Produktionen sind also im allgemeinen nicht baumförmig.

- Bemerkung: Die Frage, ob zu jeder monotonen KG eine RBB-KG gefunden werden kann, die dieselben Sprachen $\mathcal{L}_{pure}$ und $\mathcal{L}_{noisy}$ generiert, ist von relativ geringem Interesse. Zu bedenken ist dabei, daß keinerlei Einschränkungen bzgl. der zu den Produktionen gehörigen Prädikate und Funktionen gemacht wurden. Man kann also auch ganze, rekursive Algorithmen darin 'verstekken'. Durch geschickte Kodierung mag es möglich sein, die ganze monotone KG in dem Prädikat und der Funktion einer einzigen RBB-Produktion unterzubringen. Damit ist nichts gewonnen. Das Beispiel oben macht klar, daß es in der Praxis Wissen gibt, das am geschicktesten mit solchen 'kontext-sensitiven' Produktionen erfaßt wird.

- Konstruktion: Aus jeder KG kann leicht eine RBB-KG konstruiert werden, die dieselbe kumulative Sprache $\mathcal{K}$ erzeugt. Dazu sei j die Anzahl der Instanzen der Zielkonfiguration einer Produktion p. Dann ersetzt man p durch j RBB-Produktionen p'_i, die jeweils nur eine der

Instanzen der Zielkonfigurationen reduzieren. Dazu wird auch die zu p gehörige Funktion ϕ aufgesplittet in ihre j Komponenten. Am Bedingungsprädikat π wird nichts geändert. In jeder Kumulation kann die Produktion p ersetzt werden durch das Nacheinanderausführen aller p'_i auf derselben Ausgangskonfiguration, die ja nicht entfernt wird. Andersherum kann jede Kulmination mit einem p'_i auch ersetzt werden durch eine mit p selbst, denn die Zielkonfiguration von p'_i ist ja als Teilmenge in der von p enthalten.

- Folgerung: Der fundamentale Unterschied zwischen RBB-KGs und graphbildenden KGs fällt für die Betrachtung der kumulativen Sprache weg. Hier stellen die RBB-KGs eine Art Normalform dar, auf die alle KGs gebracht werden können. Man beschränkt sich trotzdem auf jeden Fall auf monotone Grammatiken
- Definition: Eine RBB-Produktion $p = (\Lambda, \Sigma, \pi, \phi)$ heißt **binärbaumbildende** Produktion, wenn die Ausgangskonfiguration höchstens zwei Instanzen enthält, also $|\Sigma| \leq 2$. Es wird dafür die Abkürzung BBB-Produktion eingeführt. KGs die nur solche Produktionen enthalten, heißen entsprechend BBB-KGs.
- Konstruktion: Sei $p = ((\lambda), (\sigma_1, ..., \sigma_k), \pi, \phi)$ eine RBB-Produktion einer KG mit Attributraum D. Dann wird eine Grammatik eingeführt, deren Attributraum $D' = D^{k-1}$ ist. Induktiv werden die Produktionen p'_i und die Nichtterminale σ'_i für $i = 2, ..., k$ definiert durch: σ'_2 ist ein geeigneter Name für das Paar (σ_1, σ_2) und $p'_2 = ((\sigma'_2), (\sigma_1, \sigma_2), \pi'_2, copy)$ reduziert eine Instanz dieses Symbols, wenn die Projektion des Prädikats π auf die ersten beiden Komponenten π'_2 erfüllt ist. Dabei werden die Attributwerte kopiert. Für $i = 3, ..., k-1$ ist σ'_i ein geeigneter Name für das Paar $(\sigma'_{i-1}, \sigma_i)$ und $p'_i = ((\sigma'_i), (\sigma'_{i-1}, \sigma_i), \pi'_i, copy)$ reduziert eine Instanz dieses Symbols, wenn die Projektion des Prädikats π auf die ersten i Komponenten π'_i erfüllt ist. Dabei werden die Attributwerte kopiert. $p'_k = ((\lambda), (\sigma'_{k-1}, \sigma_k), \pi', \phi')$ entspricht dann p. Dabei sind π' und ϕ' auf $D^{k-1} \times D$ genauso definiert wie π und ϕ auf D^k.

3. **Nicht rekursive KGs**

- Definition: Eine KG heißt **nicht rekursiv**, wenn das zugehörige Produktionsnetz zyklenfrei ist (siehe Abschnitt 5.1.2).
- Bemerkung: Nicht rekursive KGs sind nicht notwendig RBB-KGs. Sie sind auch nicht notwendig monoton. Die Tiefe einer Reduktion $S_0 \xrightarrow{*} S$ ist bei ihnen unabhängig von der Kardinalität $|S_0|$ beschränkt durch die KG selbst. Damit lassen sich polynomiale Schranken für den Rechenaufwand angeben.

4. **Lokale Produktionen**

- Definition: Sei D_i eine Komponente eines Attributraums einer KG, auf der eine Metrik d gegeben ist. Eine Produktion $p = (\Lambda, \Sigma, \pi, \phi) \in P$ heißt dann bzgl. dieser Komponente **lokal** genau dann, wenn es einen Maximalabstand d_{max} gibt, so daß $\pi(a(I_1), ..., a(I_k))$ impliziert, daß $d(a_i(I_j), a_i(I_{j'})) < d_{max}$ gilt für alle j und j' zwischen 1 und k.
- Bemerkung: Diese Definition macht zunächst nur für Komponenten einen Sinn, die einen unendlichen Abstand zulassen. Andernfalls kann man für d_{max} eine Zahl setzen, die größer ist als der in dieser Komponente des Attributbereichs mögliche Maximalabstand. So gesehen ist jede Produktion z. B. auf einem endlichen Bildbereich lokal. Informell soll aber von einer lokalen Produktion nur dann gesprochen werden, wenn d_{max} um Größenordnungen kleiner ist als der mögliche Maximalabstand. In diesem Sinne sind Produktionen, die als Bedingungsprädikat Nachbarschaften in der selben Komponente verwenden, lokal bzgl. dieser Komponente.
- Definition: Sei $\mathcal{I} \subseteq \{1, ..., n\}$ eine Indexmenge, so daß für alle $i \in \mathcal{I}$ $D_i = M$ gilt, wobei auf dem Raum M die Metrik d definiert ist. Eine Produktion $p = (\Lambda, \Sigma, \pi, \phi) \in P$ heißt dann bzgl. M **lokal** genau dann, wenn es einen Maximalabstand d_{max} gibt, so daß $\pi(a(I_1), ..., a(I_k))$ impliziert, daß $d(a_i(I_j), a_{i'}(I_{j'})) < d_{max}$ gilt für j und j' zwischen 1 und k und $i, i' \in \mathcal{I}$.
- Bemerkung: Für beschränkte metrische Räume M gelten die obigen informellen Konventionen. Betrachtet man z. B. Produktionen wie aus Abschnitt 2.2.3, und setzt hier als metrischen Raum das Bild (mit euklidischer Metrik), so gelten die Produktionen als bzgl. dieses Bildes lokal, wenn alle Bildkomponenten der Attributvektoren ihrer Ausgangskonfigurationen auf einem eng begrenzten Ausschnitt des Bildes beieinander liegen müssen. Die Instanzen müssen insbesondere klein sein relativ zur Gesamtgröße des Bildes.
- Definition: Eine bzgl. einer Indexmenge $\mathcal{I}$ und eines metrischen Raumes M lokale Produktion $p = (\Lambda, \Sigma, \pi, \phi)$ heißt darüberhinaus **ziel-lokal**, wenn die Abstandsforderung $d(a_i(I_j), a_{i'}(I_{j'})) < d_{max}$ nicht nur für alle Instanzen der rechten Seite Σ gilt, sondern für alle Instanzen aus $\mathcal{B}(\Sigma) \cup \mathcal{B}(\Lambda)$.
- Bemerkung: Natürlich sind lokale Produktionen nicht immer ziel-lokal. Wenn etwa keine Forderung bzgl. Antiparallelität gestellt wird, kann der Schnittpunkt zweier Linien, deren Endpunkte benachbart sind, beliebig weit entfernt liegen.
- Bemerkung: In [ROSE-89-1] heißen ziel-lokale KGs *C-grammars of bounded diameter*.

5. **Invarianzen**

- Definition: Sei D_i eine Komponente eines Attributbereichs einer KG, auf der eine additive Gruppenoperation + definiert ist (also z. B. ein Vektorraum) und $\delta \in D_i$. Es ist dann eine Operation $shift : D \times D_i \to D$ definiert durch $shift_i(a, \delta) = a_i + \delta$ und $shift_{i'}(a, \delta) = a_{i'}$ für $i' \neq i$. Eine Produktion $p = (\Lambda, \Sigma, \pi, \phi) \in P$ heißt dann bzgl. der Komponente D_i **verschiebungsinvariant** genau dann, wenn für alle δ und alle Attributvektoren a die beiden folgenden Identitäten gelten:

$$\pi(shift(a, \delta)) = \pi(a)$$

$$\phi_{i'}(shift(a, \delta)) = \begin{cases} \phi_i(a) + \delta & i' = i \\ \phi_{i'}(a) & i' \neq i \end{cases}$$

- Bemerkung: Für begrenzte Komponenten des Attributbereichs stößt dieser Begriff i. a. auf Schwierigkeiten. Ihnen fehlt die Gruppenstruktur. Verschiebt man z. B. einen Bildpunkt um einen gewissen Translationsvektor, so kann er dadurch außerhalb der Grenzen zu liegen kommen. Informell soll aber auch in diesen Fällen die Rede von verschiebungsinvarianten Produktionen sein. Man betrachtet dann nur die Attributwerte, die hinreichend weit im Inneren liegen und nur die Translationen, die nicht hinausführen können.
- Definition: Sei $\mathcal{I} \subseteq \{1, ..., n\}$ eine Indexmenge, so daß für alle $i \in \mathcal{I}$ $D_i = G$ gilt, wobei G eine additiv geschriebene Gruppe ist und δ ein Element daraus. Es ist dann eine Operation $shift : D \times G \to D$ definiert durch $shift_i(a, \delta) = a_i + \delta$ für $i \in \mathcal{I}$ und $shift_i(a, \delta) = a_i$ sonst. Eine Produktion $p = (\Lambda, \Sigma, \pi, \phi) \in P$ heißt dann bzgl. G **verschiebungsinvariant** genau dann, wenn

$$\pi(shift(a, \delta)) = \pi(a)$$

$$\phi_i(shift(a, \delta)) = \begin{cases} \phi_i(a) + \delta & i \in \mathcal{I} \\ \phi_i(a) & \text{sonst} \end{cases}$$

- Bemerkung: Das läßt sich natürlich verallgemeinern auf beliebige Automorphismen $f : D \to D$, also etwa auch auf Drehungen, Skalierungen, Scherungen oder affine Abbildungen.
- Bemerkung: In [ROSE-89-1] wird für verschiebungsinvariante, ziellokale KGs mit $D = Z$, die folgende Notation für eine Produktion $p = (\Sigma, \lambda, \pi, \phi)$ eingeführt:

$$p \stackrel{def}{=} (\alpha, \beta), \text{ wobei } \alpha \stackrel{def}{=} (\Sigma_1, ..., \Sigma_k | i_1, ..., i_k) \text{ mit } 0 = i_1 < ... < i_k \leq d_{max}$$

$$\text{und } \beta \stackrel{def}{=} (\Lambda_1, ..., \Lambda_j | h_1, ..., h_j) \text{ mit } -d_{max} \leq h_1 < ... < h_j \leq d_{max} \text{ gilt.}$$

Dabei gibt die aufsteigende Zahlenfolge $i_1, ..., i_k$ die Positionen an, an denen die Symbole relativ zueinander liegen müssen, damit π erfüllt ist. Wenn π mehrere solche $k-$Tupel zuläßt, wird man eben entsprechend viele Produktionen in dieser Notation erzeugen. Die Positionenfolge $h_1, ..., h_j$ kodiert die Funktion ϕ. Ähnlich wie bei der Einbettung der Stringgrammatiken (siehe Abschnitt 2.3.3) führt Rosenfeld auch hier ein nichtterminales Leerzeichen ein. Diese Struktur kann in Beziehung gesetzt werden zu den isometrischen Grammatiken. Sie ist umfassender, und es lassen sich Zusatzbedingungen bestimmen, unter denen sie isomorph ist. Damit ergeben sich Hierarchien, die der Chomsky Hierarchie vergleichbar sind. Es kann relativ leicht auf Dimensionen $d \geq 2$ verallgemeinert werden (siehe Abschnitt 7.3.3). Diese Arbeiten wurden fortgesetzt durch A. Nakamura einerseits in ihrer Relation zu *path controlled graph grammars* [NAKA-89] und andererseits in Richtung parallele Abarbeitung [NAKA-95].

6. **Relativvolumina von Produktionen**

- Definition: Zu jeder Produktion $p = (\Lambda, \Sigma, \pi, \phi) \in P$ gehört das **Relativvolumen** der Menge der zugehörigen Ausgangskonfigurationen im Vergleich zu allen Konfigurationen der Länge k vermöge der folgenden Definition für endlich attributierte Grammatiken

$$V_{rel}(p) = \frac{|\mathcal{A}_p|}{|D^k|}$$

und der folgenden Verallgemeinerung für unendliche Komponenten: Ist eine Komponente $d_i = Z$ so gilt als $V_{rel}(p)$ der Grenzwert des Relativvolumens für ins Unendliche wachsende Intervalle: Man setzt $d_i = [-d_{max}, d_{max}]$ für eine natürliche Zahl d_{max}, definiert die entsprechende endlich attributierte KG mit denselben Produktionen eingeschränkt auf das endliche Intervall, und erhält so ein von d_{max} abhängiges Relativvolumen $V_{rel}(p, d_{max})$. Mithin ergibt sich die Definition des Relativvolumens der unendlich attributierten KG durch

$$V_{rel}(p) = \lim_{d_{max} \to \infty} V_{rel}(p, d_{max}).$$

Natürlich ist hier eine Verallgemeinerung auf mehrdimensionale unendliche Komponenten leicht möglich. Bei mehreren unendlichen Komponenten kann das Relativvolumen nur sinnvoll definiert werden, wenn gezeigt ist, daß die Grenzwertbestimmung unabhängig von der Reihenfolge ist, in der die Grenzwerte durchgeführt werden.

- Bemerkung: Für viele Symbole haben nicht alle Komponenten des Attributbereichs eine Bedeutung. Im Beispiel aus Abschnitt 2.2.3 etwa

ist das für die Bildkoordinaten P_3 und P_4 bei Instanzen des Symbols *Linie* und *L_Linie* der Fall. Beim Niederschreiben einer Instanz ist dafür das Zeichen - gesetzt. Keine Produktion, in deren rechter Seite ein solches Symbol erscheint, enthält ein Bedingungsprädikat, das von solch einer Komponente abhängt. Wegen dieser Unabhängigkeit kann man die entsprechenden Möglichkeiten aus dem Bruch $\frac{|\mathcal{A}_p|}{|D^k|}$ herauskürzen und sich bei der Berechnung von $V_{rel}(p)$ auf die Komponenten des Attributbereichs beschränken, die im Bedingungsprädikat vorkommen.

2.3.2 Algorithmische Mächtigkeit und anschauliche Aussagekraft

Für endlich attributierte KGs - nur mit solchen hat man es in der Praxis zu tun - ist die folgende Argumentation möglich: Die Menge der terminalen Instanzen ist endlich und mit ihr ihre Potenzmenge. Jede solche Grammatik kann also durch ein Lexikon ersetzt werden, welches die Sprache $\mathcal{L}_{pure}$ auflistet. Solche Überlegungen helfen aber nicht viel weiter. R. Narasimhan schreibt schon 1969 in [NARA-69] (Seite 13):

- "... If, however, we also specify that the tokens have finite resolution in all their attributes (i. e. the set of attributes is finite and also each attribute has a finite range), then the cardinality of the class becomes strictly finite. It must be emphasized that practically all real life applications of picture generation and interpretation belong to this restricted variety. But this does not trivalize the problem. In fact the finitness of the token class has intrinsically very little relevance to the picture analysis and description problem. This is a fundamental assumption in our entire approach as presented in this paper."

Dem schließt sich diese Arbeit an. Die Potenzmenge aller möglichen terminalen Instanzen ist im nichttrivialen Fall zwar endlich, aber unüberschaubar groß. Ganz besonders gilt dies für echte Szenenanalysegrammatiken, wie sie etwa in Kapitel 6 vorgestellt sind. Aber auch schon relativ einfache Beispiel KGs generieren zwar endliche aber sehr große Sprachen, wie die Abschätzung im Abschnitt 3.1 zeigt.

Zur algorithmischen Mächtigkeit nicht endlich attributierter KGs geben Milgram und Rosenfeld das folgende Resultat in [MILG], das als **Hauptsatz** der Theorie der Koordinaten Grammatiken gelten kann: Man kann Maschinen, wie sie in [MINS] unter der Bezeichnung 'Universale Programm Maschinen mit einem Register' dargestellt sind, durch eine sehr einfache KG simulieren. Dazu benötigt man nur ein einziges unendliches Attribut. Jeder Zustand der Maschine wird

durch ein nichtterminales Symbol kodiert. Die Übergänge von einem Zustand zum nächsten und die Sprünge auf andere Zustände sind durch Produktionen kodiert, die also jeweils nur eine einelementige Ausgangskonfiguration in eine einelementige Zielkonfiguration verwandeln. Es wird zu den Symbolen eine Koordinate assoziiert, die den Registerinhalt des einzigen Registers der Minskymaschine kodiert. Als Prädikate auf dieser Koordinate braucht man lediglich Teilbarkeit durch die Zahlen 2 und 3 und als Funktionen nur die Multiplikation sowie Division mit bzw. durch 2 und 3. Startsymbol der Grammatik ist das Symbol, das den Startzustand kodiert, und einzig der Haltezustand ist durch ein Terminal kodiert. Bei [MINS] kann man dann nachlesen, wie diese Maschine durch eine Art kleine Gödelisierung Maschinen mit drei Registern simuliert, die wiederum Turingmaschinen simulieren. Somit läßt sich prinzipiell jede allgemein rekursive Funktion durch so extrem einfache KGs berechnen. Milgram und Rosenfeld schließen mit der Bemerkung ([MILG] Seite 191):

- "The foregoing observations indicate that if one wishes to develop a computational hierarchy for GRG's, it will be necessary to restrict the coordinate-computing relations even further. For example, one could require all functions to be of the form $x \neq k$. Since GRG's appear to be quite usefull in picture analysis and synthesis, their theory certainly deserves further study."

Die Einbettung von Turingmaschinen, die im folgenden Abschnitt dieser Arbeit durchgeführt wird, benötigt als Prädikat π lediglich die Nachfolgerelation zwischen zwei ganzen Zahlen, und die Funktion ϕ kopiert die Attributwerte nur. Es erscheint also ziemlich aussichtslos, eine Hierarchie auf den Koordinaten Grammatiken durch Einschränkungen auf den Klassen der für π und ϕ zugelassenen Prädikate und Funktionen erzwingen zu wollen. Um der Ausdrucksfähigkeit Willen ist es im Gegenteil geboten, hier keinerlei Einschränkungen zu machen, sondern diese Prädikate und Funktionen als 'primitiv'-Rechenschritte der Komplexitätsanalyse zu betrachten.

Was die Hierarchie angeht, so wurde im Abschnitt 2.3.1 gezeigt, daß monotone KGs die erste wichtige Stufe unter den allgemeinen KGs sind. Im Abschnitt 2.3.3 wird gezeigt, daß es NP-vollständige Probleme gibt, die durch Reduktion mit solchen KGs entschieden werden. Will man also Schwierigkeiten mit der Rechenzeit vermeiden, so sollte man erstens nicht rekursive KGs verwenden und zweitens die Statistik auf den gegebenen Ausgangsmengen T_{given} zusammen mit den Relativvolumina der Produktionen betrachten, wie dies im Abschnitt 5.3 geschieht. Letzteres macht man für jede einzelne KG, und nicht pauschal für ganze Klassen von KGs.

2.3.3 Einbettung anderer Strukturen

Auf ganz natürliche Weise kann jede Turingmaschine durch eine einfache isometrische KG dargestellt werden. Die vorliegende Arbeit folgt hier allerdings nicht dem in [MILG] vorgeschlagenen Umweg über die Minsky Maschinen, sondern führt die Einbettung so durch, daß die Unterschiede in der algorithmischen Struktur zwischen der Turingmaschine und der KG deutlich werden.

- Eine **Turingmaschine** sei gegeben durch ein Quadrupel (Z, A, t, z_0), wobei die einzelnen Symbole für folgende Strukturen stehen:
  - Z ist eine endliche Menge von **Zuständen**. Sie enthält eine nichtleere Menge von **Endzuständen** F.
  - A ist ein endliches **Alphabet** von Zeichen. Es enthält auch das **Leerzeichen** $*$.
  - t ist eine Funktion, die jedem Paar aus $(Z \setminus F) \times A$ einen **Zustandsübergang** aus $A \times Z \times \{L, R\}$ zuordnet.
  - $z_0 \in Z$ heißt Startzustand.

  Dabei ist folgendes Bild vorzustellen: Z ist die Menge aller Zustände einer Maschine. Diese Maschine besitzt einen Schreib-Lesekopf mit dem sie das Alphabet A von einem Band lesen oder darauf schreiben kann. Der Schreib-Lesekopf kann jeweils um eine Einheit nach links oder rechts auf dem Band bewegt werden, ohne dabei jemals an ein Bandende zu stoßen. Anfänglich befindet sich die Maschine im Zustand z_0, und das Band rechts vom Schreib-Lesekopf ist mit einem endlichen Inputwort beschrieben. Alle Einheiten links und rechts davon sind mit $*$ markiert. Die Maschine durchläuft nun eine Reihe von durch das jeweils gelesene Zeichen und den Ausgangszustand bedingten Zustandsänderungen. Erreicht sie einen Endzustand, so hält sie an. Dann gilt der Inhalt des Bandes als ihr Output.

- In ganz natürlicher und eindeutiger Weise kann eine Einbettung $\mathcal{E}_{turing}$ daraus eine KG (N, T, D, n, g) erzeugen:

  Als Attributraum D wird die Menge der ganzen Zahlen gewählt, und damit die Position auf dem Band kodiert. Anfangs befindet sich auf dem Band rechts von der Position Null ein endliches Inputwort aus A^*, und die Maschine steht auf Position Null im Zustand $z_0 \in Z$. Das kann dargestellt werden durch die Terminale $T = \{\neg, z_0\} \times A$, indem $\mathcal{E}_{turing}$ dem Bandinhalt

  $$\ldots, *, *, a_0, \ \ldots, a_n, *, *, \ \ldots$$

  die folgende Menge T_{given} von terminalen Instanzen zuordnet:

  $$\left\{ \begin{matrix} \ldots & (\neg, *) & , & (z_0, a_0) & , & (\neg, a_1) & , \ \ldots, & (\neg, a_n) & , & (\neg, *) & , \ \ldots \\ & -1 & & 0 & & 1 & & n & & n+1 & \end{matrix} \right\}.$$

Das Symbol $\neg$ steht dabei dafür, daß die Maschine an dieser Stelle nicht steht.

$N = (\{\neg, g\} \cup Z) \times A$ ist die von $\mathcal{E}_{turing}$ konstruierte Menge der Nichtterminale. Sie müssen extra als solche markiert werden, weil die Definition der KG $T \cap N = \emptyset$ verlangt. Jedem Zustandsübergang der Form

$$t : (z, a) \longmapsto (z', a', R)$$

ordnet $\mathcal{E}_{turing}$ die folgenden Produktionen für alle $b \in A$ zu:

$$p_b : (\ (z, a),\ (\neg, b)\ ) \quad \begin{matrix} suc \\ \longrightarrow \\ copy \end{matrix} \quad (\ (\neg, a'),\ (z', b)\ )$$

Dabei müssen in der Ausgangskonfiguration sowohl die terminalen als auch die nichtterminalen Versionen der Symbole (z, a) und $(\neg, b)$ zugelassen werden. Das Prädikat *suc* verlangt, daß der Attributwert der ersten Instanz unmittelbarer Vorgänger des Attributwerts der zweiten Instanz ist. *copy* ist die Identität. Analog verfährt man mit den Zustandsübergängen mit einem L. Allerdings wird hier das Prädikat *prd* verwendet, das verlangt, daß der Attributwert der ersten Instanz unmittelbarer Nachfolger des Attributwerts der zweiten Instanz ist.

Um das 'Halten' der Turingmaschine darzustellen, konstruiert $\mathcal{E}_{turing}$ zu jedem Haltezustand $z \in F$ für alle $b \in A$ die folgenden Produktionen:

$$p_b : (\ (z, b)\ ) \quad \begin{matrix} triv \\ \longrightarrow \\ copy \end{matrix} \quad (\ (g, b)\ )$$

Dabei macht *triv* keinerlei Einschränkungen.

Schwieriger ist die Einbettung von Stringgrammatiken, da sich das Problem der Umnumerierung des einer Ersetzung nachfolgenden Teilwortes stellt. Der hier skizzierte Weg folgt [ROSE-71, ROSE-79].

- Eine **Stringgrammatik** sei gegeben durch ein Quadrupel (N, T, P, g), wobei N, T und $g \in N$ die übliche Bedeutung haben. Die Produktionen aus P führen ein Wort κ in ein Wort λ über. Enthält ein Wort κ als Teilwort, so kann κ darin durch λ ersetzt werden. Dadurch verschieben sich im allgemeinen alle nachfolgenden Zeichen. Die Menge aller Worte, die so aus dem Wort g erzeugt werden können und nur aus Terminalen bestehen, ist die zugehörige Sprache.

- Eine **Grammatik mit Leerzeichen** enthält ein spezielles Nichtterminal $*$, das auch Leerzeichen oder Blank genannt wird. Der Produktionsprozeß geht nicht von dem endlichen Wort g aus, sondern von dem unendlichen String $*^\infty g *^\infty$. Als Sprache gehört dazu die Menge aller generierbaren Strings der Form $*^\infty \lambda *^\infty$, wobei λ nur Terminale enthalten darf. Eine natürliche, strukturerhaltende Einbettung ist dadurch gegeben, daß man $*$ zu den Nichtterminalen hinzufügt und $*^\infty$ hinten und vorne an jedes Wort anhängt.

- Soll die Einbettung auch in der anderen Richtung funktionieren, so muß sichergestellt werden, daß der Zusammenhang bei allen Produktionen erhalten bleibt. Wenn also κ links mit einem Nichtleerzeichen anfängt, so verlangt man das auch von λ, und analog am rechten Ende. Weiter verlangt man, daß beide Worte Nichtleerzeichen enthalten, die nicht durch Leerzeichen getrennt sind. In diesem Falle heißt eine Grammatik mit Leerzeichen **zusammenhangserhaltend**.

- Man führt ein weiteres Nichtterminal $\natural$ ein - es sei hier einmal als 'dummy' bezeichnet - und fügt für jedes Nichtleerzeichen x die folgenden drei Produktionen hinzu: $x* \mapsto x\natural$, um am rechten Rand der Nichtleerzeichen dummies zu erzeugen, $x\natural \mapsto \natural x$, um die dummies nach links durchzureichen und $*\natural x \mapsto * * x$, um am linken Rand der Nichtleerzeichen dummies in Leerzeichen zu verwandeln. Wenn man jetzt in jenen Produktionen, deren linke Seite kürzer ist als die rechte, an der linken Seite rechts solange dummies ansetzt, bis sie isometrisch werden, und analog mit denen verfährt, deren rechte Seite kürzer ist, hat man eine Einbettung der zusammenhangserhaltenden Grammatiken mit Leerzeichen in die isometrischen zusammenhangserhaltenden Grammatiken mit Leerzeichen und **mit dummy**. Man braucht die Umnumerierung nicht mehr. Dafür hat man ein Nichtleerzeichen und pro Nichtleerzeichen drei kontextsensitive Produktionen mehr. Es ändert sich aber weder die Klasse der Sprachen, die man so definiert, noch die Komplexitätsklasse der zugehörigen Parser. Es wird dadurch lediglich in der Grammatik explizit, was man beim Parsen einer klassischen Stringgrammatik in einem dynamischen Speicher sowieso machen müßte: Nämlich entsprechende Platzreservierungen, Platzfreigaben und Shiftoperationen.

- Für die Richtung des Generierens kann nun eine Einbettung $\mathcal{E}$ der isometrischen Grammatiken mit Leerzeichen in die Koordinaten Grammatiken vorgenommen werden: Sämtliche Symbole einschließlich des Startsymbols, der Terminale und des symbolischen Teils der Produktionen werden direkt übernommen.

$$N' = \mathcal{E}(N),\ T' = \mathcal{E}(T),\ g' = \mathcal{E}(g)$$

$$\forall p \in P\ \Sigma = \mathcal{E}(\kappa)\ \wedge\ \Lambda = \mathcal{E}(\lambda)$$

Es wird als einziges Attribut D die Menge der ganzen Zahlen eingeführt. Dieses Attribut steht unter dieser Einbettung für die Position der Symbole in der Symbolkette. Entsprechend wird für alle Produktionen $p \in P$ durch $\mathcal{E}$ aus der Konkatenation der Symbole auf der 1inken Seite das Prädikat

$$\pi = true \iff d_2 = d_1 + 1 \ \wedge ... \wedge \ d_{|\kappa|} = d_{|\kappa|-1} + 1$$

und aus der positionstreuen isometrische Ersetzung die Funktion

$$\phi_1 = d_1, \ ... \ \phi_{|\lambda|} = d_{|\lambda|}.$$

Damit ist auch $\mathcal{E} : p \mapsto p' = (p, \pi, \phi)$ definiert.

- Der zugehörige Sprachbegriff unterscheidet sich in den zugelassenen Ausgangsmengen von den bisher verwendeteten Sprachen zu einer KG $\mathcal{L}_{pure}$ und $\mathcal{L}_{noisy}$. Man setzt vielmehr für ganze Zahlen k statt $S_0(k) = \{(g, k)\}$

$$S_0(k) = \{..., (*, k-2), (*, k-1), (g, k), (*, k+1), (*, k+2), ...\}$$

und erhält den durch $\mathcal{E}$ induzierten Sprachbegriff

$$\mathcal{L}_{\mathcal{E}} \stackrel{def}{=} \{S \subseteq \mathcal{T} \cup \{I; s(I) = *\}; \quad \exists k \in Z \ : \quad S \stackrel{*}{\longrightarrow} S_0(k)\},$$

der in natürlicher Weise zu $\mathcal{L}_{pure}$ isomorph ist.

In recht natürlicher Weise lassen sich die in [GONZ] angeführten Netz Grammatiken (engl. *web grammars*) in die KGs einbetten:

- Eine **Netz Grammatik** sei gegeben durch ein Quadrupel (N, T, P, g), wobei N, T und $g \in N$ die übliche Bedeutung haben. Die Produktionen p aus P arbeiten auf ungerichteten Graphen, deren Knoten mit Symbolen aus $V = N \cup T$ belegt sind. Es gibt zu einem solchen Graphen G mithin eine Knotenmenge G_N, eine Kantenmenge $G_E \subseteq \{\{N_1, N_2\} \subseteq G_N\}$ und eine Belegung der Knoten mit Symbolen $b : G_N \to V$. Die Produktionen ersetzen einen Teilgraphen κ von G durch einen anderen Teilgraphen λ. Es muß also eine Untergraph Homomorphie $h : \kappa \to G$ vorliegen, die surjektiv ist, und außer der Graphstruktur auch die Belegung der Knoten homomorph transportiert. Bei der Ersetzung ist zu beachten, daß man mit den Knoten von κ natürlich auch alle Kanten aus G_E entfernen muß, die einen dieser Knoten enthalten. Zu dem verbleibenden Restgraphen (host web), kann man dann λ hinzufügen. Üblicherweise wird man λ mit dem Restgraphen verbinden wollen. Dafür muß für jede Produktion eine *Einfügungsbedingung* f definiert werden. Damit der 'syntaktische Charakter' des Verfahrens erhalten bleibt, sollte f nicht vom Restgraph abhängen, sondern nur von κ und λ und den Symbolen, die in der Grammatik definiert sind. Motiviert

durch Anwendungen wird in [GONZ] die Konstruktion einer Kante zwischen einem Knoten N_1 in λ und einem Knoten N_2 im Restgraphen genau dann vorgenommen, wenn es in κ einen Knoten N_3 gab, der mit N_2 verbunden war ($\{N_2, N_3\} \in G_E$) und wenn dieser Knoten N_2 mit einem Symbol belegt ist, das in $f(N_1, N_3)$ enthalten ist. Die Einfügungsbedingung ist mithin eine Funktion von der Produktmenge der Knoten aus κ und λ in die Potenzmenge von V.

- KGs arbeiten auf unstrukturierten Mengen. Eine natürliche Form der Einbettung ist also die, daß zu einer Netz Grammatik mit dem Alphabeth V eine KG mit dem Alphabeth $V \cup \{\{v, w\};\ v, w \in V\}$ gehört. Das macht explizit, daß es nicht nur Knoten, sondern auch ungerichtete Kanten gibt.

- Die Knoten eines Netzes müssen unterscheidbar sein. In der Einbettung kann man dies nur über ein entsprechendes, nicht begrenztes Index Attribut d_1 erreichen. In natürlicher Weise kann man dann die Kanten Instanzen doppelt indizieren, mit einem zweidimensionalen Attribut d_2.

- Weil man nicht wissen kann, wieviele Kanten eine Netz Grammatik Produktion in einem konkreten Netz ersetzt, kann man sie nicht direkt als eine einzelne KG Produktion einbetten. Man muß die Einfügungsbedingungen an den einzelnen Knoten darstellen durch KG Produktionen, die Zwischenergebnisse liefern. Auf diesen Zwischenergebnissen dürfen keine anderen Produktionen arbeiten. Man muß das Netz lokal sperren. Das geschieht über ein weiteres Attribut: $d_3 = 0$ solange keine Produktion auf diesem Knoten arbeitet.

- Aus einer Netz Grammatik Produktion $p = (\kappa, \lambda, f)$ werden in der Einbettung mehrere KG Produktionen. Die erste Produktion p_{init} hat als linke Seite die Menge der Knoten und Kanten in κ als Tupel dargestellt. Bedingungsprädikat ist $d_3 = 0$ auf der gesamten Ausgangskonfiguration und die Existenz der Untergraph-Homomorphie h. Als rechte Seite dient eine Darstellung *beider* Seiten von p. d_3 wird dabei auf den Index der Produktion gesetzt. Zur Durchführung der Produktion p muß ein h ausgewählt werden, dieses wird ebenfalls gespeichert, wozu ein weiteres Attribut d_4 erforderlich ist. λ kann zunächst nicht mit dem Restgraphen verbunden werden.

- Dazu wird für jedes Tripel (N_1, N_3, v), für das N_1 ein Knoten in λ, N_3 ein Knoten in κ und v in $f(N_1, N_3)$ ist, eine Produktion definiert, die eine Kante von N_3 zu einem Knoten, der mit v gelabelt ist, ersetzt durch eine Kante von N_1 zu diesem Knoten. Bedingung ist, daß d_3 auf dem Index der richtigen Produktion steht, und d_4 auf N_3 zeigt. In der Zielkonfiguration kann die Kante wieder freigegeben werden, indem man $d_3 = 0$ setzt. Wie oft die Produktion durchlaufen werden muß, hängt davon ab, wieviele Kanten von

dem durch h als N_3 gewählten Knoten zu einem mit v gelabelten Knoten des Restgraphen existieren.

- Um κ zu löschen, benötigt man eine Produktion mit globaler Konsistenzbedingung, die hier mit p_{finit} bezeichnet werden soll. Diese kann erst in Aktion treten, wenn alle Verbindungen von κ mit dem Restgraphen ersetzt worden sind durch Verbindungen mit λ. Der eingesetzte Teil kann nun freigegeben werden mit $d_3 = 0$.

Für die Entscheidung des aussagenlogischen Erfüllbarkeitsproblems gibt es eine RBB-KG.

- Dabei wird von folgender konjunktiver Normalform ausgegangen:

$$(C_{1,1} \vee ... \vee C_{1,n_1}) \wedge ... \wedge (C_{k,1} \vee ... \vee C_{k,n_k})$$

  Bekanntlich läßt sich in polynomialer Zeit zu jeder aussagenlogischen Formel eine Formel in konjunktiver Normalform bestimmen, die bzgl. Erfüllbarkeit äquivalent ist. Man kommt dabei sogar mit maximal drei Literalen pro Klausel aus [SCHÖ]. Ein Literal $C_{i,j}$ hat die Gestalt X_h oder $\neg X_h$ für eine Variablennummer $h \geq 0$. Da Normalform vorausgesetzt wird, tragen die Klammern keine Information, und können entfernt werden unter der Konvention '$\vee$ geht vor $\wedge$'. Am Anfang und am Ende wird eine Marke $*$ gesetzt. Danach stehen an ungeraden Positionen der Formel die logischen Zeichen $\vee, \wedge$ oder eben ein $*$ und an geraden Positionen Literale.

- Als terminale Symbole werden für die KG definiert $\{\vee, \wedge, *, lit\}$. Attributraum ist $\mathbb{N}^3$. Die erste Komponente p_a steht für die Anfangsposition. Die zweite Komponente p_e steht für die Endposition. Die dritte Komponente Vb steht für eine Variablenbelegung. Dazu wird sie in ternärer Notation geschrieben, und die Funktion $tern_h$ bestimmt die h-te Stelle in der Ternärdarstellung einer Zahl.

$$\begin{aligned} tern_h(Vb) &= 0 \iff X_h \text{ ist nicht belegt} \\ tern_h(Vb) &= 1 \iff X_h \text{ ist wahr} \\ tern_h(Vb) &= 2 \iff X_h \text{ ist falsch} \end{aligned}$$

- Die Einbettung $\mathcal{E}$ ordnet jedem Literal $C_{i,j}$ die entsprechende terminale Instanz des Symbols lit zu. Die Position in der Formel p wird als Anfangs- und Endposition gesetzt. Die Variablenbelegung Vb ist mit $1 \cdot 3^h$ wenn $C_{i,j} = X_h$ und $2 \cdot 3^h$ wenn $C_{i,j} = \neg X_h$ so gewählt, daß sie dieses Literal erfüllt. Die Einbettung der Zeichen $\vee, \wedge$ und $*$ ist trivial. Die Formel

$$(X_4 \vee \neg X_3) \wedge (X_0 \vee \neg X_2 \vee \neg X_4)$$

wird z. B. zur folgenden Terminalmenge

$$\left\{\begin{array}{cccccccccccc} * & lit & \vee & lit & \wedge & lit & \vee & lit & \vee & lit & * \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \\ & 10000 & & 2000 & & 1 & & 200 & & 20000 & \end{array}\right\}$$

Diese einfache Kodierung ist sicher in polynomialer Zeit machbar.

- Nichtterminale Symbole sind $\{dis, con, erf\}$, wobei erf als Startsymbol dient. Die folgenden beiden trivialen Produktionen werden definiert, weil jedes Literal für sich schon eine Disjunktion und eine Konjunktion ist:

$$p_1 : (lit) \quad \overset{\substack{true \\ copy}}{\longrightarrow} \quad (dis)$$

$$p_2 : (dis) \quad \overset{\substack{true \\ copy}}{\longrightarrow} \quad (con)$$

- Die Semantik der Disjunktion wird durch die folgenden beiden Produktionen erfaßt:

$$\begin{array}{rcl} suc(p_{e,1}) & = & p_{a,2} \\ \wedge\ suc(p_{e,2}) & = & p_{a,3} \\ p_3 :\ (lit, \vee, dis) & \longrightarrow & (dis) \\ p_a & = & p_{a,1} \\ p_e & = & p_{e,3} \\ Vb & = & Vb_1 \end{array}$$

$$\begin{array}{rcl} suc(p_{e,1}) & = & p_{a,2} \\ \wedge\ suc(p_{e,2}) & = & p_{a,3} \\ p_4 :\ (lit, \vee, dis) & \longrightarrow & (dis) \\ p_a & = & p_{a,1} \\ p_e & = & p_{e,3} \\ Vb & = & Vb_3 \end{array}$$

Die Gestalt der Produktionen ist sehr ähnlich. *suc* bezeichnet hier die Nachfolgerfunktion, so daß im Bedingungsprädikat jeweils verlangt wird, daß die Instanzen des Tripels unmittelbar hintereinander stehen müssen. Die Produktionen ordnen der neuen Instanz des Symbols *dis* die Anfangsposition des Literals als Anfangsposition zu und die Endposition der Disjunktion als Endposition. Der einzige Unterschied liegt in der Variablenbelegung. Die neue Disjunktion wird erfüllt, wenn die Variablenbelegung vom Literal übernommen wird *oder* wenn die Variablenbelegung von der alten Disjunktion übernommen wird. Diese beiden Produktionen bringen also genau den

entscheidenden Nichtdeterminismus ins Spiel. Z. B. sind beide folgenden Reduktionen möglich:

$$\left\{\begin{array}{ccc} lit & \vee & dis \\ 2 & 3 & 4 \\ 2 & 3 & 4 \\ 10000 & & 2000 \end{array}\right\} \xrightarrow{p_3} \left\{\begin{array}{c} dis \\ 2 \\ 4 \\ 10000 \end{array}\right\} \text{ und } \left\{\begin{array}{ccc} lit & \vee & dis \\ 2 & 3 & 4 \\ 2 & 3 & 4 \\ 10000 & & 2000 \end{array}\right\} \xrightarrow{p_4} \left\{\begin{array}{c} dis \\ 2 \\ 4 \\ 2000 \end{array}\right\}$$

- Die Semantik der Konjunktion wird durch die folgenden beiden Produktionen erfaßt:

$$\begin{array}{rcl} suc(p_{e,1}) & = & p_{a,2} \\ \wedge\ suc(p_{e,2}) & = & p_{a,3} \\ \wedge\ tern_h(Vb_1) \neq 0 & \Rightarrow & tern_h(Vb_3) = 0 \\ p_5:\ (dis, \wedge, con) & \longrightarrow & (con) \\ p_a & = & p_{a,1} \\ p_e & = & p_{e,3} \\ Vb & = & Vb_1 + Vb_3 \end{array}$$

$$\begin{array}{rcl} suc(p_{e,1}) & = & p_{a,2} \\ \wedge\ suc(p_{e,2}) & = & p_{a,3} \\ \wedge\ tern_h(Vb_1) \neq 0 & \Rightarrow & tern_h(Vb_3) = tern_h(Vb_1) \\ p_6:\ (dis, \wedge, con) & \longrightarrow & (con) \\ p_a & = & p_{a,1} \\ p_e & = & p_{e,3} \\ Vb & = & Vb_3 \end{array}$$

Was die Positionen angeht gibt es keinen Unterschied zu p_3 und p_4. Wie oben erläutert, bestimmt die Funktion $tern_h$ die h-te Stelle in der Ternärdarstellung einer Zahl. Für das Attribut Vb von Instanzen des Symbols dis gibt es nur eine Stelle, für die $tern_h(Vb) \neq 0$ gilt. Die Bedingung ist also leicht nachprüfbar. Entweder die entsprechende h-te Variable ist in der Instanz des Symbols con noch nicht belegt, dann übernimmt man die Belegung von der Disjunktion indem man die Belegungsattribute addiert (Produktion p_5), oder die Belegungen widersprechen einander nicht, weil die h-te Variable in der Konjunktion genauso belegt ist wie in der Disjunktion, dann kann die Belegung der Konjunktion übernommen werden (Produktion p_6). Im Beispiel ergäbe sich unter anderen auch die folgende Möglichkeit:

$$\left\{\begin{array}{ccc} dis & \wedge & con \\ 2 & 5 & 6 \\ 4 & 5 & 10 \\ 10000 & & 200 \end{array}\right\} \xrightarrow{p_5} \left\{\begin{array}{c} con \\ 2 \\ 10 \\ 10200 \end{array}\right\}$$

- Die letzte Produktion erzeugt eine Instanz des Starsymbols erf aus einer Konjunktion, wenn die Endmarken erreicht sind:

$$\begin{array}{rcl} suc(p_{e,1}) & = & p_{a,2} \\ \wedge\ suc(p_{e,2}) & = & p_{a,3} \\ p_7 :\ (*, con, *) & \longrightarrow & (erf) \\ p_a & = & p_{a,1} \\ p_e & = & p_{e,3} \\ Vb & = & Vb_2 \end{array}$$

  Genau dann wenn eine solche Instanz reduzierbar ist, ist die zugehörige Formel erfüllbar. Man kann sogar eine mögliche Variablenbelegung aus dem Attributwert ablesen.

Damit ist eine polynomiale Reduktion des Erfüllbarkeitsproblems auf das Wortproblem einer RBB-KG konstruiert. Das Problem, ob eine Menge terminaler Instanzen in der zugehörigen Sprache ist, ist mithin NP-hart. Da die Nachprüfung einer Reduktion in polynomialer Zeit erfolgen kann, ist es NP-vollständig [SCHÖ].

In der Abbildung 2.4 sind die Zusammenhänge zusammengefaßt. Bzgl. der Beziehungen zwischen der Chomsky Hierarchie (uneingeschränkte String Grammatik > kontextsensitiv > kontextfrei > regulär > nicht rekursiv) und der Automatentheorie (Turing Maschine > linear begrenzte Maschine > Keller Automat > endlicher Automat > table look up) sei auf die Literatur verwiesen (z. B. [AHO, SCHÖ]). Einige Bemerkungen finden sich auch im Abschnitt 7.3.1 dieser Arbeit. Für die nicht rekursiven KGs läßt sich, wie im Abschnitt 5.1.2 dargelegt eine polynomiale Schranke für die Rechenzeit angeben. Sie gehören also in etwa auf eine Stufe mit den kontextfreien Grammatiken.

Es sind nur die wesentlichen Einbettungen dargestellt. Es gibt z. B. natürlich für die Koordinaten Grammatiken mit globaler Konsistenzbedingung - wie für jedes andere algorithmische Verfahren auch - eine Einbettung in die Turingmaschinen. Diese ist jedoch mit unübersichtlichen Kodierungen verbunden, welche die gemeinte Struktur eher verschleiern. Es wird in dieser Arbeit davon ausgegangen, daß die Sprachen, die von einer sinnvollen Grammatik erzeugt werden, eine Bedeutung haben. Und diese kann in der KG sehr wohl deutlich hervortreten, während sie in einer die KG simulierenden Turing Maschine sicherlich schwer zu erkennen sein wird. Ein anderes Beispiel liegt in der Tatsache, daß endlich attributierte KGs natürlich nur eine endliche Sprache generieren können, und folglich durch einen Table Look Up ersetzt werden können. In der Mustererkennung ist diese Tatsache gänzlich ohne Belang, da dieses 'Lexikon' schon bei recht kleinen Attributräumen astronomische Ausmaße annimmt. Alle Struktur und Semantik wäre durch solche Aufzählung erschlagen. Das Diagramm 2.4 führt nur solche Einbettungen auf, die wirklich Struktur transportieren.

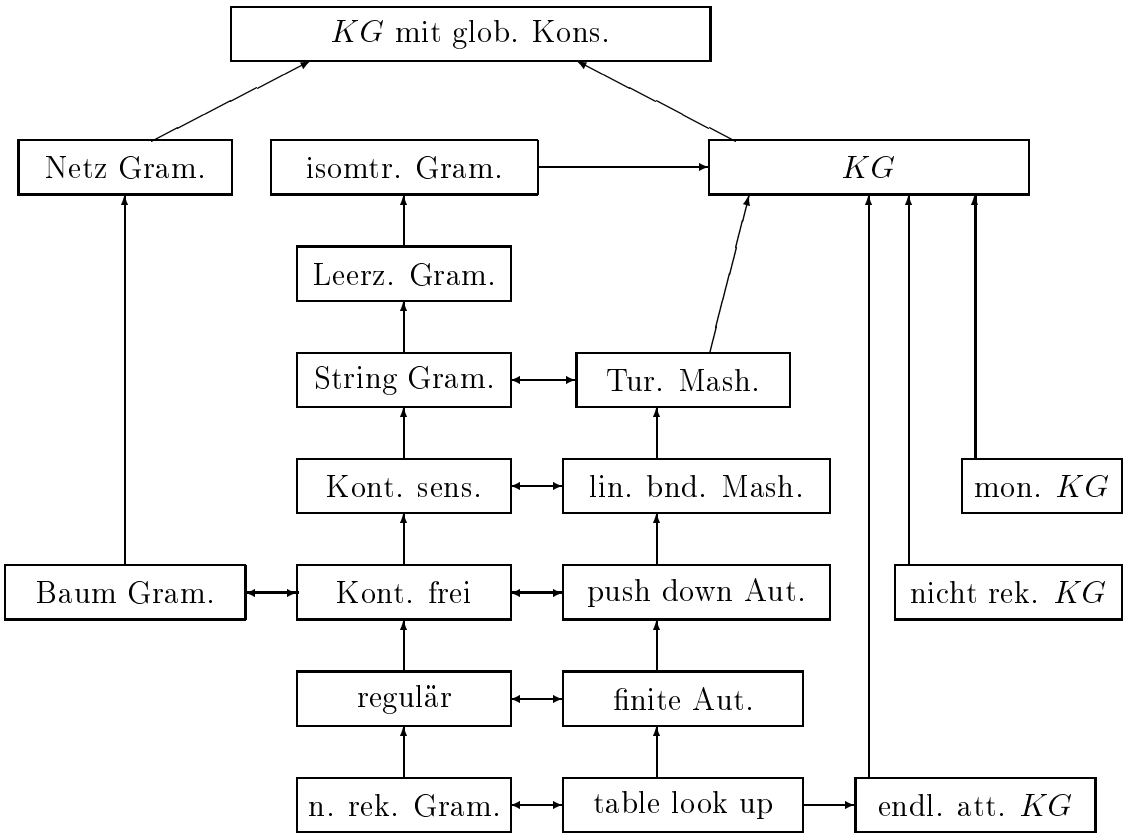


Figure 2.4: **Hierarchie der Grammatiken**
Die Pfeile weisen jeweils in Richtung einer Einbettung;
Das Diagramm dient der Anschauung und stellt
keinen Anspruch auf Vollständigkeit.

Chapter 3

Algorithmen

Im Kapitel 2 wurde vorgeschlagen, für die (nach [MARR]) erste Ebene der Probleme des Computersehens Koordinaten Grammatiken zu verwenden, weil sie die Möglichkeit bieten, komplexe logische Strukturen zusammen mit geometrischen Constraints zu betrachten. Dieses Kapitel beschäftigt sich mit der nach Marr zweiten Problemebene, der Repräsentation der Daten und den Algorithmen, die die theoretische Struktur der KG zuläßt bzw. nahelegt. Insbesondere die algorithmische Komplexität bzgl. der Kardinalität der Eingangsdaten wird diskutiert. Zur Definition der Algorithmen wurde speziell eine Sprache konstruiert. Die BNF Syntaxdefinition dazu findet sich im Anhang E. Dort ist auch die Semantik erläutert. Die Sprache unterstützt besonders das Bilden und Aufzählen von Mengen. Für die Bestimmung der algorithmischen Komplexität werden die Prozeduraufrufe, Vergleichsoperationen, Produktionsdurchführungen usw. dieser Sprache als elementar gesetzt. Soll die Komplexität für eine gegebene KG z. B. in Turingmaschinenschritten ausgedrückt werden, so wären noch entsprechende Umrechnungen nötig. Dabei würde die Komplexität der in den Produktionen durchzuprüfenden Prädikate und der darin enthaltenen Funktionen eingehen. Auch wäre zu berücksichtigen, daß der Vergleich zweier Mengen von Instanzen auf Identität zwar von polynomialer Komplexität ist, aber für Turingmaschinen mit einigem Aufwand verbunden.

Durch die Algorithmen ist das Wortproblem $T_{given} \stackrel{?}{\in} \mathcal{L}_{pure}$ oder $T_{given} \stackrel{?}{\in} \mathcal{L}_{noisy}$ für eine gegebene Menge von terminalen Instanzen T_{given} und eine gegebene KG zu entscheiden. Die im Abschnitt 3.1 vorgestellte *Top Down* Methode mit *Back Tracking* liefert hierfür bezüglich des Rechenaufwands völlig inakzeptale Werte. Dies liegt natürlich auch daran, daß die Produktionen der KG, wie im Abschnitt 2.1.1 definiert, schon eher auf die Reduktionsrichtung hin konzipiert sind. Die Generierung ist ein anderes Problem. Das bedeutet, daß die eigentliche 'straight forward' Methode für die algorithmische Behandlung der KGs *Bottom Up* läuft. Da es sich um ersetzende Produktionssysteme handelt, untersucht der Abschnitt

3.2 zunächst wiederum die einfachste serielle Form der Suche, das ist Tiefensuche mit *Back Tracking*. Solche rekursiven Verfahren sind in der Praxis vom Aufwand her oft suboptimal und bei der Fehlersuche und Wartung häufig unübersichtlich. Man kann sie näherungsweise ersetzen durch nicht rekursive *kumulative* Verfahren analog zu den bekannten Tabellenparsern für String Grammatiken. Der Abschnitt 3.3 stellt diese Vereinfachung vor, und zeigt auf, wie dadurch praktische Parallelisierbarkeit erreicht werden kann. Bevor dann im nächsten Kapitel auf die Implementierung und Hardware eingegangen werden kann, wird im Abschnitt 3.4 eine wichtige Möglichkeit vorgestellt, erheblich Rechenzeit zu sparen, durch assoziativ nach Attributwerten adressierbares Abspeichern der Instanzen.

3.1 Top Down mit Back Tracking

Verfahren, wie das durch Algorithmus 1 definierte Top Down Parsing mit Back Tracking, werden in der syntaxorientierten Informatik nicht wirklich in der Praxis durchgeführt, weil sie annähernd auf ein vollständiges, serielles *depth first* Durchsuchen des Sprachraumes einer Grammatik hinauslaufen. Man setzt aber syntaktische Verfahren ja genau da ein, wo dieser 'Raum der Möglichkeiten' so groß ist, daß er nicht aufgelistet werden kann. Eine Grammatik kann rekursive Strukturen enthalten, die eine 'depth first' Suche in eine Endlosschleife führen. Bei Stringgrammatiken ist das der Fall, wenn es links-rekursive Produktionen gibt. Bei KGs kann jede mögliche Rekursion einen solchen Algorithmus in eine Endlosschleife führen. Aber auch wenn rekursive Strukturen vollkommen fehlen und der Attributraum endlich ist, führt dieses Vorgehen in der Regel auf eine astronomische Zahl von Möglichkeiten. Dieser Algorithmus wird also nicht angegeben, um ihn in der Praxis durchzuführen, sondern weil er die kombinatorische Struktur des Sprachraums verdeutlicht. Zunächst soll dies für die Sprache $\mathcal{L}_{pure}$ einer endlich attributierten KG geschehen. Gegeben sind das Universum $\mathcal{U}$ (also das Alphabet und der Attributbereich wie in Abschnitt 2.1 definiert) und eine Menge von Produktionen P. Ein Nichtterminal g ist als Startsymbol ausgezeichnet. Zu untersuchen ist eine Menge von terminalen Instanzen $T_{given} \subseteq \mathcal{U}$. Ergebnis ist ein Signal, ob die Suche erfolgreich war oder nicht.

Wie stark sich der Rekursionsbaum, der hier ausgespannt wird, verzweigt, hängt hauptsächlich von der Kardinalität der Menge $\{S' \subseteq \mathcal{U};\ p : S' \longrightarrow S\}$ für gegebene S und p ab. Ist die Funktion ϕ, die die Attributwerte der Zielkonfiguration von p bestimmt, injektiv, so gibt es nur eine Ausgangskonfiguration für jede Zielkonfiguration. Anhand des Beispiels in Abschnitt 2.1.3 wird aber klar, daß diese Funktionen im allgemeinen nicht injektiv sind, sondern für jeden Zielwert recht umfangreiche Urbilder haben, deren Größe von den Schwellwerten im Bedingungsprädikat π abhängt. Abbildung 3.1a) zeigt die Lage der einelementigen Zielkonfiguration. Dadurch ist zwar die Lage des Ortsattributs P_2 für beide

ALGORITHM $Top\_Down\_Backtrack($ $\mathcal{U}$: $Universe$;
P : $Set\ of\ Production$;
g : $Nonterminal$;
T_{given} : $Set\ of\ Instance$;
$VAR\ \ signal$: $\{failure, success\}$);

$VAR\ \ I\ :\ Instance$;

PROCEDURE $Search\_Deeper($ $S\ :\ Set\ of\ Instance$;);

$VAR\ S'\ :\ Set\ of\ Instance$;
$p\ :\ Production$;

$Begin$
$\forall p \in P\ do\ \forall S' \in \{S' \subseteq \mathcal{U};\ p : S' \longrightarrow S\}\ do$
$If S' = T_{given}\ Then$
$signal := success$;
$stop$;
$Else$
$Search\_Deeper(\ S'\ )$;
$EndIf$;
$end\ \forall S'$; $end\ \forall p$;
$end\ Search\_Deeper$;

$Begin\ main$
$\forall I \in \{I \in \mathcal{U};\ s(I) = g\}\ do$
$Search\_Deeper(\{I\})$;
$end\ \forall I$;
$signal := failure$;
$stop$;
$end.\ main$

Algorithmus 1: Top Down Parser

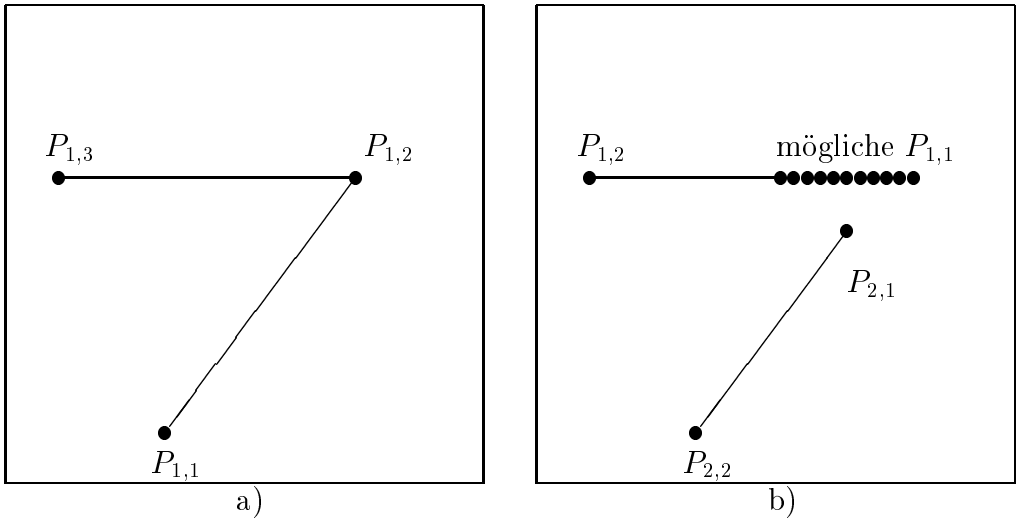


Figure 3.1: **Urbild einer Zielkonfiguration**
a) Menge S mit einer Zielkonfiguration bestehend aus
einer Instanz des Symbols *Winkel*
b) Ein Teil der Mengen S', aus denen S reduziert werden kann,
$P_{2,1}$ wurde festgehalten

Instanzen möglicher Ausgangskonfigurationen festgelegt, aber für das Attribut P_1 gibt es jeweils viele Möglichkeiten. Setzt man für das Nachbarschaftsprädikat z. B. $d_{max} = 50$ und wählt man $P_{2,1}$ auf der Geraden von $P_{2,2}$ zum Schnittpunkt, so kann $P_{1,1}$ auf der Geraden von $P_{1,2}$ zum Schnittpunkt frei gewählt werden, solange er innerhalb eines Kreises von 50 Pixel Radius um $P_{2,1}$ bleibt. Abbildung 3.1b) zeigt einige solche Möglichkeiten. Bei fester Lage von $P_{2,1}$ entspricht die Anzahl der Möglichkeiten für $P_{1,1}$ mithin der Länge der entsprechenden Sehne. Insgesamt ergeben sich zu der Zielkonfiguration aus Abbildung 3.1a) $50^2\pi \approx 7800$ Augangskonfigurationen S'.

Man kann solche Berechnungen und Abschätzungen nicht nur für einzelne Konfigurationen durchführen, sondern auch für ganze KGs. Im folgenden wird dies an der Beispiel KG aus dem Abschnitt 2.2.3 demonstriert: Das Universum hat hier die Kardinalität

$$|\mathcal{U}| \;=\; |V| \cdot |D| \;=\; 4 \cdot 180^{(4 \cdot 2+1)},$$

und die Sprache der KG ist eine Teilmenge der Potenzmenge dieses Universums $2^{\mathcal{U}}$. Für Algorithmus 1 wird der maximale Rechenaufwand mit einer Menge T_{given} erreicht, die nicht in der Sprache enthalten ist, z. B. mit $T_{given} = \emptyset$. Für solche Mengen wird das Abbruchkriterium $S' = T_{given}$ nie erfüllt. Da das die einzige

Stelle ist, wo T_{given} überhaupt eingeht, ist für alle diese Mengen der Aufwand gleich und maximal.

Zeile zwei des Hauptteils in Algorithmus 1 zählt alle möglichen Zielinstanzen und damit den gesamten Attributbereich D auf. Es wird jeweils eine einelementige Menge $\{I\}$ gebildet und damit die Prozedur *Search_Deeper* aufgerufen. Nur die Produktion p_3 kann solche Mengen reduzieren. Die Kardinalität der Menge $\{S' \subseteq \mathcal{U};\ S' \xrightarrow{p_3} \{I\}\}$ ist dann abhängig von dem jeweiligen I, läßt sich aber wie oben abschätzen aus dem Schwellwert des Prädikates *adj* als $d_{max}^2\pi$. Für Instanzen am Bildrand ist die Kardinalität kleiner. Wegen der zusätzlich eingehenden Prädikate *apa* und *spin* liegt der tatsächliche Wert oft auch bei Null. $d_{max}^2\pi$ taugt nur als obere Grenze. Mit den nun generierten zweielementigen Mengen ruft sich die Prozedur dann wieder jeweils selbst auf. Eine der beiden Instanzen ist ein Terminal, kann also nicht reduziert werden. Die andere hat den symbolischen Teil $Winkel$. Sie kann durch p_2 reduziert werden. Es gelten dieselben Überlegungen wie oben. Die so entstehenden dreielementigen Mengen enthalten zwei Terminale und eine Instanz des Symbols L_Line. Nur die Produktion p_1 kann solche Mengen reduzieren. Die Kardinalität der entsprechenden Menge von Ausgangskonfigurationen läßt sich mit ähnlichen Überlegungen wie oben abschätzen. Auch hier ist jeweils ein Punkt der möglichen Instanzen der Ausgangskonfiguration durch die Zielkonfiguration festgelegt. In einem Streifen von $2 \cdot d_{max}$ Breite zwischen diesen Punkten kann der erste der beiden anderen Punkte frei gewählt werden. Die längste Strecke im Bild mit einer Länge von $\sqrt{2} \cdot 180$ Pixeln liefert hier als eine Abschätzung nach oben $2 \cdot d_{max} \cdot (\sqrt{2} \cdot 180)$ möglichen Lagen dafür. Der zweite Punkt muß im gleichen Bereich überlappend gewählt werden. Das halbiert im Mittel die Zahl der Möglichkeiten. Es ergeben sich also maximal $4 \cdot d_{max}^2 \cdot 180^2$ mögliche Ausgangskonfigurationen. Die entstehenden vierelementigen Mengen enthalten dann nur noch Terminale. Damit ergibt sich insgesamt für die Anzahl der Aufrufe der Prozedur *Search_Deeper* auf unterster Rekursionsebene eine obere Schranke von

$$180^{(4\cdot 2+1)} \ \cdot\ d_{max}^2\pi \ \cdot\ d_{max}^2\pi \ \cdot\ 4 \cdot d_{max}^2 \cdot 180^2 \approx 2.5 \cdot 10^{25} \cdot d_{max}^6.$$

Wenn T_{given} in der Sprache ist, wird der Algorithmus früher halten. Praktikabel ist so etwas sicher trotzdem nicht. Diese Zahl gibt eine Vorstellung von der Kardinalität der Sprache $\mathcal{L}_{pure}$. Relativ zur Potenzmenge $2^{\mathcal{U}}$, deren Teilmenge sie ist, ist $\mathcal{L}_{pure}$ dagegen offensichtlich sehr klein.

Ist eine Lokalisationsaufgabe gestellt, so wird das *stop* in der sechsten Zeile dieser Prozedur entfernt. Die Variable *signal* wird für jedes I mit $failure$ initialisiert. Man führt eine Menge ein, die leer initialisiert wird und in die die aktuelle Instanz I des Startsymbols eingetragen wird, wenn *signal* nach dem *Search_Deeper* auf *success* steht. Dann hält der Algorithmus erst, wenn die ganze Sprache $\mathcal{L}_{pure}$ durchsucht ist. In diesem Falle hängt der Rechenaufwand also nicht von der Eingangsmenge T_{given} ab, sondern nur von der KG.

Der Algorithmus gilt für $\mathcal{L}_{pure}$. Normalerweise wird man sich in der Mustererkennung eher für $\mathcal{L}_{noisy}$ interessieren. Dann muß lediglich in der If-Bedingung in der vierten Zeile der Prozedur $Search\_Deeper$ das $=$ durch ein $\subseteq$ ersetzt werden.

Die aus dem sehr einfachen Beispiel resultierenden großen Zahlen machen deutlich, was mit dem zweiten Prinzip 'least commitment' aus [MARR] (Seite 106) gemeint ist: Algorithmen, die Hypothesen generieren und diese dann verifizieren, sind zu vermeiden, weil sie überflüssige Arbeiten, die durch die Daten nicht gerechtfertigt sind, durchführen. Im Falle der KGs ist es nicht nötig, den ganzen Sprachraum zu durchsuchen. Es genügt den Teil zu bearbeiten, der durch die Daten gestützt wird. Dies leistet der im nächsten Abschnitt beschriebene Algorithmus.

3.2 Bottom Up mit Back Tracking

Die Definition der KG, wie in Abschnitt 2.1.1 - analog zu Anderson und Rosenfeld - durchgeführt, legt die an sich für Grammatiken nicht übliche Richtung von rechts nach links nahe. Diese reduzierende Richtung bleibt bei normalen syntaktischen Verfahren den Parsern vorbehalten. Man geht von den Terminalen in Richtung Startsymbol, ohne dabei den gesamten Sprachraum abzusuchen. Es werden nur Alternativen untersucht, die von den Attributwerten der Terminale in T_{given} aus möglich sind. Die Funktionen ϕ in den Produktionen geben jeweils die neuen Attributwerte für die Nichtterminale, und so schreitet man reduzierend voran. Dabei wird die Menge der aktuell zu betrachtenden Instanzen für monotone Grammatiken immer kleiner. Dies ist die kanonische Vorgehensweise für die Koordinaten Grammatik. Allerdings muß man alle Möglichkeiten durchprobieren, und erhält so den ebenfalls rekursiven Algorithmus 2.

Will man statt $\mathcal{L}_{pure}$ $\mathcal{L}_{noisy}$ bearbeiten, so ist $S' = \{I\}$ in der vierten Zeile der rekursiven Prozedur zu ersetzen durch $I \in S'$. Auch hier liegt ein 'depth-first' Suchprozeß vor. Das bedeutet, daß Ersetzungen, die nicht zum Erfolg führen, rückgängig gemacht werden müssen. Man darf also die Ausgangskonfigurationen nicht global löschen. Vielmehr übergibt man die ganze aktuelle Menge als Parameter in einer Rekursion. Im schlimmsten Fall führt das auf Endlosschleifen. In besonders gutmütigen Fällen ist der Rechenaufwand hingegen minimal. Das Beispiel aus dem Abschnitt 2.2.3 wird z. B. genau so durchgerechnet, wie es dort präsentiert ist. Es ergeben sich keine alternativen Möglichkeiten. Somit kommt es hier gar nicht zum Back Tracking. In der Regel wird man aber nicht vorhersagen können, wieviele Alternativen beim Reduzieren einer bestimmten Menge T_{given} mit einer gegebenen KG geprüft werden müssen. Für monotone KGs beträgt die Tiefe der Rekursion maximal $n = |T_{given}|$. Damit muß für $\frac{n(n-1)}{2}$ Instanzen Speicherplatz zur Verfügung stehen. Für die Verwaltung der Rekursion braucht

ALGORITHM $Bottom\_Up\_Backtrack(\ \mathcal{U} : Universe;$
$P : Set\ of\ Productions;$
$g : Nonterminal;$
$T_{given} : Set\ of\ Instance;$
$VAR\ signal : \{failure, success\}\ );$

PROCEDURE $Search\_Deeper(\ S : Set\ of\ Instance;\ );$

$VAR\ S' : Set\ of\ Instance;$
$p : Production;$
$I : Instance;$

$Begin$
$\forall p \in P\ do\ \forall S' \in \{S' \subseteq \mathcal{U};\ p : S \longrightarrow S'\}\ do$
$If\ S' = \{I\}\ \wedge\ s(I) = g\ Then$
$signal := success;$
$stop$
$Else$
$If\ S' \neq \emptyset\ Then\ Search\_Deeper(\ S'\ );\ EndIf;$
$EndIf;$
$end\ \forall S';\ end\ \forall p;$
$end\ Search\_Deeper;$

$Begin\ main$
$Search\_Deeper(\ T_{given}\ );$
$signal := failure;$
$stop;$
$end.\ main$

Algorithmus 2: Bottom Up Parser

man noch etwas zusätzlichen Platz, aber insgesamt erscheint der Platzbedarf akzeptabel.

Für die Rechenzeiten können hingegen nur sehr rasch wachsende 'worst case' Grenzen angegeben werden. Die folgende einfache Beispiel-BBB-KG verdeutlicht das:

$$V = T \cup N \text{ mit } T = \{te\} \text{ und } N = \{nt, go\}$$

Als Attributraum D dienen die natürlichen Zahlen $\{1, 2, ...\}$. Es gibt vier Produktionen $P = \{p_1, ..., p_4\}$. Definiert wird

$$p_1 : (te, te) \quad \begin{matrix} true \\ \longrightarrow \\ d = {d_1}^{d_2} \end{matrix} \quad (nt),$$

wobei *true* für das triviale Prädikat steht, das immer erfüllt ist. Zielkonfiguration ist ein einzelnes Nichtterminal nt, und als Attributwert wird der Exponent ${d_1}^{d_2}$ aus den Attributwerten der Ausgangskonfiguration zugewiesen. Die anderen Produktionen unterscheiden sich nur im Symbolteil der Ausgangskonfiguration. Man läßt noch die Paare (nt, te), (te, nt) und (nt, nt) zu. Eine Produktion, deren Zielkonfiguration das Startsymbol *go* enthält, gibt es nicht. Die zugehörigen Sprachen sind also leer. Man kann sich *Top Down* Algorithmen vorstellen, die so etwas gleich am Anfang abprüfen.

Sei als Beispiel die Menge $T_{given} = \{\binom{te}{2}, \binom{te}{3}, \binom{te}{5}\} = S_0$ gegeben. So erhalten wir zunächst z. B.

$$S_0 \xrightarrow{p_1} S_1 = \{\binom{nt}{8}, \binom{te}{5}\} \xrightarrow{p_2} S_2 = \{\binom{nt}{32768}\}$$

und dann nach Back Tracking $S_1 \xrightarrow{p_3} S_3 = \{\binom{nt}{390625}\}$. Damit ist S_1 abgearbeitet und zweimaliges Back Tracking liefert

$$S_0 \xrightarrow{p_1} S_4 = \{\binom{nt}{32}, \binom{te}{3}\} \xrightarrow{p_2} S_5 = \{\binom{nt}{32768}\}$$

und dann nach Back Tracking $S_4 \xrightarrow{p_3} S_6 = \{\binom{nt}{3^{32}}\}$. Damit ist S_4 abgearbeitet und zweimaliges Back Tracking liefert

$$S_0 \xrightarrow{p_1} S_7 = \{\binom{nt}{9}, \binom{te}{5}\} \xrightarrow{p_2} S_8 = \{\binom{nt}{59049}\}$$

und dann nach Back Tracking $S_7 \xrightarrow{p_3} S_9 = \{\binom{nt}{1953125}\}$. Damit ist S_7 abgearbeitet und zweimaliges Back Tracking liefert

$$S_0 \xrightarrow{p_1} S_{10} = \{\binom{te}{2}, \binom{nt}{243}\} \xrightarrow{p_2} S_{11} = \{\binom{nt}{59049}\}$$

und dann nach Back Tracking $S_{10} \xrightarrow{p_3} S_{12} = \{\binom{nt}{2^{243}}\}$. Damit ist S_{10} abgearbeitet und zweimaliges Back Tracking liefert

$$S_0 \xrightarrow{p_1} S_{13} = \{\binom{nt}{25}, \binom{te}{3}\} \xrightarrow{p_2} S_{14} = \{\binom{nt}{15625}\}$$

und dann nach Back Tracking $S_{13} \xrightarrow{p_3} S_{15} = \{\binom{nt}{3^{25}}\}$. Damit ist S_{13} abgearbeitet und zweimaliges Back Tracking liefert

$$S_0 \xrightarrow{p_1} S_{16} = \{\binom{nt}{125}, \binom{te}{2}\} \xrightarrow{p_2} S_{17} = \{\binom{nt}{15625}\}$$

und dann nach Back Tracking $S_{16} \xrightarrow{p_3} S_{18} = \{\binom{nt}{2^{125}}\}$. Damit sind S_{16} und S_0 abgearbeitet. Dreimaliges Back Tracking führt aus der Rekursion hinaus ins Hauptprogramm, wo vor dem Halt noch die Variable *signal* auf *failure* gesetzt wird.

Jeweils werden alle geordneten Paare ohne Wiederholung aus einer Menge S aufgezählt. Für $|S| = n$ sind das $n(n-1)$ Paare. S' hat dann in der Regel die Kardinalität $n-1$. Wenn es die Zielinstanz schon in der Restmenge gibt, gilt $|S'| = n-2$. In diesem Beispiel ist dieser Fall durch die Wahl der Attributfunktion ${d_1}^{d_2}$, wie man leicht einsieht, beschränkt auf die Zahlen 2 und 4. Wenn man von Mengen, die diese beiden Terminale enthalten, absieht, so ergibt sich durch Induktion die exakte Rechenkomplexität als $n(n-1)(1+(n-1)(n-2)(1+...3 \cdot 2(1+2 \cdot 1)...))$ Ersetzungen. Das ist größer als $(n-1)!^2$. Die Induktionsformel lautet

$$f(i+1) = (i+1) \cdot i \cdot (1+f(i)) = (i+1) \cdot i \cdot f(i) + (i+1) \cdot i.$$

Das ist kleiner als $n!^2$ mit der Induktionsformel

$$f(i+1) = (i+1)^2 \cdot f(i) = (i+1) \cdot i \cdot f(i) + (i+1) \cdot f(i).$$

Schon ganz primitive BBB-KGs können demnach mit diesem Algorithmus einen Aufwand von der Ordnung $n!^2$ erfordern. Auch die nachfolgend beschriebenen Verfahren bringen hier keine Verbesserung. Aber durch das Beispiel werden auch einige Nachteile von Algorithmus 2 deutlich, die behoben werden können:

- Es wird ein Rekursionsbaum aufgespannt, der sich stark verzweigt. Berechnungen auf zwei getrennten Zweigen (also an Knoten, die nicht in einer Vorgänger-Nachfolger-Beziehung stehen) kommunizieren nicht miteinander. Insbesondere können an unterschiedlichen Stellen, unabhängig voneinander identische Mengen reduziert werden. Im Beispiel gilt $S_2 = S_5$, $S_8 = S_{11}$ und $S_{14} = S_{17}$. Enthielte S_0 mehr Instanzen, so würde der Parser von diesen Knoten aus mit völlig identischen Berechnungen fortfahren. Er würde Rechenkapazität für überflüssige, weil schon durchgeführte Unterbäume verschwenden.

- Die Rekursion ist unübersichtlich beim Verwalten und damit vor allem bei der Entwicklung, Fehlersuche und Wartung von KGs.

Es bietet sich an, nach einem nicht-rekursiven Algorithmus zu suchen, der das gleiche leistet, aber die aktuelle Menge global hält, statt sie dauernd - mit kleinen Veränderungen - als Parameter zu kopieren. Dies leistet der nächste Abschnitt.

3.3 Kumulative Parser

Das folgende Verfahren ist eine Näherungslösung, denn erstens löst es das Wortproblem nicht für $\mathcal{L}_{noisy}$ sondern für die Obermenge $\mathcal{K}$ und zweitens kann es in eine Endlosschleife geraten, auch wenn das Wortproblem entscheidbar ist. Die Instanzen der Ausgangskonfiguration werden nicht aus der Datenbasis entfernt. Man nimmt lediglich alle möglichen Zielkonfigurationen hinzu. Hier sollen solche Verfahren daher analog zu den Sprachbegriffen als *kumulative Parser* bezeichnet werden.

Eine gewisse Verwandtschaft zu den aus der klassischen Syntaxanalyse der Stringgrammatiken unter dem Begriff *Tabellenparser* seit langem bekannten Verfahren kann nicht geleugnet werden. Auch diese arbeiten nicht 'ersetzend'. Erstes Beispiel ist der CYK-Parser (siehe [KASA-65, YNGR, HAYS]). Für nicht mehrdeutige Grammatiken - das ist keine wesenliche Einschränkung - gibt es noch eine schnellere Version, den Early-Parser, (siehe [EARL, KASA-69]). Auch diese Verfahren sind in [AHO] beschrieben. Man fügt eine zusätzliche 'Dimension' hinzu, indem man, statt auf den Worten, auf den besagten Tabellen arbeitet. Darin werden alle möglichen Reduktionsbäume konstruiert.

Die Analogie zwischen kumulativen Parsern für KGs und Tabellenparsern darf aber nicht überstrapaziert werden. Die KGs sind auf Mengen definiert. Es gibt keine 'Positionen' mit denen man die Reduktionsgraphen kodieren könnte. Es gibt auch keine Garantie, daß man auf niedrige Rechenkomplexität herunterkommt. Man kann nicht einmal Endlosschleifen oder überexponentielles Wachstum im *worst case* ausschließen. Darüberhinaus gilt: Für Koordinaten Grammatiken kann man mit einem kumulativen Parser eine Ausgangsmenge von Terminalen T_{given} nur daraufhin untersuchen, ob sie in der kumulativen Sprache $\mathcal{K}$ der KG liegt. Wenn aber einmal eine Kumulation einer Instanz des Startsymbols g konstruiert ist, läßt sich leicht - so zu sagen 'straight forward' also ohne Rekursion oder 'back tracking' - entscheiden, ob sich daraus eine Reduktion dieser Instanz ergibt, wenn man jeweils die Ausgangskonfigurationen aus der Datenbasis entfernt. Ein kumulativer Parser ist mit dem Algorithmus 3. gegeben.

Die Menge Q (für 'queue') enthält alle aktuell abzuarbeitenden Ausgangskonfigurationen zusammen mit den jeweiligen Produktionen. Pr (für 'processed') enthält alle bereits abgearbeiteten Paare dieser Art. Zusammen ergeben sie eine Art Warteschlange. In der letzten Zeile vor dem $end\ While$; muß Q aktualisiert werden, da die Menge S ja größer geworden ist. Im allgemeinen gibt es daher jetzt mehr Ausgangskonfigurationen darin. Wenn die Bedingungsprädikate zu schwach sind, dann wird Q stark wachsen. Pr wächst mit jedem Durchlauf der $While$ Schleife um ein Element. Die Beispiel KG aus dem Abschnitt 3.2 mit der dortigen Menge T_{given}, mit der der Algorithmus 2 ja durchaus hält, führt den Algorithmus 3 in eine Endlosschleife. Q wächst dort schon nach wenigen Schritten

ALGORITHM $Kumulative\_Bottom\_Up($ $\mathcal{U}$: $Universe$;
P : $Set\ of\ Productions$;
g : $Nonterminal$;
T_{given} : $Set\ of\ Terminal\_Instance$;
$VAR\ \ signal$: $\{failure, success\}$);

$VAR\ S$: $Set\ of\ Instance$;
p : $Production$;
Q, Pr : $Set\ of\ Pair\ of\ Production, Configuration$;
κ_a, κ_z : $Configuration$;
I : $Instance$;

Begin main
$S := T_{given}$;
$Q := \{(p, \kappa) \in P \times \mathcal{U}^*;\ \kappa \in \mathcal{A}_p \ \wedge\ \mathcal{B}(\kappa) \subseteq S\}$;
$Pr := \emptyset$;
While $Q \neq Pr$ *do*
Choose $(p, \kappa_a) \in Q \setminus Pr$;
$Pr := Pr \cup \{(p, \kappa_a)\}$;
$p : \kappa_a \longmapsto \kappa_z$;
$S := S \cup \mathcal{B}(\kappa_z)$;
If $\exists I \in S$: $s(I) = g$ *Then*
$signal := success$;
stop;
end If;
$Q := \{(p, \kappa) \in P \times \mathcal{U}^*;\ \kappa \in \mathcal{A}_p \ \wedge\ \mathcal{B}(\kappa) \subseteq S\}$;
end While;
$signal := failure$;
stop;
end main.

Algorithmus 3: Kumulativer Parser

S	$Q \setminus Pr$
$\{I_1, I_2, I_3, I_4, I_5, I_6\}$	$\{(p_1, (I_3, I_4)), (p_1, (I_4, I_3))\}$
$\{I_1, I_2, I_3, I_4, I_5, I_6, I_7\}$	$\{(p_1, (I_4, I_3)), (p_2, (I_2, I_7))\}$
$\{I_1, I_2, I_3, I_4, I_5, I_6, I_7, I_8\}$	$\{(p_2, (I_2, I_7))\}$
$\{I_1, I_2, I_3, I_4, I_5, I_6, I_7, I_8, I_9\}$	$\{(p_3, (I_6, I_9))\}$
$\{I_1, I_2, I_3, I_4, I_5, I_6, I_7, I_8, I_9, I_{10}\}$	–

Table 3.1: **Beispiel für einen kumulativen Parserlauf**

sehr viel schneller als Pr, denn mit jeder zu S hinzukommenden Instanz ergeben sich immer mehr Kombinationsmöglichkeiten, von denen das triviale Prädikat $true$ keine verbietet.

In gutmütigeren Fällen ergibt sich ein anderer Ablauf. Das Beispiel aus dem Abschnitt 2.2.3 ergibt den in der Tabelle 3.1 abgebildeten Parserlauf, wenn der $Choose$ Operator jeweils das erste Element in der angegebenen Reihenfolge liefert. Die Instanzen I_1 bis I_6 sind dabei die Terminale. I_7 ist die auch im Abschnitt 2.2.3 angegebene Instanz des Symbols L_Linie. Die Instanz I_8 wird in der Reduktion im Abschnitt 2.2.3 nicht konstruiert. Sie entsteht, indem die Produktion p_1 angewandt wird auf das Paar (I_4, I_3), auf das das Bedingungsprädikat $col \wedge ovl$ auch zutrifft. Die Aufzählungsreihenfolge ist hier gegenüber der Reduktion der Instanz I_7 vertauscht. Es ergibt sich eine Instanz des Symbols L_Linie mit den Attributwerten $P_1 = (70, 136)$ $P_2 = (70, 113)$ und $O = 90^o$. Diese Instanz kann mit keiner anderen vorhandenen Instanz zu einer Ausgangskonfiguration gepaart werden. Q wird also an dieser Stelle nicht größer. Die Instanzen I_9 und I_{10} sind die in Abschnitt 2.2.3 ebenfalls konstruierten Instanzen der Symbole $Winkel$ bzw. $O\_Viereck$. Da letzteres das Startsymbol ist, hält der Algorithmus 3 an dieser Stelle mit der Meldung $success$. Das aktuelle Q wird gar nicht mehr gebildet. Es würde sich vom vorangehenden auch nicht unterscheiden. Damit wäre die $While$ Schleife ohnehin abgearbeitet, da nunmehr $Q = Pr$ gelten würde. Ob Instanzen wie I_8 konstruiert werden, hängt von der Reihenfolge der Abarbeitung von $Q \setminus Pr$ ab. Wenn die durch Aktualisierung hinzukommenden Verarbeitungselemente hinten angefügt werden, und das jeweils Vorderste ausgewählt wird (Warteschlange), wird *breadth first* gesucht. Wenn das jeweils zuletzt hinzugefügte zuerst verarbeitet wird (Stapel), dann gibt es eine *depth first* Suche. Zwischenstrategien sind möglich durch zufällige Auswahl oder Auswahl nach Bewertungskriterien.

Nach Marr wäre dieser Algorithmus suboptimal. Es kann sein, daß eine Kumulation konstruiert wurde, aus der keine Reduktion ableitbar ist. In der Praxis hat sich dieser Nachteil als kaum gravierend erwiesen. Wichtiger ist, daß es nicht möglich ist, eine einfache polynomiale Schranke für den Speicherplatzbedarf anzugeben. Es wird keine Instanz gelöscht. Wenn also die Anzahl der möglichen

Ausgangskonfigurationen mit der Tiefe der Kumulation rasch anwächst, so wächst für diesen Algorithmus auch der *worst case* Speicherplatzbedarf mit. Es ist leicht, sehr einfache, streng monotone KGs zu konstruieren, die schon mit moderaten n jeden verfügbaren Speicher überlasten. Der Algorithmus 2 mag vom Speicheraufwand zwar unkritisch sein, aber dafür überschreitet er schon für moderate n jede vernünftige Rechenzeit. Die vorliegende Arbeit stellt sich auf den Standpunkt, daß Verfahren, die in machbarer Zeit fertig werden, speicherplatzmäßig ebenfalls unkritisch sind.

Algorithmus 3 bietet eine gute Möglichkeit zur parallelen Abarbeitung: Man kann $Q \setminus Pr$ und S global halten, den Auswahloperator $Choose$ soviele Elemente aus $Q \setminus Pr$ entnehmen lassen, wie Prozessoren zur Verfügung stehen, und den Block danach bis zum *end While*; parallel durcharbeiten lassen. Mit der Aktualisierung von $Q \setminus Pr$ muß dabei jeweils gewartet werden, bis alle Prozessoren fertig sind mit der Aktualisierung von S.

Sonst an dieser Stelle in syntaktisch inspirierten Arbeiten übliche Nachweise über eine bestimmte polynomiale Komplexität bestimmter Parser können nicht geführt werden. Das bedeutet aber keinen Nachteil für die Praxis. Im Abschnitt 5.3 werden Methoden vorgestellt, die es erlauben, die *voraussichtliche* Rechenkomplexität bestimmter KGs aus einer Statistik der in T_{given} vorkommenden Attributwerte abzuschätzen.

3.4 Assoziativspeicher

Der Algorithmus 3 beinhaltet eine Auflistung von möglichen Paaren aus je einer Produktion p und einer Ausgangskonfiguration aus $\mathcal{A}_p \subset \mathcal{U}^*$. In der Praxis kann die Bildung und Aktualisierung dieser Menge bzgl. einer wachsenden Datenbasis S erheblichen Aufwand bedeuten. Wenn k die Länge der rechten Seite Σ einer Produktion p ist und n die Anzahl der Elemente in einer aktuellen Menge S von Instanzen, dann hat die Menge aller k-Tupel in S n^k Elemente. Die Auflistung aller p-Ausgangskonfigurationen darin hat also eine polynomiale Komplexität der Ordnung k. Hierin liegt ein gewichtiger Grund BBB-KGs zu verwenden, weil dann $k = 2$ gilt.

Natürlich braucht man nur jene Paare von Instanzen zu betrachten, in denen der Symbolteil mit Σ übereinstimmt. Besonders wichtig ist aber, daß für viele Prädikate π die *Menge aller möglichen Partner* in S, die eine gegebene Instanz zu einer Ausgangskonfiguration vervollständigen, leicht durch Suchbereiche im Attributbereich D konstruiert werden kann. Abbildung 3.2 zeigt, daß sich solche Suchbereiche zu einfachen Prädikaten (in diesem Falle ist es Nachbarschaft im Bildkoordinatenraum) auch sehr schön veranschaulichen lassen.

Wenn man also die Instanzenmenge S so abspeichert, daß man Untermengen da-

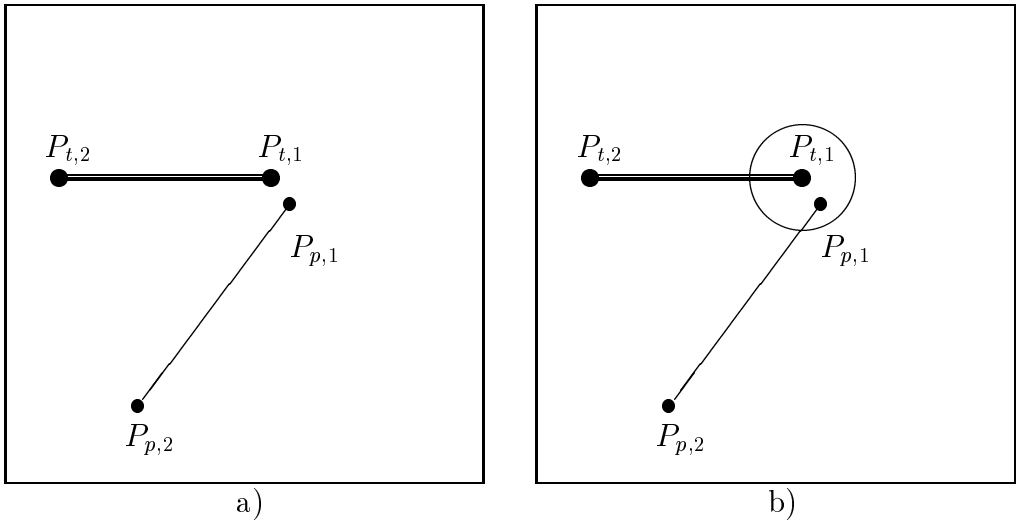


Figure 3.2: **Graphische Veranschaulichung eines Suchbereichs**
a) Die fett gezeichnete 'triggernde' Linien Instanz I_t sei gegeben. Gesucht ist die Menge aller 'Partner' Instanzen I_p, des Symbols Linie, die $adj(P_{t,1}, P_{p,1})$ erfüllen.
b) Das Prädikat ist (für euklidisches Abstandsmaß) erfüllt,
wenn der Punkt in dem Kreis mit Radius d_{max} liegt.

raus bzgl. solcher Suchbereiche assoziativ nach ihren Attributwerten bilden kann, dann hat man eine Größenordnung gewonnen, und ist bei linearer Komplexität. Voraussetzung ist, daß der Zugriff unabhängig von der Größe von S ist ('constant time' in n), und daß die Größe der Untermengen, die man erhält, ebenfalls nicht von n abhängt.

Die Instanz I_t aus der Abbildung 3.2 ist jene Instanz, aus deren Attributwerten der Suchbereich berechnet wird. Entsprechend der üblichen Diktion im KI Diskurs wird I_t als 'triggernde' Instanz bezeichnet. Dies entspricht der Deutung der Produktionen einer KG als *Regeln* oder *Wissensquellen* in einem wissensbasierten KI System (siehe Abschnitt 4.1). Die Instanz alleine genügt nicht, um die Geometrie des Suchbereichs eindeutig festzulegen. Dazu benötigt man zusäztlich die Produktion und im allgemeinen auch die Position innerhalb der Ausgangskonfiguration[1]. Es wird daher definiert: Ein Tripel

$$VE \;=\; (I, p, j)$$

heißt **Verarbeitungselement** einer $KG = (T, N, D, n, P, g)$ genau dann, wenn $I \in \mathcal{U}$, $p = (\Lambda, \Sigma, \pi, \phi) \in P$ mit $1 \leq j \leq k$ und $s(I) = \Sigma_j$ gilt. Die Menge aller Verarbeitungselemente sei mit $\mathcal{V}$ bezeichnet. Zu solchen Verarbeitungselement gehört eine Partnermenge

$$Pa_{VE} \;=\; \{\kappa \in \mathcal{A}_p;\; \kappa_j = I \;\wedge\; \mathcal{B}(\kappa) \subseteq S\}.$$

Somit gewinnt der kumulative Parser die in Algorithmus 4. gegebene Gestalt.

Man beachte, daß hier zum Aufstellen und Aktualisieren der Warteschlange Q kein Wissen über die in den Produktionen enthaltenen Prädikate und Funktionen erforderlich ist. Für jede Instanz $I \in S$ ist lediglich aufzulisten, in welcher Produktion p in der rechten Seite Σ der Symbolteil $s(I)$ an j-ter Stelle auftaucht. Das Wissen für die globale Verwaltung von S und Q ist also recht übersichtlich. Die Details jeder Produktion (Bestimmung der Suchbereiche, Bildung der Partnermenge Pa und Konstruktion der Zielkonfiguration κ_z mit der Attributberechnungsfunktion ϕ) können lokal in einem speziellen Modul dafür zusammengefaßt werden. Gegenüber dem Algorithmus 3 ergibt sich allerdings ein erhöhter Rechenaufwand.

- Die Bestimmung der Geometrie der Suchbereiche für die Partnermengen ist zusätzliche Arbeit.
- Die Menge Q ist sehr viel größer. Anhand des Beispiels aus Abschnitt 2.2.3 wird das deutlich. S_0 besteht aus sechs Instanzen des Symbols *Linie*. Da

[1] Im Falle der Beispielproduktion zur Winkelbildung muß nur eine Position berücksichtigt werden, wenn man für die Winkel Instanzen die $\mathcal{S}_2$ Symmetrie zuläßt, die sich in der rechten Seite und in π und ϕ wiederfindet (siehe Abschnitt 5.2.3)

ALGORITHM $Associative\_Parse($ $\mathcal{U}$: $Universe$;
P : $Set\ of\ Productions$;
g : $Nonterminal$;
T_{given} : $Set\ of\ Instance$;
$VAR\ signal$: $\{failure, success\}$);

$VAR\ S$: $Set\ of\ Instance$;
p : $Production$;
Q : $Set\ of\ Tripel\ of\ Instance,\ Production,\ Number$;
Pa : $Set\ of\ Configuration$;
κ_a, κ_z : $Configuration$;
I : $Instance$;

Begin main
$S := T_{given}$;
$Q := \{(I,p,j) \in \mathcal{U} \times P \times \mathbb{N};\ I \in S \ \wedge\ s(I) = \Sigma_{p,j}\}$;
While $Q \neq \emptyset$ *do*
Choose $(I,p,j) \in Q$;
$Q := Q \setminus \{(I,p,j)\}$;
$Pa := \{\kappa \in \mathcal{A}_p;\ \kappa_j = I \ \wedge\ \mathcal{B}(\kappa) \subseteq S\}$;
$\forall \kappa_a \in Pa$ *do*
$p:\ \kappa_a \longmapsto \kappa_z$;
$S := S \cup \mathcal{B}(\kappa_z)$;
$If\ \exists I \in S\ :\ s(I) = g\ Then$
$signal := success$;
stop;
end If;
$Q := Q \cup \{(I,p,i) \in \mathcal{U} \times P \times \mathbb{N};\ I \in \mathcal{B}(\kappa_z) \ \wedge\ s(I) = \Sigma_{p,j}\}$;
end $\forall \kappa_a$;
end $While$;
$signal := failure$;
stop;
end. main

Algorithmus 4: Parser mit assoziativer Abfrage

dieses Symbol in den rechten Seiten der Produktionen p_1 bis p_3 insgesamt viermal auftaucht, besteht Q anfangs aus vierundzwanzig Verarbeitungselementen. Entsprechend viele Suchbereiche werden konstruiert.

- Jede rechte Seite Σ wird k-fach untersucht, nämlich mit jeder der Instanzen der Ausgangskonfiguration in der Rolle des triggernden Verarbeitungselementes einmal. Man hat also etwa bei BBB-KGs den doppelten Aufwand. Dieser Mehraufwand kann verhindert werden, wenn man für jedes Σ eine einzige Position für das triggernde Element festlegt (z. B. immer die erste). Dann muß aber sichergestellt sein, daß alle aus S_0 reduzierbaren Instanzen der Symbole Σ_2 bis Σ_k bis dahin konstruiert wurden. Und damit ist die Freiheit in der Verarbeitungsreihenfolge verloren. Im Prinzip läuft es dann auf *breadth first* Suche hinaus, und man blockiert sich die Beschleunigungspotentiale einer Abarbeitungssteuerung nach Bewertungen.

Für sehr einfache KGs und bei kleinem T_{given} (wie im Beispiel aus Abschnitt 2.2.3) bringt der Algorithmus 4 also nur ungerechtfertigten Mehraufwand. Für KGs mit zu schwachen Bedingungsprädikaten, wie aus dem Abschnitt 3.2, wächst Q sehr schnell an. Wie bei Algorithmus 3 gibt es keine Garantie, daß Algorithmus 4 überhaupt hält - nicht einmal dann, wenn Algorithmus 2 noch korrekte Resultate liefert. Die Sache lohnt also nur dann, wenn einerseits T_{given} groß genug ist, so daß der besagte Komplexitätsvorteil wirklich durchschlägt, und andererseits die Bedingungsprädikate die Kombinatorik hinreichend beschränken.

Für den assoziativen Zugriff nach Attributwerten in Suchbereichen darf die Datenbasis nicht einfach als Liste organisiert sein, sondern es bedarf gewisser Standardverfahren (siehe [SCHL]) der Kodierung für assoziative Datenbanken. In der Praxis können nicht alle relevanten Prädikate durch assoziativen Zugriff mit einfachen Suchbereichen voll beschrieben werden. Es bringt aber oft schon etwas, eine enge Obermenge von Pa_{VE} durch den assoziativen Zugriff bestimmen zu können, und darin dann das Prädikat seriell abzuprüfen.

Chapter 4

Parallelisierbarkeit und Implementierungsmöglichkeiten

Der Abschnitt 4.1 beschäftigt sich zunächst mit einem klassischen, sehr modularen KI Ansatz, der Blackboard Architektur. Dabei liegt die Betonung auf Wissenserwerb und Übersichtlichkeit in der Verfahrensentwicklung. Es wird aber keine der gängigen KI Sprachen und KI Maschinen verwendet. Zwar wurden auch damit gelegentlich Versuche unternommen, insbesondere wurden solche Produktionssyteme in LISP auf einer SYMBOLICS-Maschine und in OPS5 auf einem VAX-Rechner implementiert, aber diese Arbeiten wurden 1989 zu Gunsten von spezieller Software und Hardware aufgegeben. Das Rechenzeitverhalten war mit steigender Zahl der Einträge nicht akzeptabel [ANDE]. Dies mag wohl daran liegen, daß die zugehörigen Interpreter in der Regel das korrekte, ersetzende Verfahren verwenden. Die Speziallösung implementiert das in Abschnitt 3.4 beschriebene kumulierende Näherungsverfahren mit einem Blackboard.

Die Art, wie die Daten repräsentiert und gespeichert werden, und wie die Algorithmen in welcher Parallelität durchführbar sind, gewinnt entscheidende Bedeutung. Es ist notwendig, daß die Verfahren durch massiv parallele Spezialrechner implementierbar sind, deren Architektur speziell auf sie zugeschnitten ist. Im Abschnitt 4.2 wird auf eine, für viele Bildverarbeitungsaufgaben natürliche, Rechnerarchitektur - die Pyramide - eingegangen. Hierfür müssen die Produktionen im wesentlichen ziel-lokal und verschiebungsinvariant sein, und man beschränkt sich meist auf zweidimensionale Muster. Am FIM geht man einen anderen Weg, der hohe Parallelität verspricht, ohne diese starken Einschränkungen zu machen. Dieser Ansatz ist in 4.3 beschrieben.

4.1 Blackboard Architektur

Die algorithmische Struktur der Parser in den Abschnitten 3.3 und 3.4 kann als **Blackboard Modell** im Sinne der KI verstanden werden, wenn man z. B. [ENGE] oder [WINS] heranzieht. Die Menge S der aktuell kumulierten Instanzen ist dann das Blackboard. Als Wissensquellen dienen die einzelnen Produktionen. Zwischen den Produktionen ist kein direkter Informationsfluß nötig. Sie können selbständig in S^k nach entsprechenden Ausgangskonfigurationen suchen, und im Erfolgsfalle eine Zielkonfiguration hinzufügen. Die Auswahloperatoren $Choose$ und $\forall$ lassen die Reihenfolge bewußt offen. Zu einem solchen Balckboard Modell stehen die klassischen Experten Systeme in Kontrast, bei denen die Datenbasis, die Wissensbasis und die Abarbeitungskontrolle die Reihenfolge der Produktionen festlegen. Die Semantik eines solchen Systems kann nur *als Ganzes* verstanden werden. Wenn man aber die Wissensbasis zerlegt in übersichtlichere Wissensquellen, die nur über ein Blackboard kommunizieren, so kann die Semantik jeder Wissensquelle für sich betrachtet werden. Wenn das Endergebnis dann nicht von der Abarbeitungsreihenfolge abhängt, so kann die Semantik des gesamten Verfahrens als Summe der Teilbedeutungen aufgefaßt werden [STIL-97]. Im vorliegenden Fall kann die Struktur der Sprachen $\mathcal{L}$ und $\mathcal{K}$ einer KG verstanden werden durch das isolierte Studium der Relation zwischen $\mathcal{A}_p$ und $\mathcal{Z}_p$ der einzelnen Produktionen p und die anschließende Analyse der Möglichkeiten, die dadurch zugelassen werden.

Auf dem Weg von den semantischen Überlegungen zur Implementierung ist der nächste Schritt in der Spezifizierung der Abarbeitungsreihenfolge zu sehen. Man fügt ein Kontrollmodul (den sogenannten Dispatcher) hinzu und gelangt so zur **Blackboard Architektur**. Klassisches Vorbild für Mustererkennungsverfahren dieser Art ist wohl das Hearsay-II System (siehe [LESS]). Die Abbildung 4.1 verdeutlicht die Modularität und den Informationsfluß zwischen den Modulen. Die Menge S der Instanzen, die im Laufe des Verfahrens akkumuliert werden, wird im Assoziativspeicher (rechts als große Box dargestellt) global gehalten. Das geschieht so, daß alle Regelmodule (dargestellt in der Mitte als Kästchen $r_{i,j}$) assoziativ nach den Attributwerten und dem Symbolteil Mengen darin bilden können. In der Abbildung 4.1 ist dieser Zugriff durch kleine Kreise angedeutet. Wenn ein Regelmodul die so gebildete Menge Pa nicht leer findet, so trägt es die Instanzen der entstehenden Zielkonfigurationen (sofern noch nicht darin enthalten) in das Blackboard ein. Außerdem kommen die neuen Instanzen auch in die Menge der Verarbeitungselemente Q, die in der *Process Queue* (als flache Box oben in der Abbildung dargestellt) gehalten wird. Dazu wird zunächst der Verarbeitungsindex *nil* für 'noch nicht festgelegt' zugeordnet. Dies geschieht im Interesse der Modularität. Das Wissen, welches Symbol aus der linken (Ziel) Seite Λ einer Produktion p in der rechten (Ausgangs) Seite welcher anderen Produktionen q wo gebraucht wird, gehört nicht zu einer Regel, die p implementiert.

Dieses Wissens soll in einem globalen Verteilermodul dem *Dispatcher* (in der Zeichnung links als schmale, hohe Box dargestellt) gehalten werden. Dieses Modul erhält von der *Process Queue* jeweils ein Verarbeitungselement. Ist dieses von der Form (I, nil), so reicht der *Dispatcher* die Instanz I für jede rechte Seite einer Produktion p_i, in der das Symbol $s(I)$ vorkommt und für jede Position j, in der es vorkommt, als entsprechend besetztes Verarbeitungselement (I, p_i, j) wieder zurück an die *Process Queue*, was in der Abbildung 4.1 durch die Pfeile links oben angedeutet ist. Ist ein dem *Dispatcher* zufließendes Verarbeitungselement von der Form (I, p_i, j), so wird es an das entsprechende Regelmodul $r_{i,j}$ weitergereicht, und triggert dort den Assoziativzugriff zur Bestimmung der Menge der passenden Ausgangskonfigurationen $Pa_{(I,p_i,j)}$ (siehe auch [STIL-95-1]).

Die Reihenfolge der Abarbeitung ist, wie oben dargelegt, irrelevant. Alle im Kapitel 3 angeführten Algorithmen sind so konstruiert, daß für die Auswahl- bzw. Auflistungsoperatoren keine bestimmte Reihenfolge vorgeschrieben werden muß. Das Steuermodul *Process Queue* muß nicht unbedingt eine Warteschlange sein. Ebensogut kann es als Stapel oder als Zufallsurne organisiert sein. Der Verfahrensentwickler wird eine Umgebung als komfortabler empfinden, die 'gute' Instanzen zuerst abarbeitet. Voraussetzung ist die Definition eines Kriteriums, welches die Qualität einer Instanz nicht für sich allein, sondern die Wahrscheinlichkeit seiner Eignung für die weitere Reduktion bis hin zum Startsymbol widerspiegelt.[1] Der Algorithmus wird so zu einer gezielten *depth-first* Suche, die sogar dann noch brauchbare Resultate bei Detektionsaufgaben liefern kann, wenn das Verfahren an sich auf der vorhandenen Hardware nicht fertig laufen könnte. Man kann ja hier abbrechen, wenn etwas gefunden wurde, und eine Schranke setzen, wie lange man andernfalls rechnen möchte. Auch bei zeitkritischen Anwendungen kann die Auflistung der Verarbeitungselemente in einer Warteschlange nach gewissen Bewertungskriterien hilfreich sein. Diese Arbeit verzichtet darauf, den Versuch zu unternehmen, solche Optimierungen etwa mit den Methoden der Wahrscheinlichkeitsrechnung, mit der Evidenztheorie nach Dempster-Shafer oder den Zadehschen *fuzzy sets* zu modellieren, und stellt sich auf den Standpunkt, daß im Regelfall ein sicheres Kriterium nicht gegeben werden kann. Ein Gegenbeispiel zeigt dies deutlich: Eine terminale Konturinstanz kann nur bewertet werden auf-

[1] Hier sind Überlegungen zum Thema *dynamische Optimierung* anstellbar (siehe etwa [BELL] oder die üblichen Handbücher). In diesem Zusammenhang heißt das entsprechende Kriterium *Separierbarkeit*. Danach muß ein Problem zerlegbar sein in Stufen und eine z. B. additive Kostenfunktion existieren, die sich auf jeder Stufe neu errechnet zum einen aus den Kosten der verschiedenen Möglichkeiten bis dahin und zum anderen aus den Kosten für die aktuell möglichen Schritte. Wird ein 'billigerer' Weg zu einem schon bewerteten Zwischenzustand gefunden, so muß der teurere nicht weiter verfolgt werden, denn die optimale Gesamtlösung kann keine suboptimalen Teillösungen enthalten. Wenn noch eine zweite, ebenfalls additive Funktion zur Abschätzung der voraussichtlichen Kosten bis zum Ziel (hier Reduktion einer Instanz des Startsymbols) vorliegt und diese Schätzung optimistisch einerseits und nichttrivial andererseits ist, so kann die Anzahl der zu erzeugenden Zwischenzustände (hier Nichtterminale) durch Anwendung des A^*-Algorithmus weiter gesenkt werden ([NILS]).

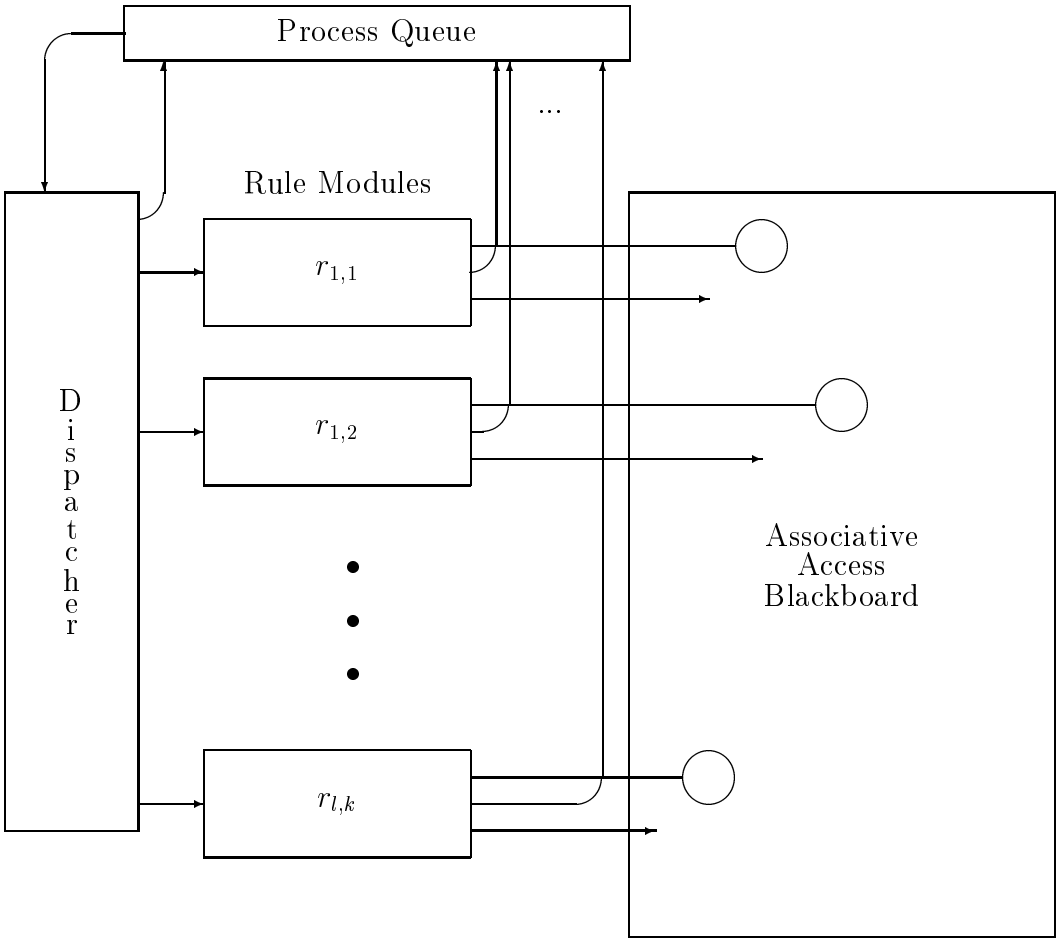


Figure 4.1: **Blackboard Flußdiagramm**
Die Terminologie orientiert sich am KI Diskurs.
Die Regeln $r_{i,j}$ sind doppelt indiziert, weil für jede Produktion p_i
auch die Position j des 'triggernden' Elements in der Ausgangskonfiguration
feststehen muß, um den durch Kreise angedeuteten assoz. Zugriff durchzuführen.

grund ihrer Güte als Kontur, also ihrer Schärfe, ihrer Glattheit, ihres Kontrastes usw. Das sagt nichts darüber aus, wie gut sie sich einfügt in eine Ausgangskonfiguration für ein größeres Gebilde. Weiter kann man einen Polygonzug aus solchen Konturterminalen nur bewerten anhand der Güte seiner Teile und ihrer Lage zueinander. Sein Wert für die weitere Analyse stellt sich erst nach Leerlaufen der Warteschlange heraus. Oft sind gerade die auffälligen Strukturen irrelevant und umgekehrt. Die Bewertung kann, zusätzlich zu ihrer Rolle in der *Process Queue*, als Attribut den Instanzen mitgegeben werden und so in die Bewertung ihrer Väter mit einfließen. Die Bewertung der Instanzen des Startsymbols kann als Entscheidungsgrundlage dienen, wenn das Verfahren eingebunden wird in einen größeren Zusammenhang (etwa einer autonomen Maschine).

Dieses serielle Vorgehen ist für die Simulation auf Standard Neumann Rechnern gedacht oder für nicht sehr zeitkritische Anwendungen mit kleinen Datenbasen. Es ist ohne allzu großen Aufwand parallelisierbar. Die größte minimale serielle Tiefe des Algorithmus ist streng genommen das Doppelte der Tiefe der tiefsten vorkommenden Reduktionen, denn alle möglichen Kumulationen können unabhängig voneinander und parallel verfolgt werden, und die anschließende Überprüfung des Reduktionsgraphen ist nochmal von der selben Tiefe. Die Betonung liegt hier aber mehr auf der Modularität in der Handhabung des Wissens, also auf Wartbarkeit, Erweiterbarkeit, Komfortabilität bei der Fehlersuche usw.

4.2 Parallelrechner mit Pyramidenstruktur

Rosenfeld schlägt in [ROSE-89-2] einen Parallelrechner mit pyramidenförmiger Kommunikationsstrukur wie in Abbildung 4.2 veranschaulicht sowohl für die Bildverarbeitungsprozesse zur Gewinnung der Primitive als auch für den Parser selbst vor. Diese Rechnerstruktur ist, wie Rosenfeld feststellt, sicher die 'natürliche' Wahl einer Hardware für solche Parser (auch im Sinne einer Absicherung durch Analogie mit neurophysiologischen Strukturen im visuellen Wahrnehmungsapparat der Tiere und des Menschen).

Das Bestechende daran ist, daß sich eine eventuell in der Grammatik enthaltene Ordnung $\mathcal{O}$ auf den Symbolen (siehe Abschnitt 5.1.2) in der Konstruktion der Maschine abbildet. Das hat eine gewisse *form follows function* Ästhetik. Für eine ausgereifte KG sollte nach der Faustregel 1 aus Abschnitt 5.3 die Zahl der Instanzen mit steigender Ordnung des Symbolteils abnehmen. Damit wäre wohl auch zu erwarten, daß der Grad an möglicher Parallelität mit zunehmender Ordnung des Symbolteils schwindet und die Pyramide dann für eine optimale Prozessorauslastung sorgen würde.

Allerdings ist für die Erforschung und Entwicklung von Verfahren auf dem Gebiet der Bildverarbeitung und Szenenanalyse von der Konstruktion solcher Hardware

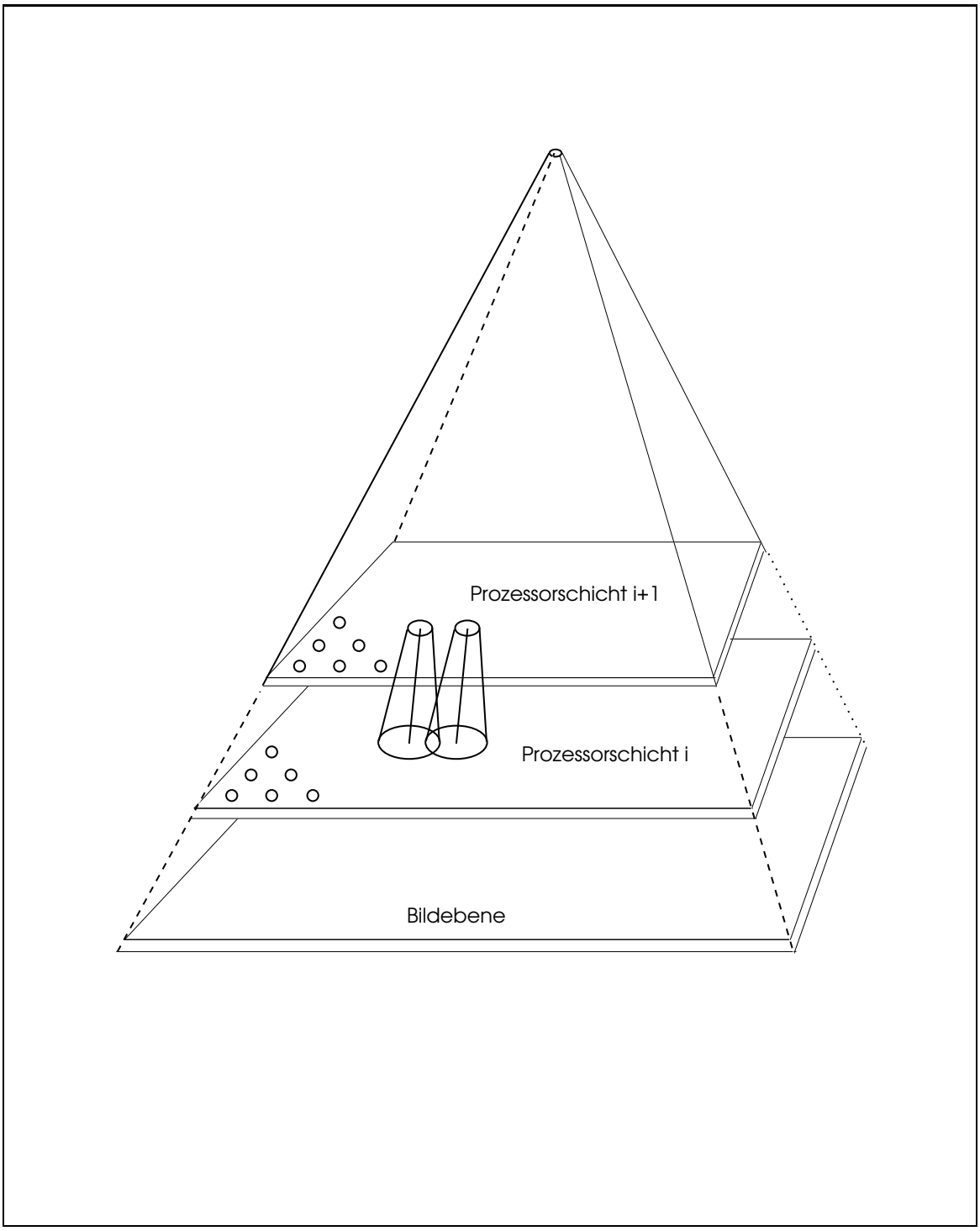


Figure 4.2: **Graphische Veranschaulichung eines Pyramidenrechners**
Die einzelnen Zellen kommunizieren nur mit den Zellen der
jeweils darunter und darüber liegenden 'Fenster'.

aus folgenden Gründen abzuraten: Erstens kann eine solche Ordnung nur für nicht rekursive KGs definiert werden, und zweitens hängt die Kommunikationstopologie - und damit die ganze Konstruktion überhaupt - von Verfahrensparametern ab. Die Prädikate und Funktionen in den Produktionen müssen ziel-lokal im Sinne des Abschnitts 2.3 sein. Es muß eine Überlappung der Kommunikationsfenster der Prozessorzellen einer Ebene auf der Ebene darunter geben, die solchen lokalen Begrenzungen entspricht. Sollen die Prozessoren auf einer Ebene alle mit demselben Programm arbeiten (was bei solchen Architekturen immer implizit mit gemeint ist), so müssen für die Produktionen entsprechende Invarianzen gelten (siehe ebenfalls Abschnitt 2.3). Die Verwendung von dreidimensionalen Attributen würde eine ganz andere Kommunikationstopologie (so etwas wie eine 'Hyperpyramide') erfordern. Dasselbe gilt für endliche, aber randlose Attributbereiche. Die ganze Architektur orientiert sich zu sehr an der Geometrie und Topologie eines bestimmten Attributraumes (bei Rosenfeld ist es wohl einfach nur das Bild). Man weiß aber, bevor man ein Verfahren für einen bestimmten Zweck implementiert und getestet hat, normalerweise noch gar nicht, welche Attributbereiche notwendig und sinnvoll sein werden. Die Erfahrung zeigt, daß während der Entwicklung gerade der Attributraum durch Hinzunahme oder Weglassen einzelner Attributbereiche, durch Änderung in Topologie und Auflösung usw. vielfachen Änderungen unterworfen ist. Für die Entwicklungsphase sind daher andere Formen von Parallelität angemessener. Ein möglicher Weg wird im nächsten Abschnitt kurz umrissen.

4.3 Assoziativspeicher und SIMD Maschinen

Der Algorithmus 4 aus Abschnitt 3.4 verlangt assoziativen Zugriff auf Teilmengen der aktuellen Instanzen nach Suchbereichen im Attributbereich. Das kann durch entsprechende Software auf Standardrechnern realisiert werden. Soll der zeitliche Aufwand zur Bildung einer solchen Menge unabhängig von der Anzahl der aktuellen Instanzen sein, so braucht man speziell dafür ausgelegte Maschinen (siehe [SCHL]). Solch ein Gerät wurde für das FIM unter der Bezeichnung WPP1 entwickelt und angefertigt. Es ist seit 1989 in Betrieb und wurde für die Entwicklung und Durchführung der in der vorliegenden Arbeit dokumentierten Beispiele eingesetzt. Im Anhang D.1 sind vergleichende Untersuchungen angestellt bzgl. des Wachstums der Rechenzeiten auf Datensätzen, die kontrolliert mit dem Zufallsgenerator erzeugt wurden. Daraus ist ersichtlich, daß die WPP1 tatsächlich im wesentlichen linear skaliert, wo die Standardrechner quadratisch skalieren. Die absoluten Zahlen verdeutlichen aber auch, daß diese Maschine inzwischen veraltet ist.

Man kann solche Spezialrechner nicht nur für den assoziativen Zugriff auslegen, sondern auch für jedes eingetragene Element einen simplen arithmetischen

Prozessor vorsehen, der einen globalen Parametersatz mit seinem Element nach globalem Programm verrechnet. So läßt sich z. B. leicht (durch Hessesche Normalform) parallel die Menge aller Elemente bestimmen, die einen bestimmten Attributwert auf einer bestimmten Seite einer Entscheidungsgerade im Bild oder Trennfläche im Raum haben. Außer den linearen Formen sind quadratische Formen interessant, die Ellipsoide, Kegel, Hyperparaboloide usw. als Trennflächen erlauben. Im übrigen kann außer dem Prädikat auch die Funktion von den einzelnen Prozessoren übernommen werden. Bei all dem ist solche Hardware nicht nur geeignet für die Ausführung solcher Produktionen in Bild oder Szene. Auch und gerade die Gewinnung der 3D Primitive durch Stereokorrespondenz aus 2D Instanzen, wie sie in Abschnitt 6.2 beschrieben ist, kann damit wesentlich eleganter und effizienter geschehen, als dies durch Eingrenzen von Epipolarbereichen in der rein assoziativen Hardware möglich ist. Insgesamt gehört eine solche Rechnerarchitektur in die Klasse der **Single Instruktion Multiple Data** Maschinen, da die Prozessoren mit einem globalen Programm auf lokalen Daten arbeiten. Eine solche Maschine ist im FIM als natürliche Fortsetzung der assoziativen WPP1 Maschinen konzipiert worden. Mit etwa 10^4 Prozessoren ist sie massiv parallel ausgelegt. Sie heißt WPP2 , und wurde im Herbst 1996 ausgeliefert und testweise in Betrieb genommen.

Chapter 5

Wissenserwerb

Der Begriff *Wissenserwerb* entstammt dem KI Diskurs. Er bezeichnet den Informationsfluß, der bei der Konstruktion eines Expertensystems vom Experten zum System durch den Konstrukteur hergestellt werden muß. An dieser Stelle ist es notwendig darauf hinzuweisen, daß sich die Forschungen am Forschungsinstitut für Informationsverarbeitung und Mustererkennung in Ettlingen auf dem Gebiet der KI nicht unerheblich von den normalerweise verfolgten Ansätzen unterscheiden. Die KI blickt inzwischen auf einige Jahrzehnte eigener Wissenschaftsgeschichte zurück und hat in [WINS, MINS] usw. ihre 'Klassiker'. Auch Rosenfeld, Fu und Anderson, auf deren Veröffentlichungen die vorliegende Arbeit aufbaut, gehören im weiteren Sinne dazu. Bei KI ist man aber geneigt, an listenverarbeitende Interpreter, an Symbolics Hardware usw. zu denken. Der Großteil der Veröffentlichungen auch in Deutschland (etwa [SAGR]) verwendet solche Strukturen. Die spezielle Aufgabenstellung des FIM mit ihren großen Datenmengen[1] führte zu anderen Soft- und Hardwarelösungen. Das Hauptgewicht liegt nicht auf einer möglichst großen Allgemeinheit und Abstraktheit der Verfahren, sondern darauf, zu demonstrieren, daß Ansätze und Verfahren aus der KI auch auf Mustererkennungsprobleme anwendbar sind, die sich durch große Datenmengen auszeichnen. Schon im Kapitel 4 wurde auf die speziellen Hardwarearchitekturen am FIM verwiesen, die für das in Kapitel 6 vorgestellte Beispiel zum Teil auch verwendet wurden. Hier werden nun Begrifflichkeiten und Konzepte vorgestellt, die beim Erstellen solcher Verfahren einen unverzichtbaren Beitrag leisten. Sie dienen der Wissenserklärung, womit sich der Abschnitt 5.1 beschäftigt, bzw. dem Wissenserwerb (Abschnitt 5.2).

Ein Vergleich mit entsprechenden Verfahren der 'klassischen KI' ist auf der Ebene der Konzepte und Ansätze durchaus möglich, auf der niedrigeren Ebene der Im-

[1] Von großen Datenmengen im Kontext der KGs kann die Rede sein, wenn $n \gg r$ gilt, wobei $n = |T_{given}|$ und r die Tiefe der tiefsten Reduktion ist. Bei den in Kapitel 6 angeführten Beispielen liegt n im Bereich von $10^5 ... 10^6$ und r im Bereich von 10^1.

plementierung aber kaum sinnvoll. Das Wissen, das bei den hier vorzustellenden Anwendungen zugrundeliegt, ist im wesentlichen *topologischer, geometrischer* und *statistischer* Natur. Die Betonung liegt also auch bei Wissenserwerb und Wissenserklärung nicht so sehr auf Logik oder Sprache sondern auf topologischen und geometrischen Relationen und Funktionen im Attributraum sowie auf der Verteilung der Attributwerte in der Präsenz großer Anzahlen von Instanzen. Das bedeutet, die Betonung liegt auf Veranschaulichung durch Computergraphiken und Statistiken. Anders als etwa bei den Stringgrammatiken kann bei den KGs, wie im Abschnitt 3 dargelegt, nicht aus einem bestimmten Grad in der Hierarchie (etwa kontextfrei) auf eine für große Datenmengen brauchbare obere Schranke in der Komplexität (etwa n^2) geschlossen werden. Es ist daher sinnvoll 'Faustregeln' aufzustellen, die es erlauben, aus einer Statistik über die Attributwerte der Terminale im Datenmaterial und den gegebenen Produktionen die voraussichtliche Rechenkomplexität abzuschätzen. So läßt sich vermeiden, daß ein Verfahren nur auf einem konkreten Bild (oder einer konkreten Bildfolge) funktioniert, auf anderen, eigentlich ähnlich erscheinenden Daten aber u. U. mit hohem Rechenaufwand in die Irre läuft. Mit solchen Schätzverfahren beschäftigt sich der Abschnitt 5.3.

5.1 Hilfsstrukturen für die Wissenserklärung

Der im Abschnitt 2.2.2 eingeführte Reduktionsbegriff erlaubt die Verwendung sehr komplexer Strukturen. Es ist daher von praktischem Nutzen, sich gewisse Systematiken und Begriffe zu schaffen, welche der Erklärung der Funktionsweise einer KG und auch der Fehlersuche beim Aufbau einer KG dienen. Die Einführung der *Reduktionsgraphen* im Abschnitt 5.1.1 folgt einem bei syntaktischen Verfahren gemeinhin üblichen Weg. Stehen nicht die einzelnen Instanzen einer bestimmten Reduktion im Mittelpunkt des Interesses, sondern sollen vielmehr allgemein die Konzepte und mit ihnen die Funktionsweise der KG insgesamt erklärt werden, so hilft es, die Zusammenhänge mit einem *Produktionsnetz* zu verdeutlichen. Mit diesem Begriff beschäftigt sich der Abschnitt 5.1.2. Im FIM, in dem seit Jahren mit Produktionssystemen gearbeitet wird, als deren mögliches theoretisches Fundament die KG nach Anderson/Rosenfeld in dieser Arbeit durchgeführt ist, hat sich der Begriff der *Dimension* eines Systems eingebürgert. Diesen Begriff präzisiert Abschnitt 5.1.3.

5.1.1 Reduktionsgraphen

Zu jeder Reduktion $S_0 \xrightarrow{*} S_r$, die durch einen der im Abschnitt 3 konstruierten Parser für eine KG durchgeführt wurde, kann ein gerichteter Graph

$$\mathcal{RG}_{S_0 \xrightarrow{*} S_r} = (K, E) \text{ mit } K \subset \mathcal{U},\ E \subset \mathcal{U}^2$$

abgespeichert werden, der sie veranschaulicht. Die Parser sind ja konstruktiv. Sie setzen eine solche Reduktion aus direkten Reduktionen $S_i \longrightarrow S_{i+1},\ i = 0, ..., r-1$ zusammen, wobei es jeweils eine konkrete Ausgangskonfiguration $\kappa_{a,i}$ und eine konkrete Zielkonfiguration $\kappa_{z,i}$ gibt. Der Graph wird schrittweise miterzeugt, indem man mit

$$K_0 = \emptyset \text{ und } E_0 = \emptyset$$

initialisiert und mit

$$K_{i+1} \ = \ K_i \cup \mathcal{B}(\kappa_{a,i}) \cup \mathcal{B}(\kappa_{z,i})$$

und

$$(I_1, I_2) \in E_{i+1} \iff I_1 \in \mathcal{B}(\kappa_{a,i}) \ \wedge \ I_2 \in \mathcal{B}(\kappa_{z,i})$$

fortschreitet. Mit $\mathcal{RG}_{S_0 \xrightarrow{*} S_r} = (K_r, E_r)$ ist dann der besagte Graph konstruiert. Dieser Graph veranschaulicht, welche Instanz an der Reduktion welcher anderen Instanz in irgendeiner Form beteiligt war. Er heißt daher auch **Ableitungsgraph** oder **Reduktionsgraph**. Im Laufe der nun schon langjährigen Arbeit mit solchen Graphen im FIM hat es sich eingebürgert, den Ausgangsknoten einer Kante als 'Vater' und den Zielknoten als 'Sohn' zu bezeichnen. Entsprechend spricht man von 'Nachfahren' und 'Vorfahren' oder 'Vorgängern'. Man gibt die Instanzen an den Knoten des Graphen normalerweise auch nicht vollständig aus, sondern beschränkt sich auf die Elementindizes der Instanzen in der Datenbasis, den Symbolteil und (oder oder) auf wenige wichtige Attributwerte. Abbildung 5.1 zeigt ein Beispiel.

Reduktionsgraphen haben sich bei der praktischen Fehlersuche und Programmierarbeit sehr bewährt. Natürlich findet sich auch einer im Kapitel 6 bei der Dokumentation der implementierten KG, wie in vielen anderen Veröffentlichungen, Forschungsberichten und Vorträgen des FIM seit etwa 1988.

5.1.2 Produktionsnetze

Steht die einzelne Reduktion im Hintergrund, und möchte man mehr die Struktur der Grammatik insgesamt verdeutlichen, so kann man ein **Produktionsnetz** zeichnen. Das Produktionsnetz ist ein Graph. Knoten sind die Symbole und Produktionen einer KG. Eine Kante führt von einer Produktion $p = (\Lambda, \Sigma, \pi.\phi)$ zu einem Symbol S, wenn S in der linken Seite Λ (Zielseite) vorkommt. Eine Kante führt von einem Symbol S zu einer Produktion $p = (\Lambda, \Sigma, \pi.\phi)$, wenn

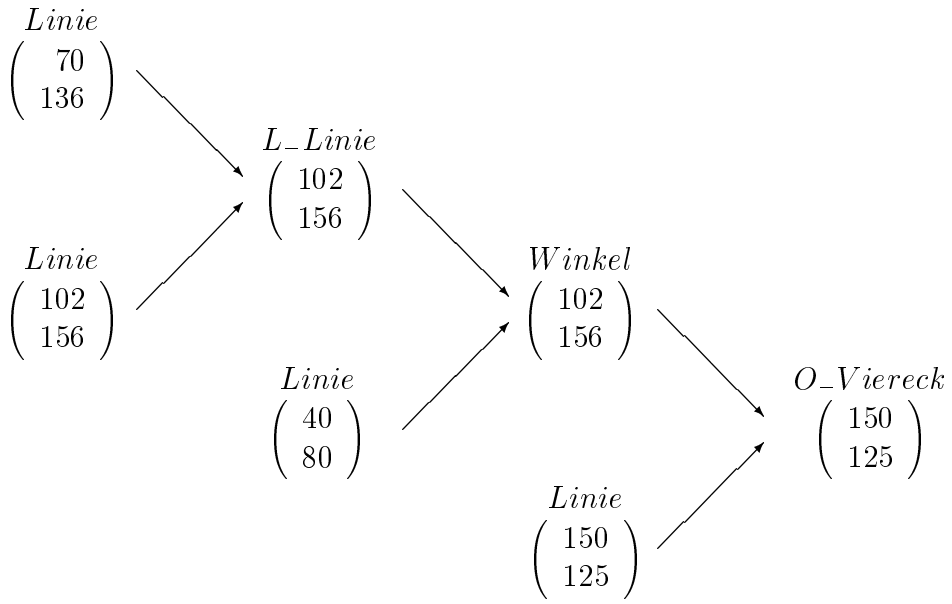


Figure 5.1: **Beispiel eines Reduktionsgraphen**
Veranschaulichung der Beispielreduktion aus Abschnitt 2.2.3. Zur Unterscheidung
der Instanzen genügen der Symbolteil und die Koordinaten des ersten Punktes.

S in der rechten Seite Σ (Ausgangsseite) vorkommt. Kanten von Symbolen zu Symbolen oder Kanten von Produktionen zu Produktionen gibt es nicht. Den Kanten werden Vielfachheiten zugeordnet, die angeben wie oft das Symbol in dem Wort steht. Sie werden dann entsprechend oft im gezeichneten Produktionsnetz dargestellt.

Man wird solch ein Produktionsnetz immer als erstes anzeichnen, wenn man jemandem erklären will, wie eine konkrete KG funktioniert. Die Abbildung 5.2 zeigt das Produktionsnetz der kleinen Beispielgrammatik aus dem Abschnitt 2.2.3. Da dieser gerichtete Graph zyklenfrei ist, ist die KG nicht rekursiv. Für solche Grammatiken kann dem Alphabet eine Ordnung zugeordnet werden:

$$\begin{aligned} \mathcal{O}: V &\longrightarrow \{0,1,2,...\} \\ s &\longmapsto \text{halbe Länge des längsten Weges im Produktionsnetz nach } s \end{aligned}$$

Für RBB-Grammatiken ist die Ordnung der Terminale immer Null, weil deren linke Seite ein Nichtterminal enthalten muß (nach Definition der KG) und außer diesem kein anderes Symbol enthalten kann (wegen der RBB-Eigenschaft). In der Beispielgrammatik aus Abschnitt 2.2.3 ist $\mathcal{O}(Linie) = 0$, $\mathcal{O}(L\_Linie) = 1$, $\mathcal{O}(Winkel) = 2$ und $\mathcal{O}(O\_Viereck) = 3$.

Wenn nun $\mathcal{O}_{max} = max_{S \in V}\mathcal{O}(S)$ die maximale Ordnung der Symbole einer nicht rekursiven KG ist und man den Algorithmus 2 aus dem Kapitel 3 in Anwendung auf diese KG betrachtet, so stellt man fest, daß für die Tiefe der Rekursion dort $\mathcal{O}_{max}$ als obere Schranke dienen kann. Daraus läßt sich dann auch eine polynomiale obere Schranke für den Rechenaufwand bestimmen. Für nicht rekursive BBB KGs liegt sie z. B. bei Polynomen der Ordnung $2^{\mathcal{O}_{max}}$. Im Gegensatz zu nicht rekursiven Stringgrammatiken, die nur eine endliche Sprache erzeugen können und daher komplexitätsmäßig im Bereich *constant time* liegen, können nicht rekursive KGs einen nicht endlichen Attributbereich haben, und stehen daher in dem Diagramm 2.4 zwischen den endlich attributierten und den monotonen KGs. Dem entsprechen die Komplexitätsstufen konstant $<$ polynomial $<$ über polynomial.

Man kann an den Kanten eines Produktionsnetzes informelle Hinweise auf die wichtigsten Teil-Prädikate, etwa Nachbarschaft oder Parallelität, anbringen, und so die Suggestivkraft einer solchen Zeichnung erhöhen. Insgesamt bleibt zu bemerken, daß graphische Veranschaulichungen in Bildern, insbesondere solche, die Graphen verwenden, große Vorteile bei der Verfahrens Konzeption, Entwicklung, Wartung und Dokumentation bieten, oft aber eben Dinge trivial erscheinen lassen, die gar nicht so einfach zu formalisieren und mithin zu programmieren sind. Insbesondere die Kombinatorik und mit ihr die Rechenkomplexität, die sich aus einem Konzept oder einer KG ergibt, wird in diesen Veranschaulichungen nicht so gut sichtbar.

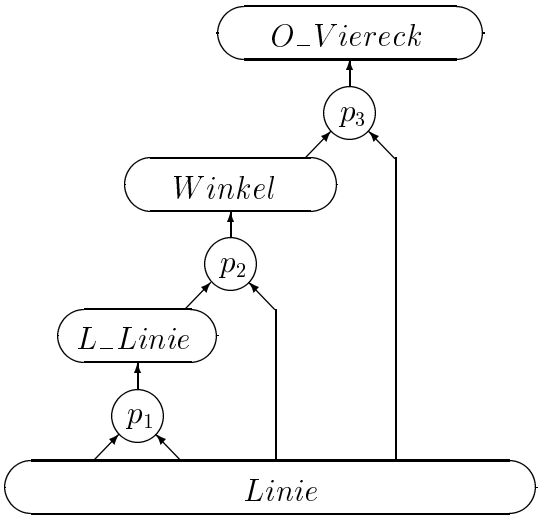


Figure 5.2: **Beispiel eines Produktionsnetzes**
Veranschaulichung der Beispiel KG aus Abschnitt 2.2.3 durch das Produktionsnetz

5.1.3 Die Dimension einer Grammatik

Die Betonung liegt auf den Prädikaten auf dem Attributraum, also auf den Konfigurationen, nicht auf der Konkatenation der Symbole. Dies ist in der Definition in Abschnitt 2.1.1 der KG, gegenüber der Rosenfeld/Andersonschen Definition der GRG, noch besonders hervorgehoben. Entsprechend dient als Attributraum auch nicht einfach der Z^n, der weder von der Anwendung sinnvoll noch von der Implementierung her machbar ist. Attribute, die Orte in einem Bild beschreiben, sind notwendig durch die Bildränder begrenzt. Ein Orientierungsattribut für Linien verlangt nach einer geschlossenen Topologie. Orte in der Szene können nicht mit beliebiger Präzision in beliebig großen Räumen effizient abgespeichert werden. Richtungsattribute im 3D Raum weisen die Topologie der Kugel auf. Die Dimensionen d_i der einzelnen Attributintervalle weisen üblicherweise die Werte 1, 2 oder 3 auf, so daß sie einzeln der menschlichen Anschauung zugänglich bleiben. Definiert man $d_{max} = max\{d_1, ..., d_n\}$, so kann man von einer d_{max} **dimensional attributierten** Grammatik reden um so zu klären, ob sie auf Strings, Bildern oder in der 3D Szene definiert ist.

In der Beispiel KG aus Abschnitt 2.2.3 kommen nur maximal zweidimensionale Attribute vor. Eine Grammatik mit solchem Attributraum ist nach dieser Definition also zweidimensional attributiert. Die in Abbildung 5.3 skizzierte Produktion setzt hingegen zwei mit jeweils vier Ortsattributen in einem begrenzten 3D Intervall (der Szene) beschriebene Vierecke zusammen, wenn deren Normalvek-

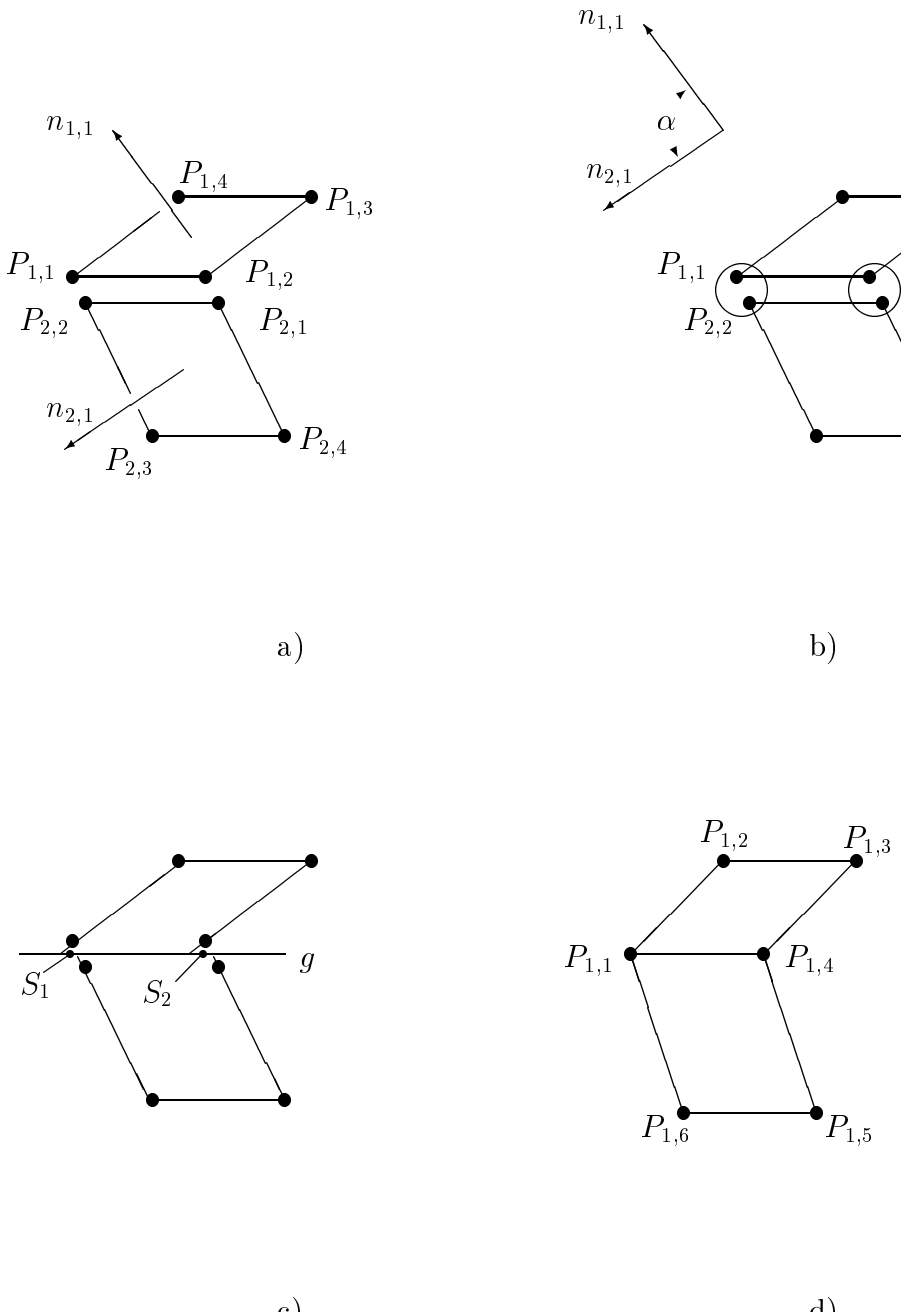


Figure 5.3: **Graphische Veranschaulichung einer 3D Produktion**
a) Ausgangskonfiguration sind zwei Symbole vom Typ 'Viereck';
b) die Punkte $P_{1,1}$) und $P_{2,2}$) und die Punkte $P_{1,2}$) und $P_{2,1}$)
sind benachbart, und der Winkel zwischen den Normalen ist etwa α;
c) auf der Schnittgeraden g der Ebenen werden durch Mittelung
die Werte der neuen Attribute bestimmt;
d) die neue Symbolkette besteht aus einem einzelnen 'Zweiflächner'
mit den dargestellten Attributwerten. 79

toren ungefähr im Winkel α zueinander stehen und die entsprechenden Punkte im 3D Raum benachbart sind. Eine KG, die solche 3D Ortskoordinaten als Attribute verwendet, heißt also dreidimensional attributiert. Sie dient eigentlich mehr der Szenenanalyse als der Bildverarbeitung, löst sich weitgehend von der Pixelstruktur, und orientiert sich in der Struktur ihres Attributraums an den zu suchenden Objekten. Das im Kapitel 6 vorgestellte Beispiel eines implementierten Verfahrens zur Detektion und Lokalisation bestimmter Fahrzeuge gehört zu dieser Kategorie. Hier dienen zwar gerasterte Bilder als Ausgangsdaten, aber außer den zweidimensionalen Ortsattributen im Bild gibt es auch dreidimensionale Ortsattribute für Instanzen in der Szene.

5.2 Hilfsstrukturen zum Wissenserwerb

Die Erstellung eines syntaktischen Verfahrens zur Lösung von Identifikations-, Klassifikations-, Lokalisations- oder Detektionsaufgaben aus Beispieldatensätzen wird in der Literatur - etwa in [GONZ] oder [FU-82] - unter dem Begriff *grammar inference* diskutiert. Hier soll auf derartige vollautomatische Lernmethoden nicht eingegangen werden. Die Entwicklung brauchbarer Verfahren bleibt wohl noch auf lange Zeit dem menschlichen Programmierer vorbehalten. Das schließt gewisse Systematiken und Standardbegriffsbildungen nicht aus. Unter den speziellen Werkzeugen, die man sich zur Erzeugung einer KG vorteilhafterweise zurechtlegt, stellen insbesondere die Nachbarschaftsgraphen eine zweckmäßige Hilfe dar. Sie werden im Abschnitt 5.2.1 erläutert. Oft ist explizites Wissen über die geometrische Gestalt des gesuchten Objekts und die möglichen Invarianzen, unter denen es erscheinen kann, vorhanden. Dann kann mit speziellen Produktionen ein Vergleich zwischen dem Modell und einer Konfiguration von instantiierten, möglichen Teilen durchgeführt werden. Diese in der Literatur unter dem Begriff *template matching* geführte Vorgehensweise wird im Abschnitt 5.2.2 diskutiert. Symmetrien in der Beschreibung der Teile eines komplexen Objektes können zu einer Kombinatorik der Möglichkeiten des Zusammensetzens führen. Darauf geht Abschnitt 5.2.3 ein. Der Abschnitt 5.2.4 zeigt auf, daß rekursive Produktionen bzgl. ihrer Semantik und Numerik problematisch sein können. Abschnitt 5.2.5 liefert hierzu für gewisse Fälle eine Lösung in Gestalt von Produktionen, die nicht auf Tupeln, sondern auf Mengen arbeiten. Dabei werden rekursive Berechnungen weitgehend vermieden.

5.2.1 Nachbarschaftsgraphen

Bei der Konzipierung einer KG können Nachbarschafts-, Teil-von- und andere wichtige Relationen vorteilhaft und übersichtlich durch einen **Nachbarschaftsgraphen** veranschaulicht werden. Das zu identifizierende Objekt wird stufen-

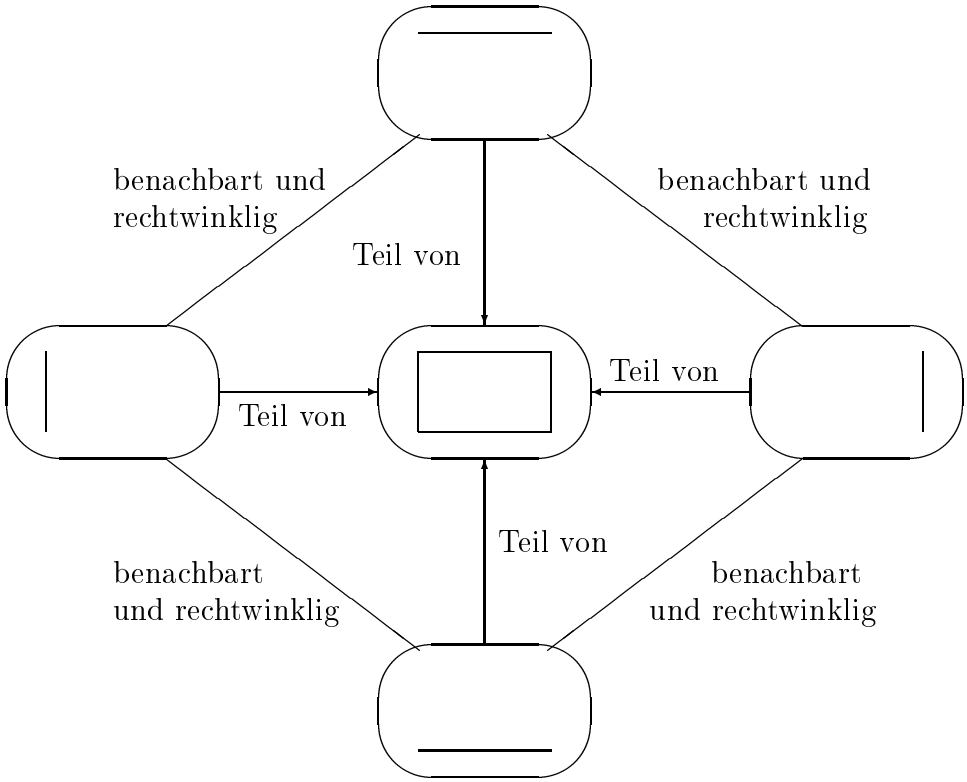


Figure 5.4: **Nachbarschaftsgraph eines Rechtecks**

weise in eine Hierarchie von prägnant erscheinenden Teilen zerlegt. Man benennt die Teile, und erhält so die Menge der Knoten K des Nachbarschaftsgraphen $\mathcal{N}$. Die geometrischen und insbesondere topologischen Relationen, die zwischen den einzelnen in dieser Hierarchie niedrigeren Teilen bestehen müssen, damit sie zu höheren Teilen der Hierarchie zusammenwachsen, werden jeweils an den Kanten notiert. Man fügt also zur Kantenmenge $E \subset K \times K$ des Graphen $\mathcal{N}$ eine solche Kante (n_1, n_2) hinzu, wenn eine solche Relation zwischen zwei Teilen besteht und im Erkennungsprozeß ausgenutzt werden soll. Zweckmäßigerweise markiert man die Kante entsprechend. Symmetrische Relationen, wie Nachbarschaft, Parallelität usw. führen natürlich zu ungerichteten Kanten. Dann ist mit (n_1, n_2) auch (n_2, n_1) automatisch in K. Besteht zwischen zwei Knoten n_1 und n_2 eine 'Teil von' Relation, die Teil der Hierarchie des Erkennungsprozesses werden soll, so wird die Kante (n_1, n_2) ebenfalls hinzugefügt und entsprechend markiert. Da diese Relation antisymmetrisch ist, ist es eine gerichtete Kante.

Abbildung 5.4 zeigt ein Beispiel. Es soll ein Rechteck gefunden werden, welches

in seine vier Seiten zerfällt. Als Knotenmenge wird also gesetzt

$$K = \{Rechteck, Seite_{links}, Seite_{rechts}, Seite_{oben}, Seite_{unten}\}$$

und als Relationen sollen Nachbarschaft und Rechtwinkligkeit ausgenutzt werden, wie in der Abbildung dargestellt.

Aus der Knotenmenge des Nachbarschaftsgraphen ergibt sich das Vokabular V der zu konstruierenden KG. Allerdings kann sie nicht direkt übernommen werden, da Teile des Objektes, wenn man ihre Position im Gesamtmuster ignoriert, ununterscheidbar werden können. Sie werden dann unter einem Symbol zusammengefaßt. Wann Teile voneinander ununterscheidbar sind, hängt von den zugelassenen Transformationen ab. Darauf geht der Abschnitt 5.2.2 noch ein. Im Beispiel könnte man setzen, daß die waagerechten Seiten nicht unterscheidbar sind, wenn die Position des Rechtecks unbestimmt ist. Dasselbe gilt für die senkrechten Seiten. Man wählt also zu diesen vier Objektteilen zwei Symbole. Da sie nicht weiter in Teile zerfallen, werden es Terminale. Im Beispiel bleibt nur noch ein Knoten übrig, nämlich das Rechteck selbst. Dieser Knoten wird Startsymbol und einziges Nichtterminal der Grammatik. Es gibt also auch erstmal nur eine Produktion. Sie setzt die vier Seiten auf einen Schlag zusammen. Die Attribute der Terminale ergeben sich aus den zur Verfügung stehenden Vorverarbeitungsverfahren, die Attribute des Startsymbols aus dem Objekt selbst und den zugelassenen Transformationen. Damit sind dann auch das Prädikat π und die Funktion ϕ festgelegt. Allerdings muß man noch achtgeben auf eventuell vorhandene Symmetrien in den betrachteten Attributräumen. Damit beschäftigt sich Abschnitt 5.2.3.

Daß die Erstellung eines Nachbarschaftsgraphen dem Benutzer weitgehende Freiheiten läßt, zeigt die in Abbildung 5.5 dargestellte alternative Zerlegung desselben Objektes 'Rechteck' unter Verwendung folgender Knoten:

$$\{Rechteck, Ecke_{l,o}, Ecke_{r,u}, Seite_l, Seite_r, Seite_o, Seite_u\}$$

Hier wird also der 'Umweg' über die nichtterminalen Ecken eingeschlagen. Das entspricht in etwa der Konstruktion in 2.3.1, in der beliebige RBB-Produktionen in BBB-Produktionen umgewandelt werden, durch Zuhilfenahme zusätzlicher Nichtterminale. Wenn man mit dem Algorithmus aus Abschnitt 3.3 arbeitet, ist die Aufzählung der Paare von Instanzen bestimmter Symbole zwecks Überprüfung auf Vorliegen einer BBB-Produktions Ausgangskonfiguration **quadratisch** in der Komplexität nach der Anzahl der Instanzen dieser Symbole. Nur wenn eine Ausgangskonfiguration vorliegt, muß weitergerechnet werden. Es besteht die Hoffnung, daß dies relativ selten ist. Nach den Faustregeln aus Abschnitt 5.3 sollte der Erwartungswert für die Anzahl der Partner zu einer Instanz konstant bleiben bei wachsender Größe der Datenbasis. Im vorliegenden Beispiel kann dies dann angenommen werden, wenn mit der Anzahl der Linien im Bild auch das Bild

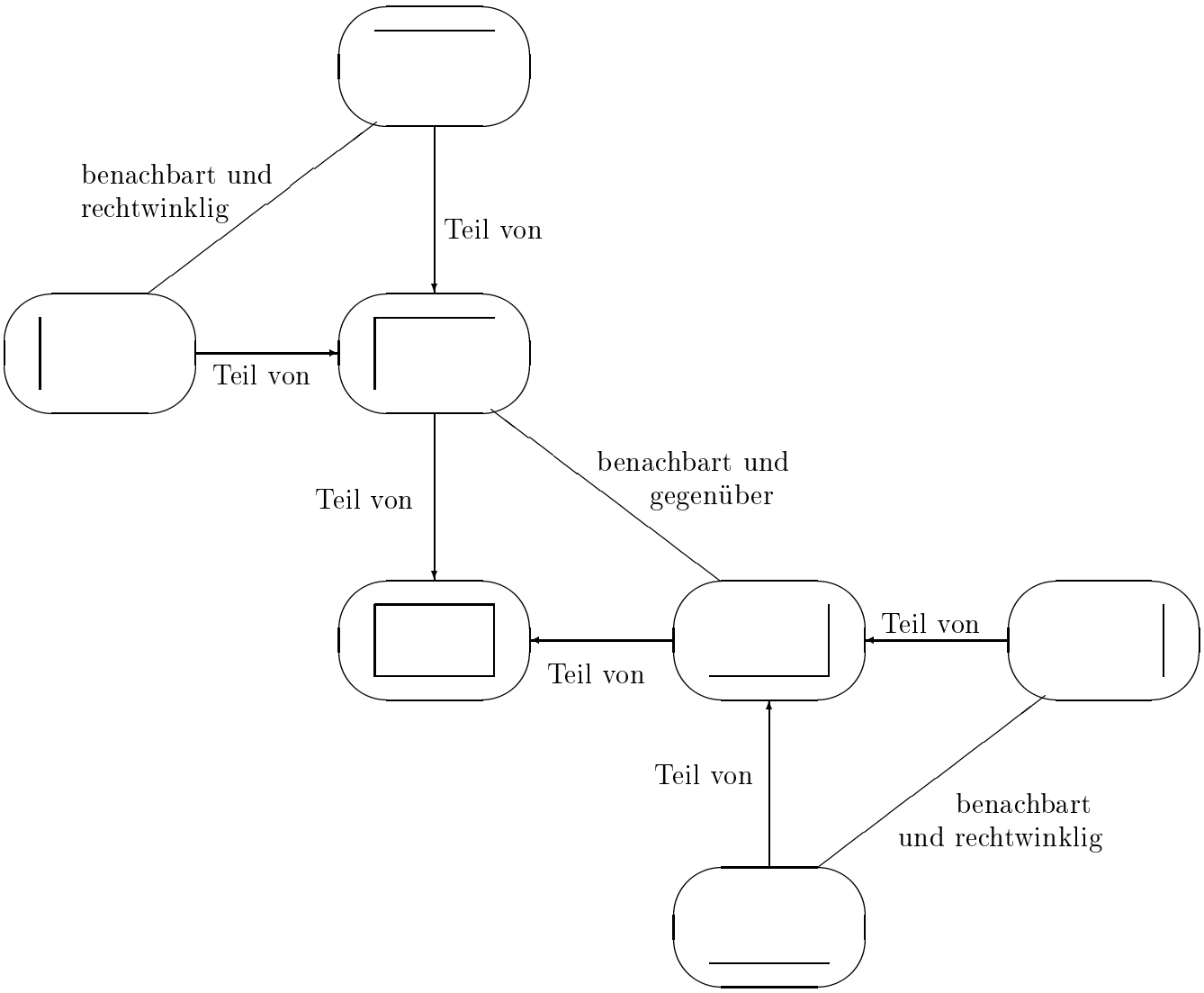


Figure 5.5: **Alternativer Nachbarschaftsgraph**

selber wächst, also die 'Dichte' der Linien pro Bildflächeneinheit gleich bleibt. In diesem Falle kommt man durch Einführung dieser in gewisser Weise 'künstlichen' Hierarchie, und damit des Zwischenschrittes über zusätzliche Nichtterminale, von einer Rechenkomplexität **vierter Ordnung** (für die Aufzählung aller Quadrupel als Kanditaten für die Ausgangskonfiguration) herunter auf quadratische Komplexität. Wenn allerdings mit der Anzahl der Linien das Bild nicht wächst, sondern die Linien immer dichter zu liegen kommen, dann steigt die Anzahl der benachbarten und rechtwinkligen Partner pro Linie eben auch an. Damit wächst die Zahl der Hilfsnichtterminale quadratisch mit der Anzahl der Terminale, und mithin die Anzahl der Startinstanzen dann auch wieder mit der Ordnung vier. Zwar mag das Nichtweiterverfolgen aussichtloser Paare und Tripel durchaus vernünftig sein, aber größenordnungsmäßig ist nichts gewonnen. Die Ordnung des Polynoms bleibt gleich.

Wie im Abschnitt 3.4 erläutert, kann mit Hilfe von assoziativ ansprechbaren Datenbasen die Komplexität von BBB-Produktionen von quadratisch auf **linear** gesenkt werden, wenn das Bedingungsprädikat dies zuläßt und die Partnermenge nicht mit der Datenbasis wächst. Wenn es also gelingt, für ein gesuchtes Objekt einen Nachbarschaftsgraphen aufzustellen, der diese Bedingungen erfüllt und den man nach der skizzierten Methode soweit durch Einführung von Hilfsnichtterminalen 'hierarchisieren' kann, daß nur noch jeweils zwei Teilobjekte zusammengesetzt werden, so kann dieses Objekt in linearer Komplexität identifiziert werden durch eine aus diesem Nachbarschaftsgraphen hergeleitete nicht rekursive BBB-KG und unter Zuhilfenahme eines hinreichend mächtigen Assoziativspeichers. Im Abschnitt 5.2.2 werden gewisse spezielle Produktionen untersucht, die auf explizitem Modellwissen über das gesuchte Objekt beruhen. Dann können unter Umständen durch assoziativen Zugriff die korrekten Ausgangskonfigurationen auch für rechte Seiten mit mehr als zwei Symbolen in linearer Komplexität gefunden werden.

Es bleibt zu bedenken, daß durch Einführung zusätzlicher Hilfsnichtterminale ein Produktionsnetz an Übersichtlichkeit verliert. Unter Umständen muß man sich recht willkürliche Bezeichnungen ausdenken, und die Objekte an unnatürlichen Stellen segmentieren. Besonders augenfällig kann das werden an 3D CAD Modellen mit vielen Flächen. Dann wächst die Anzahl der möglichen Zerlegungen, so daß ein Teil immer in zwei andere zerfällt, mit der Zahl der Flächen sehr rasch ins Unüberschaubare. Die allermeisten Zerlegungen spiegeln in keinster Weise mehr irgendwie die Gestalt des Objekts. Man erzeugt nur zunehmende Verwirrung und damit Instabilität, auch Mangel an Erklärbarkeit und also Wartbarkeit und Flexibilität.

5.2.2 Modellvergleichende Produktionen

Wenn von einem zu suchenden Objekt bestimmte geometrische Größen - etwa Abmaße und Winkel - oder Verhältnisse solcher Größen bekannt sind, so können damit Produktionen konstruiert werden, die eine Konfiguration aus Teilen des Objektes mit einem Modell des Objektes vergleichen. Das Modell kann zum Beispiel ein CAD Modell sein. Es besteht aus einer Liste von Punkten. Außerdem müssen die zulässigen Transformationen des Modells und die Korrespondenzen zwischen den Modellpunkten und den Attributen der Instanzen der Ausgangskonfiguration festgelegt werden. Bezüglich einer festzulegenden Metrik werden die Abstände zwischen den jeweils korrespondierenden Punkten berechnet. Sie müssen festzulegende Schwellwerte unterschreiten. Die Transformation wird so gewählt, daß ein festzulegendes Funktional dieser Abstände minimiert wird. Üblicherweise können die zulässigen Transformationen als Attributteilbereich aufgefaßt werden. Wenn dann garantiert ist, daß die Optimierung zu einem eindeutigen Ergebnis führt, so kann die optimale Transformation als Attributwert einer einzelnen Zielinstanz eingetragen werden. Die so entstehende Produktion ist also eine RBB-Produktion. Allerdings enthält die Ausgangskonfiguration i. a. mehr als zwei Instanzen. Insgesamt erhält man Verfahren, wie sie in der Musterkennungsliteratur als *template matching* geführt werden. Die folgende Definition präzisiert das:

Eine RBB-Produktion $p = ((s), \Sigma, \pi, \phi)$ heißt **modellvergleichend** genau dann, wenn

- es eine Liste $M = (p_1, ..., p_m)$ von Punkten in einem metrischen Raum D' gibt (genannt das *Modell*),
- eine Menge Tr von *zulässigen Transformationen* $t : D' \to D'$ festgelegt ist, und eine Kodierung des Transformationsraumes Tr in Komponenten des Attributbereiches D besteht,
- eine *Korrespondenz* Ko jedem Punkt des Modells eine Menge von Paaren von Indizes (j, i) zuordnet, dergestalt daß j eine Position in Σ ist, i eine Komponente des Attributbereichs D bezeichnet, und D_i eingebettet ist in D',
- ein Funktional F auf den metrischen Abständen zwischen den transformierten Modellpunkten $t(p_l)$ und den korrespondierenden Attributwerten der Ausgangskonfiguration definiert ist (also $F : D^k \times Tr \to R$), welches durch Variation von t eindeutig minimiert wird,
- ϕ jeder Ausgangskonfiguration die Kodierung eben dieser F minimierenden Transformation t_{min} zuordnet und

- π die Forderung enthält, daß alle diese Abstände festgelegte Schwellwerte unterschreiten.

Ein wesentlicher Vorteil von solchen modellvergleichenden Produktionen liegt darin, daß man nicht verlangen muß, daß die Ausgangskonfiguration vollständig vorliegt. Formal kann man dies erreichen, indem man eine Schwelle $k' < k = |\Sigma|$ setzt. Soviele Teile von Σ sollen mindestens gefunden werden, damit die Produktion noch durchgeführt wird. Man zählt dann alle mindestens k'-elementigen Teilworte durch sukzessives Herausnehmen von Symbolen auf. Durch Weglassen der entsprechenden Komponenten in π und ϕ entstehen jeweils neue Produktionen, die der KG hinzugefügt werden. Programmiertechnisch müssen diese Teilworte nicht aufgezählt werden.

Abbildung 5.6 verdeutlicht eine modellvergleichende Produktion an einem Beispiel. Als Zielinstanz ist wiederum das Rechteck gesetzt. Dabei findet folgende Reduktion statt:

$$\begin{pmatrix} \mathit{Linie} & \mathit{Linie} & \mathit{Linie} \\ \begin{pmatrix} 42 \\ 73 \end{pmatrix} & \begin{pmatrix} 53 \\ 85 \end{pmatrix} & \begin{pmatrix} 120 \\ 144 \end{pmatrix} \\ \begin{pmatrix} 86 \\ 29 \end{pmatrix} & \begin{pmatrix} 104 \\ 153 \end{pmatrix} & \begin{pmatrix} 152 \\ 100 \end{pmatrix} \end{pmatrix} \longmapsto \begin{pmatrix} \mathit{Rechteck}_{100,60} \\ \begin{pmatrix} 10 \\ 0 \end{pmatrix} \\ 45^o \end{pmatrix}$$

Für gewisse Transformationsmengen Tr kann die Menge der möglichen k-Tupel, bzw. k'-Teiltupel davon, wiederum durch assoziative Anfrage an die Datenbasis und logische Verknüpfungen so realisiert werden, daß ein lineares Vorgehen nach einer Warteschlange aus Verarbeitungselementen $VE = (I, p, j)$, wie sie in den Abschnitten 3.4 und 4.1 beschrieben sind, alle entsprechenden Ausgangskonfigurationen aufzählt. Dabei liefert das aus der Warteschlange entnommene Verarbeitungselement eine Ausgangsnäherung $t \in T$ für die Transformation, indem das Funktional F nur unter Berücksichtigung der Attributwerte der einen Instanz I an der $j-$ten Position in κ_a minimiert wird. Um jeden Punkt des so transformierten Modells herum kann dann ein Suchbereich konstruiert werden. Die Gestalt, Größe und Lage dieses Suchbereichs hängt von der Metrik, dem Schwellwert und vor allen Dingen auch von der zugelassenen Transformationsmenge Tr ab. Die Korrespondenz Ko des Punktes zu jedem Suchbereich liefert dann Mengen von Paaren von Indizes (i, j). Hier variiert i zwischen 1 und der Anzahl der Komponenten von D und j bezeichnet Positionen in der rechten Seite der modellvergleichenden Produktion p. Man erhält somit durch assoziative Zugriffe Instanzenmengen $K_{i,j}$, die an dieser Stelle von κ_a mit diesem Attribut in Frage kommen. Da die Instanz $\kappa_{a,j}$ i. a. mit mehreren Attributwerten in mehreren Suchbereichen liegen soll, müssen die entsprechenden Mengen geschnitten werden. Es ergeben sich die Kanditatenmengen $Kand_j = \bigcap_i K_{i,j}$. Sind mindestens

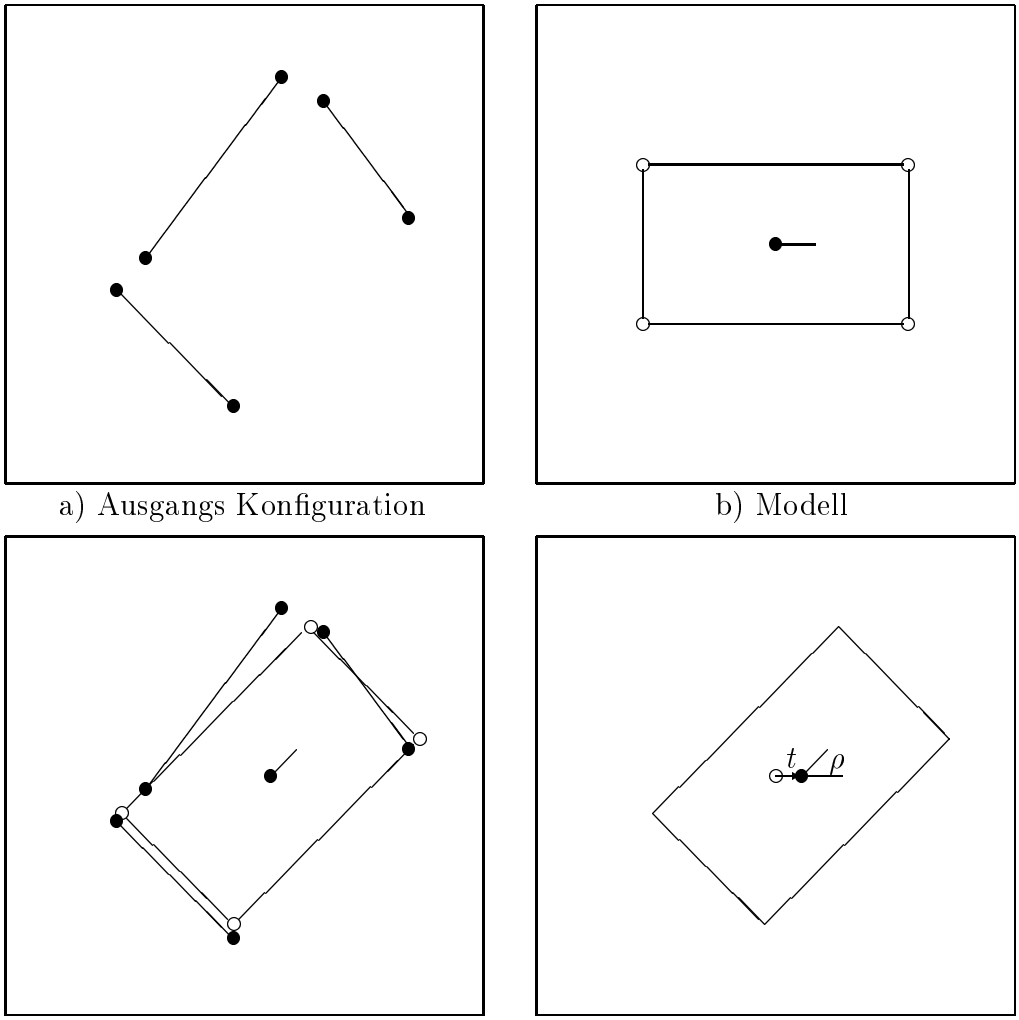


Figure 5.6: **Graphische Veranschaulichung einer modellvergleichenden Produktion**

k' von diesen Mengen nicht leer, so können zulässige Ausgangskonfigurationen gebildet werden. Natürlich sollte der Parser alle Kombinationen konstruieren. Die Anzahl der somit durchzuführenden Reduktionen ergibt sich also aus dem Produkt der Mächtigkeiten der nichtleeren $Kand_j$.

Abbildung 5.7 zeigt die Suchbereiche für die modellvergleichende Beispielproduktion. Die Rotation in der Ausgangsnäherung für die Transformation richtet sich nach der Ausrichtung der triggernden Linieninstanz I und der Position j, die beide mit dem Verarbeitungselement gegeben sind. Die Konfiguration in a) ergibt die in b), c) und d) dargestellten Suchbereiche. Das triggernde Element ist jeweils hervorgehoben. Die Translation wird so bestimmt, daß die Fehlerquadratsumme der beiden Endpunkte mit ihren korrespondierenden Modellpunkten minimiert wird. Die Abbildung zeigt, daß die konkrete Lage der Suchbereiche sich ändert, je nachdem welches Element triggert. Je schlechter die Ausgangsnäherung für die Transformation ist, z. B. weil die schmale Seite eines gestreckten Objektes die Rotation nur ungenau abschätzen läßt, desto größer müssen die Suchvolumina werden, desto größer werden die Kandidatenmengen. In solchen Fällen kann es helfen, wenn man durch assoziativen Zugriff, ausgehend von einem Verarbeitungselement, mit der schmalen Instanz zunächst eines von den langen Teilen sucht, und dann die Ausgangsnäherung für die assoziativen Zugriffe der eigentlichen modellvergleichenden Produktion nach diesem Element richtet. Die 3D Grammatik im Kapitel 6 verwendet in der modellvergleichenden Produktion zur Erzeugung der Startinstanzen eine solche Ausgangsnäherung. Daß diese Methode nicht immer zum Erfolg führen muß, zeigen die Suchbereiche in Abbildung 5.7c). Obwohl sie sich nach der längsten Linie ausrichten, führen sie nicht auf die korrekte Ausgangskonfiguration. Der Suchradius müßte hier also noch etwas größer gewählt werden.

Insgesamt ist es für modellvergleichende Produktionen von Vorteil, wenn die Transformationsmenge Tr klein gehalten werden kann. Dadurch ergeben sich kleinere Suchbereiche und damit Kanditatenmengen, und auch eine stabilere Bestimmung der optimalen Transformation t für die Zielinstanz. Es hat sich in der Arbeit mit 2D Produktionssytemen in der Praxis ergeben, daß modellvergleichende Produktionen gut und stabil arbeiten, wenn lediglich die Translationen zu berücksichtigen sind (etwa bei kartengestützter Luftbildanalyse oder Zeichenerkennung). Auch, wenn noch die Rotationen hinzukommen - etwa bei der modellgestützten Luftbildanalyse - kann man noch damit arbeiten. In der Transformationsmenge Tr, und damit in der Form der Suchbereiche, können sogar noch gewisse Verzerrungen, die das Objektiv erzeugt, berücksichtigt werden (siehe [MICH-89]).

In der allgemeineren Bildverarbeitung oder bei Aufnahmen aus Schrägsicht und tiefer Flughöhe sind aber in Tr auch Skalierungen und affine und perspektivische Verzerrungen aufzunehmen. Dann wird ein solcher assoziativer Zugriff unmöglich,

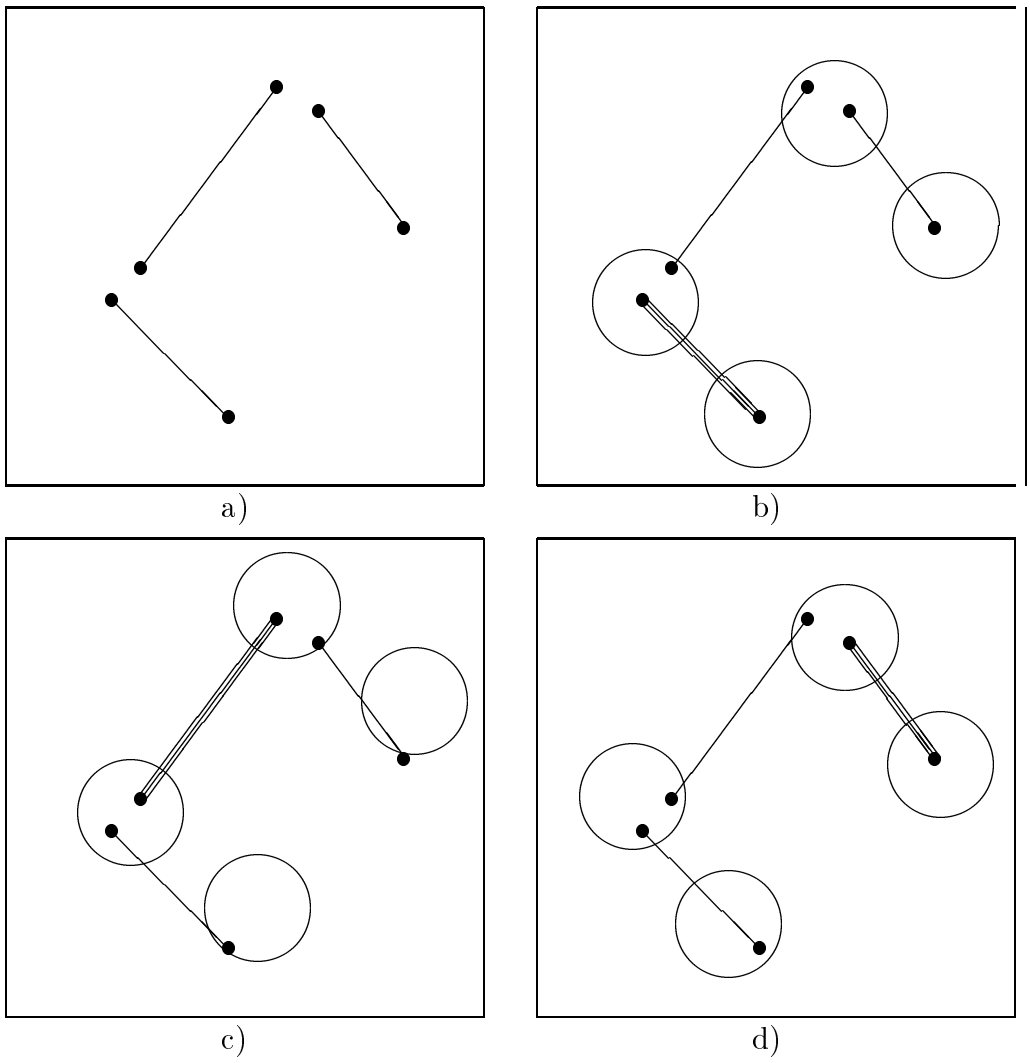


Figure 5.7: **Assoziativer Zugriff für eine modellvergleichende Produktion**

und man sieht sich gezwungen, eine Hierarchisierung des Nachbarschaftsgraphen wie im Abschnitt 5.2.1 durchzuführen, um von der Aufzählung der k-Tupel herunterzukommen auf ein lineares Vorgehen nach einer Warteschlange mit assoziativer Partnermenge.

5.2.3 Symmetrien im Attributraum und in der Konfiguration

Bereits in Abschnitt 2.1.3 wurde darauf hingewiesen, daß die dort ausgeführte Beispielproduktion zwei Instanzen des Symbols *Linie* nur dann zu einer Instanz des Symbols *Winkel* zusammensetzt, wenn sie mit den 'richtigen' Attributen benachbart sind. Die Veranschaulichung der Instanzen des Symbols *Linie* als Linie mit zwei gleichaussehenden Punkten birgt die Gefahr, daß der Anwender meint, mit dieser Produktion sein Konzept über die logische und geometrische Struktur von Linien und Winkeln programmiert zu haben. In seinen Konzepten über Linien und Winkel gibt es aber Symmetrien, die ihm aufgrund seiner Gestaltwahrnehmung so selbstverständlich sind, daß sie nicht bewußt werden. Dieselbe Linieninstanz kann auf zwei verschiedene Arten beschrieben sein. Man kann die Attributwerte der ersten beiden Ortsattribute vertauschen, ohne daß dadurch für den Anwender die Identität der Instanz angetastet ist. Für ihn bleibt es dieselbe Linie. Für die KG hingegen sind das zwei verschiedene Instanzen.

Da dies in der Praxis eine häufige Fehlerquelle ist, empfiehlt es sich hier, einen sauber definierten Begriffsapparat zu schaffen. Dergleichen Symmetrien sind häufig. Sie führen auch dazu, daß praxisrelevante KGs große Mengen von Produktionen beinhalten, die in natürlicher Weise, entsprechend der Symmetrien der zu erkennenden Objekte und Teilobjekte, zu *Familien* von Produktionen zusammengefaßt werden können, deren Mitglieder durch Umnumerierungen der Attribute in den Konfigurationen auseinander hervorgehen. Der Anwender braucht eigentlich nur eine Produktion zu definieren, und die Symmetrien seiner Objektkonzepte festzulegen. Dann kann die ganze Produktionenfamilie automatisch generiert werden. Die folgenden Definitionen liefern die Grundlage:

- Unter einer Rekombination ρ wird eine Abbildung verstanden, die eine Instanz eines Symbols in eine andere Instanz desselben Symbols überführt, indem sie die Attributwerte anders anordnet. Dies entspricht einer Permutation der Indizes der Attributintervalle. Also $\rho : \mathcal{U} \to \mathcal{U}$ heißt **Rekombination** genau dann, wenn es eine bijektive Abbildung $\sigma : \{1, ..., n\} \to \{1, ..., n\}$ gibt, mit

  $$\forall I \in \mathcal{U} : \; s(\rho(I)) = s(I) \; \wedge \; \forall i = 1, ..., n : \; a_i(\rho(I)) = a_{\sigma(i)}(I).$$

  Zur Erinnerung: n ist die Anzahl der Komponenten des Attributraums. Es

ist bekannt, daß es $n!$ solche Indexpermutationen gibt. Da sie die Rekombinationen eindeutig festlegen, ist dies auch die Anzahl der Rekombinationen. Bzgl. Nacheinanderausführung bilden diese Abbildungen eine endliche, im allgemeinen nichtabelsche Gruppe mit der Identität als neutralem Element. In der Algebra spricht man von der *symmetrischen* Gruppe n-ter Ordnung $\mathcal{S}_n$. Die übliche Zyklenschreibweise wird übernommen. So steht (1) für die Identität; $(2,3,5)$ bildet den zweiten Index auf den dritten, den dritten auf den fünften und den fünften wieder auf den zweiten Index ab, alle anderen bleiben identisch; $(1,4),(2,3)$ vertauscht den ersten mit dem vierten und den zweiten mit dem dritten Index.

- Der Grund, warum die Rekombination ρ von der Indexpermutation σ überhaupt unterschieden wird, liegt darin, daß erstens die Rekombinationen auf $\mathcal{U}$ und nicht auf D definiert sind, und zweitens zwar alle Permutationen automatisch wohldefiniert sind, nicht aber die zugehörigen Rekombinationen. Es kann sein, daß es Instanzen I gibt, für die $a_{\sigma^{-1}(i)}(I)$ nicht in D_i ist. Die zugehörige Rekombination von I ist dann *nicht wohldefiniert*, sie ist keine Instanz der KG mehr:

$$\rho \begin{pmatrix} s(I) \\ a_1(I) \\ \vdots \\ a_i(I) \\ \vdots \\ a_n(I) \end{pmatrix} = \begin{pmatrix} s(I) & \\ a_{\sigma^{-1}(1)}(I) & \\ \vdots & \\ a_{\sigma^{-1}(i)}(I) & \overset{!}{\notin} D_i \\ \vdots & \\ a_{\sigma^{-1}(n)}(I) & \end{pmatrix} \notin \mathcal{U}$$

  Um das auszuschließen wird definiert: **Zulässig** ist eine Rekombinationen ρ genau dann, wenn für die zugehörige Permutation σ

$$\forall i = 1, ..., n: \; D_i = D_{\sigma(i)}$$

  gilt. Innerhalb der Permutationsgruppe bilden die Permutationen zu diesen zulässigen Rekombinationen wegen der Reflexivität und Assoziativität der Gleichheit eine Untergruppe S_{zul}.

- Es wird definiert: Jede Untergruppe S der zulässigen Rekombinationen S_{zul} der Attribute zu einem Symbol ist eine **Symmetrie**. Hier liegt ein wesentlicher Punkt für eine korrekte Semantik zu einer KG: Oft sind dem Anwender die Symmetrien seiner Objektkonzepte nicht bewußt. Er wird sie also bei der Konstruktion der KG nicht angeben. Dann wird der entsprechende Parser nicht das leisten, was beabsichtigt war. Der Attributraum der semantisch korrekten KG ist modulo der zugelassenen Symmetrien aufzufassen. Wenn also eine Instanz I durch Anwendung einer zugelassenen Rekombination auf den Attributraum übergeht in eine Instanz I',

so sind I und I' als identisch zu betrachten. Sie liegen in einer Restklasse. Nur ein Repräsentant einer solchen Restklasse sollte in der Datenbasis gehalten werden. Es ist von Vorteil, wenn man in der Praxis eine eindeutige kanonische Darstellung jeder Restklasse angeben kann und zur Abspeicherung verwendet. Z. B. : 'Ein Winkel wird so abgespeichert, daß es vom Punkt 1 zum Scheitelpunkt zum Punkt 3 linksherum geht (mathematisch positiv drehend)'.

- Entsprechend der Symmetrie in den zu formalisierenden Objektkonzepten gibt es **Kombinationen** zu Produktionen. Ist eine Produktion p gegeben, und beinhalten die Objektkonzepte zu den Symbolen in der rechten Seite Σ Symmetrien, so entsteht, durch Anwendung der zu den Rekombinationen gehörigen Permutationen auf die *Indizes der Attribute* in dem Prädikat und in der Funktion, eine neue Produktion: Aus

$$\Sigma \xrightarrow{\pi \begin{pmatrix} a_{\Sigma_1,1}, ..., a_{\Sigma_1,n} \\ \vdots \\ a_{\Sigma_k,1}, ..., a_{\Sigma_k,n} \end{pmatrix}} \Lambda$$

$$\begin{array}{c} a_{\Lambda_1,1} = \phi_{1,1}[(a_{\Sigma_1,1}, ..., a_{\Sigma_1,n}), ..., (a_{\Sigma_k,1}, ..., a_{\Sigma_k,n})] \\ \vdots \\ a_{\Lambda_1,n} = \phi_{1,n}[(a_{\Sigma_1,1}, ..., a_{\Sigma_1,n}), ..., (a_{\Sigma_k,1}, ..., a_{\Sigma_k,n})] \\ \\ \vdots \\ \\ a_{\Lambda_j,1} = \phi_{j,1}[(a_{\Sigma_1,1}, ..., a_{\Sigma_1,n}), ..., (a_{\Sigma_k,1}, ..., a_{\Sigma_k,n})] \\ \vdots \\ a_{\Lambda_j,n} = \phi_{j,n}[(a_{\Sigma_1,1}, ..., a_{\Sigma_1,n}), ..., (a_{\Sigma_k,1}, ..., a_{\Sigma_k,n})] \end{array}$$

wird durch Anwendung der Permutationen $\sigma_1, ..., \sigma_k$ innerhalb der einzelnen Zeilen der Argumente

$$\Sigma \xrightarrow{\pi \begin{pmatrix} a_{\Sigma_1,\sigma_1^{-1}(1)}, ..., a_{\Sigma_1,\sigma_1^{-1}(n)} \\ \vdots \\ a_{\Sigma_k,\sigma_k^{-1}(1)}, ..., a_{\Sigma_k,\sigma_k^{-1}(n)} \end{pmatrix}} \Lambda$$

$$\begin{array}{l} a_{\Lambda_1,1} = \phi_{1,1}[(a_{\Sigma_1,\sigma_1^{-1}(1)}, ..., a_{\Sigma_1,\sigma_1^{-1}(n)}), ..., (a_{\Sigma_k,\sigma_k^{-1}(1)}, ..., a_{\Sigma_k,\sigma_k^{-1}(n)})] \\ \vdots \\ a_{\Lambda_1,n} = \phi_{1,n}[(a_{\Sigma_1,\sigma_1^{-1}(1)}, ..., a_{\Sigma_1,\sigma_1^{-1}(n)}), ..., (a_{\Sigma_k,\sigma_k^{-1}(1)}, ..., a_{\Sigma_k,\sigma_k^{-1}(n)})] \\ \\ \vdots \\ \\ a_{\Lambda_j,1} = \phi_{j,1}[(a_{\Sigma_1,\sigma_k^{-1}(1)}, ..., a_{\Sigma_1,\sigma_1^{-1}(n)}), ..., (a_{\Sigma_k,\sigma_k^{-1}(1)}, ..., a_{\Sigma_k,\sigma_k^{-1}(n)})] \\ \vdots \\ a_{\Lambda_j,n} = \phi_{j,n}[(a_{\Sigma_1,\sigma_1^{-1}(1)}, ..., a_{\Sigma_1,\sigma_1^{-1}(n)}), ..., (a_{\Sigma_k,\sigma_k^{-1}(1)}, ..., a_{\Sigma_k,\sigma_k^{-1}(n)})] \end{array}$$

- Aus einer Produktion p wird so eine **Familie** von Kombinationen von p

$$F(p) \stackrel{def}{=} \{p_{\sigma_1,...,\sigma_k};\ (\sigma_1, ..., \sigma_k) \in \prod_{i=1}^{k} S_i\},$$

  die alle in die Grammatik aufgenommen werden müssen, damit die Semantik dieser Grammatik mit den Konzepten des Anwenders konsistent ist.

Sind also die Teile, aus denen ein Objekt besteht, sehr symmetrisch, so bekommt man Probleme mit der Anzahl der erforderlichen Produktionen. Es gibt eben sehr viele Möglichkeiten, wie es sich zusammensetzen läßt. Eine zweiflächige Struktur, wie in der Abbildung 5.3 beispielhaft vor Augen geführt, mag bestehen aus zwei Quadraten, die jeweils mit ihren vier Endpunkten im Raum attributiert sind. Zulässig wären im Prinzip in diesem Attributraum alle 24 Permutationen der 3D Attribute. Man wird aber bei diesem Konzept nur die 4 echten Rotationen (1), (1,2,3,4), (1,3)(2,4) und (1,4,3,2) zulassen. Die Spiegelungen und die 'kreuzweisen' Vertauschungen entsprechen nicht der Symmetrie von quadratischen Oberflächen im Raum (mit Innen- und Außenseite). Die Familie zu dieser Produktion hat also 16 Elemente. Es wird noch unangenehmer, wenn man etwa einen Würfel aus sechs Quadraten auf einen Schlag durch Modellvergleich zusammensetzen wollte. Die Anzahl der durch diese Symmetrie der 3D Quadrate gegebenen Möglichkeiten, wie die Ausgangskonfiguration für einen Würfel im Detail ausfällt, ist $4^6 = 4096$.

Immer, wenn man auf solche Effekte stößt, sollte man noch einmal darüber nachdenken, ob diese Kombinationen wirklich gemeint waren, in dem Sinne daß sie mögliche Fälle in der Lage von Objekten zueinander im Raum beschreiben, oder ob die Kombinatorik nicht eher durch eine ungeschickte Attributierung selbst erzeugt wurde. Im Zweifel hilft eine Besinnung auf die wertheimerschen Gestaltprinzipien (etwa in Gestalt von [WERT]). Kann z. B. ein Quadrat mit weniger Information beschrieben werden als mit vier 3D Koordinaten (also 12 Zahlen)? Bei so symmetrischen Objekten ist das i. a. möglich: Verwendet man etwa ein 3D Attribut für den Schwerpunkt, ein 3D Richtungsattribut für die Oberflächennormale, ein Attribut mit geschlossener Topologie zwischen 0 und $\frac{\pi}{2}$ für den Drehwinkel um die Oberflächennormale und eine Kantenlänge, so erhält man ein 3D ein 2D und zwei 1D Attribute, insgesamt also sieben Zahlen. Keine Rekombination ist mehr zulässig. Sowohl die zweiflächige Struktur als auch der Würfel können aus so beschriebenen Instanzen jeweils durch eine einzige Produktion zusammengesetzt werden.

Bisher hat sich dieser Abschnitt mit den Symmetrien im Attributraum der einzelnen Instanzen der Ausgangskonfiguration beschäftigt. Betrachtet man aber die im Abschnitt 2.1.3 angeführte Beispielproduktion zum Zusammensetzen von Instanzen des Symbols *Winkel* aus Instanzen des Symbols *Linie*, so fällt auf, daß auch die Zielinstanz eine Symmetrie hat. Man kann die Vertauschung der ersten 2D Koordinate mit der dritten 2D Koordinate zulassen. Diese Symmetrie spiegelt sich wider in einer Symmetrie der Augangskonfiguration. Das Prädikat hängt nicht von der Reihenfolge, in der die Instanzen in der Ausgangskonfiguration stehen, ab; die Funktion transportiert eine Vertauschung der Instanzen in der Ausgangskonfiguration in eine Vertauschung der Attributwerte in der Zielinstanz. Da diese Vertauschung zugelassen wird, braucht man beim Parsen nicht alle geordneten Paare von Linieninstanzen durchzutesten. Es genügen die ungeordneten. Der Gewinn in der Rechenkomplexität ist immerhin etwa ein Faktor 2. Allerdings gibt es auch einen Nachteil: Der Parser sollte die Zielinstanz in die Datenbasis nur dann eintragen, wenn noch keine identische Instanz vorliegt. Wenn es eine kanonische Form gibt, rechnet er die zu erzeugende Instanz in diese Form um, prüft - unter Zuhilfenahme eines assoziativen Zugriffs - ob sie schon vorhanden ist, und fügt sie, wenn das nicht der Fall ist, hinzu. Wenn es keine kanonische Form gibt, muß er *alle* zugelassenen Rekombinationen der Instanz berechnen und für jede eine solche Prüfung vornehmen. Nur wenn gar kein Repräsentant der Restklasse modulo der zugelassenen Rekombinationen vorliegt, darf die neue Instanz eingetragen werden.

5.2.4 Schwierigkeiten mit rekursiven Produktionen

Die Produktion p_1 (Linienverlängerung) in der kleinen KG aus Abschnitt 2.2.3 ist ein typisches Beispiel für Produktionen, die eine für die Mustererkennung oft sehr wichtige Aufgabe übernehmen, die man etwas unscharf mit den Stichworten *Datenreduktion* oder *Vereinfachung* umschreiben kann. Hier geht es darum, die Beschreibung eines Konturs egments, das durch eine Instanz beschrieben werden kann, aber durch zwei Instanzen beschrieben ist, zu vereinfachen. Die Produktion reduziert zwei hinreichend kollineare, überlappende Linieninstanzen zu einer Linieninstanz, indem sie die als Störung anzusehenden inneren Punkte löscht, und nur die äußeren Konturenden übernimmt. Dadurch tritt, wie die Abbildung 2.3 zeigt, die Gestalt der Kontur deutlicher hervor.

Wenn man in der Zielkonfiguration hier dasselbe Symbol verwendet wie in der Ausgangskonfiguration, so erhält man eine rekursiv anwendbare Produktion. An sich ist gerade in dieser Möglichkeit rekursiver Anwendung von Produktionen die eigentliche Stärke syntaktischer Verfahren zu sehen. Andererseits ist auch hier wieder Vorsicht geboten. Man sollte nur solche Produktionen benutzen, die semantisch korrekt sind, also das realisieren, was man gemeint hat. Abbildung 5.8 zeigt eine Menge von Linieninstanzen, und was eine rekursive Version der Produktion p_1 im Endeffekt daraus machen würde.

Es hat sich in der praktischen Arbeit am FIM mit solchen Produktionen herausgestellt, daß viele solche bizarren 'Driftphänomene' ganz plötzlich auftreten können. Das erste Prinzip 'gracefull degradation' siehe [MARR] (Seite 106) ist verletzt. Betreibt man etwa Luftbildanalyse mit einer solchen rekursiven Linienverlängerung, und liegen die engsten parallelen Strukturen (etwa die Furchen gepflügter Felder) etwas weiter auseinander als der Schwellwert für Kollinearität, so mag das Verfahren ausgezeichnet funktionieren. Steigt man aber dann mit dem Sensor etwas in der Flughöhe oder wählt den Kollinearitätsparameter nur ein wenig großzügiger, so entstehen plötzlich sehr tiefe Reduktionsbäume, und die dabei reduzierten Instanzen driften *quer* zur Kontur über weite Bildbereiche. Es gibt dann immer auch Probleme mit dem Speicher und der Rechenzeit. Angesichts solcher Vorkommnisse kann vor rekursiven Verfahren nicht genug gewarnt werden.

Hinzu kommt, daß die KGs in der vorliegenden Form recht grob gerasterte Attribute verwenden, die eigentlich kontinuierliche Werte codieren. Es wird also bei der Bestimmung der Funktionswerte für die Zielkonfiguration im Laufe einer Reduktion immer wieder gerundet. Damit empfiehlt sich eine numerische Analyse oder doch wenigstens Abschätzung des Verfahrens. Die höchstzulässige Reduktionstiefe ergibt sich aus den möglichen oder anzunehmenden Rundungsfehlern. Sie darf in solchen Fällen nicht unbegrenzt sein.

Hier wird ein Dilemma des strukturorientierten Computersehens deutlich: Ein-

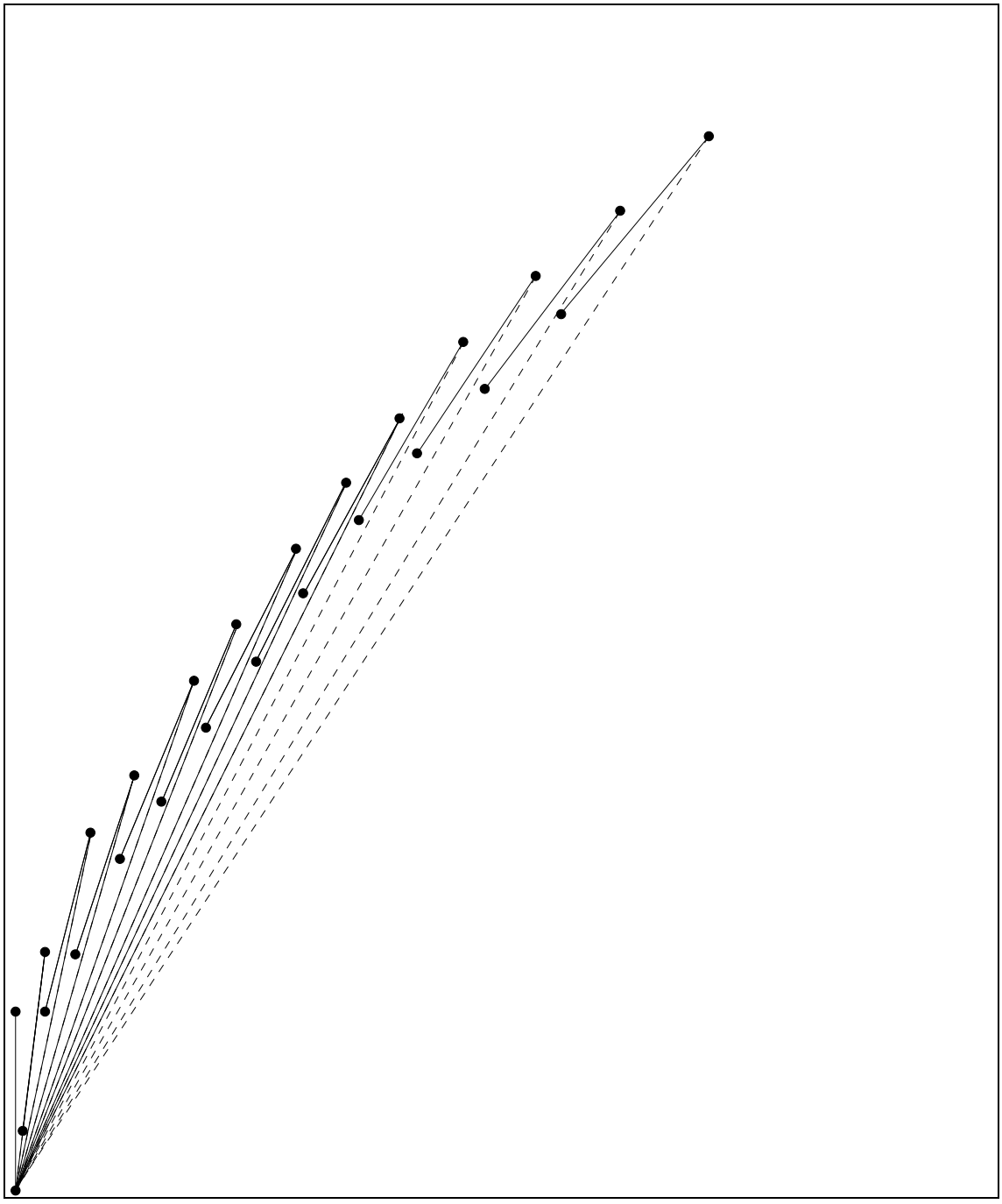


Figure 5.8: **Semantische Inkorrektheit einer rekursiven Produktion**
Im Verlaufe der schrittweisen Konturverlängerung
treten immer stärkere Abweichungen auf.

erseits sollte aus den oben dargelegten Gründen die Reduktion nicht tiefer sein als das Produktionsnetz. Andererseits steht man aber gleich beim ersten Schritt nach der Extraktion der Terminale - nämlich normalerweise der Konturverfolgung - vor Aufgaben offensichtlich serieller Natur. Es stellt sich eine umgangsprachlich etwa wie folgt zu formulierende Aufgabe: 'Verfolge und mittele die Kontur auf geradem Wege solange, bis sie ungebührlich von der Gerade abweicht.'

Will man hier streng innerhalb des in dieser Arbeit gegeben KG-Ansatzes bleiben, so bietet sich folgende Lösung an: Die Gerade ist zu bestimmen durch Fehlerquadratsummenminimierung der beteiligten Punkte. Vorzuziehen ist eine rotationsinvariante Form, also eine Minimierung der Summe der Fehlerquadrate in X- und Y-Richtung. Das heißt, man macht eine Regressionsrechnung. Solche Verfahren können iterativ durchgeführt werden. Man speichert die Koordinatensummen in X- und Y-Richtung, die drei quadratischen Summen (in X^2, $X \times Y$ und Y^2) und die Anzahl der beteiligten Punkte als Attributwerte ab. Liegen zwei solche, unter Umständen bereits mehrfach verlängerten, Linien hinreichend kollinear und überlappend, so werden die Summen und die Anzahl jeweils addiert. Durch Lösen des korrespondierenden kleinen Gleichungssystems gibt es eine neue Regressionsgerade. Senkrecht auf diese Gerade projiziert man die beiden äußeren Punkte der Ausgangskonfiguration, und erhält so die Endpunkte der neuen verlängerten Linie, in die alle vorgehenden Terminale gleichberechtigt eingeflossen sind. Soll eine Drift, wie in Abbildung 5.8, vermieden werden, so kann man im Prädikat zusätzlich verlangen, daß die senkrechten Abstandsquadrate aller vier Punkte von der Regressionsgeraden einen Schwellwert unterschreiten. Numerische Probleme sind ausgeschlossen, weil in den entscheidenden Berechnungen (den Summierungen) nicht gerundet wird. Man benötigt allerdings im Prinzip unendliche und in der Praxis sehr große Attributintervalle für die Quadratsummenattribute. Diese Attribute kodieren keine dem Benutzer anschaulich brauchbare Größe. Daher lehnt der Verfasser solch ein Verfahren ab. Es entspricht wohl den Buchstaben des verfolgten Ansatzes, nicht aber seinem Geist.

5.2.5 Mengen als Konfigurationen

Für die Konturverfolgung (insbesondere die geradlinige Linienverlängerung) empfiehlt es sich, von der auf binärbaumbildende Grammatiken fixierten Sichtweise abzuweichen. Die Regressionsgerade ist invariant gegenüber Permutationen der Ausgangskonfiguration, da nur die Koordinaten- und Koordinaten-Quadrat-*Summen* und die Gesamtanzahl in die Berechnung eingehen. Bezüglich einer so auf einer *Menge* von Linienelementen definierten Gerade lassen sich auch die Prädikate für 'überlappend' und 'kollinear' entscheiden, ohne daß diese Entscheidung von einer konkreten Aufzählungsreihenfolge dieser Menge abhinge. Semantisch sauber ist es also, in solchen Fällen statt des Tupels von Instanzen eine

Instanzenmenge als Ausgangskonfiguration zu verwenden. So vermeidet man auch die Rekursion in der Berechnung. Allerdings verläßt man damit das Gebiet der Koordinaten Grammatiken. Es wird sogar fragwürdig, ob solche Verfahren überhaupt noch als 'syntaktisch' anzusehen sind. Die Konturverfolgung liegt eben mehr im Grenzgebiet zwischen Vorverarbeitung und struktureller Analyse.

Es gibt ein ernstes kombinatorisches Problem mit Mengen als Ausgangskonfigurationen: Es ist relativ einfach, alle geordneten Paare einer Menge von Linien aufzuzählen, um ein Prädikat zu testen. Diese Menge wächst nur quadratisch mit der Zahl der Linien. Es ist aber für größere Linienmengen unmöglich, alle Teilmengen mit mehr als einem Element aufzulisten, um sie auf Überlappung und Kollinearität zu testen. Diese Aufgabe ist von exponentieller Komplexität. Produktionen mit Mengen als Ausgangskonfiguration können in der Präsenz größerer Datenmengen also nur für solche Bedingungsprädikate operationell durchgeführt werden, die nur für sehr wenige der vielen möglichen Untermengen erfüllt sind. Nachzuweisen ist, daß die Anzahl dieser Untermengen nicht in problematischer Weise mit der Zahl der Linien wächst.

Dazu sollte man sich wieder auf das besinnen, was eigentlich gemeint war. In diesem Falle war die Intention, im Sinne des Wertheimerschen Gestaltbegriffs durch maximale Datenreduktion und Vereinfachung die prägnanteste Beschreibung zu gewinnen. Man interessiert sich also gar nicht für Potenzmengen. Nur die bezüglich der Untermengenordnungsrelation maximalen Mengen sind interesant, also jene verlängerten Linien, die sich *nicht* weiter verlängern lassen. Wenn man das Bedingungsprädikat so faßt, ist die bösartige Kombinatorik, die aus der Potenzmengenstruktur resultierte, u. U. schon gebrochen. Allerdings verlangt ein solches Prädikat globale Konsistenz. Ein Parser kann in diesem Falle, wenn er einmal eine Menge L gefunden hat, die kollinear und überlappend ist, alle Untermengen $K \subset L$ aus seiner Aufzählung der Potenzmenge für die Verifikation dieser Ausgangskonfiguration streichen. Sie verletzen die zusätzliche globale Konsistenzbedingung der Maximalität bzgl. $\subset$.

Damit reduziert sich die zunächst böse erscheinende Suche im Potenzmengenraum der Linieninstanzen auf die Suche nach möglichst großen Mengen, die solch ein Prädikat erfüllen, und dabei kann der assoziative Speicherzugriff wertvolle Hilfe leisten. Abbildung 5.9 zeigt eine Instanz I des Symbols *Linie* im Bild, und einen zugehörigen möglichen Suchbereich zur Linienverlängerung. Die Breite dieses Bereichs entspricht der Toleranz des Prädikats 'kollinear'. Wenn es also Mengen gibt, die die Instanz I verlängern, dann müssen sie andere Instanzen mit ähnlicher Orientierung wie I enthalten, die einen Endpunkt in diesem Bereich haben. Man macht den Bereich möglichst lang, um viele Instanzen einzubeziehen. Natürlich ist nicht gesichert, daß alle Instanzen, die mit einem Endpunkt in einem solchen Suchbereich liegen und hinreichend ähnlich orientiert sind, wirklich zu einer Linienverlängerung beitragen können. Bedingungsprädikat π und Attributberech-

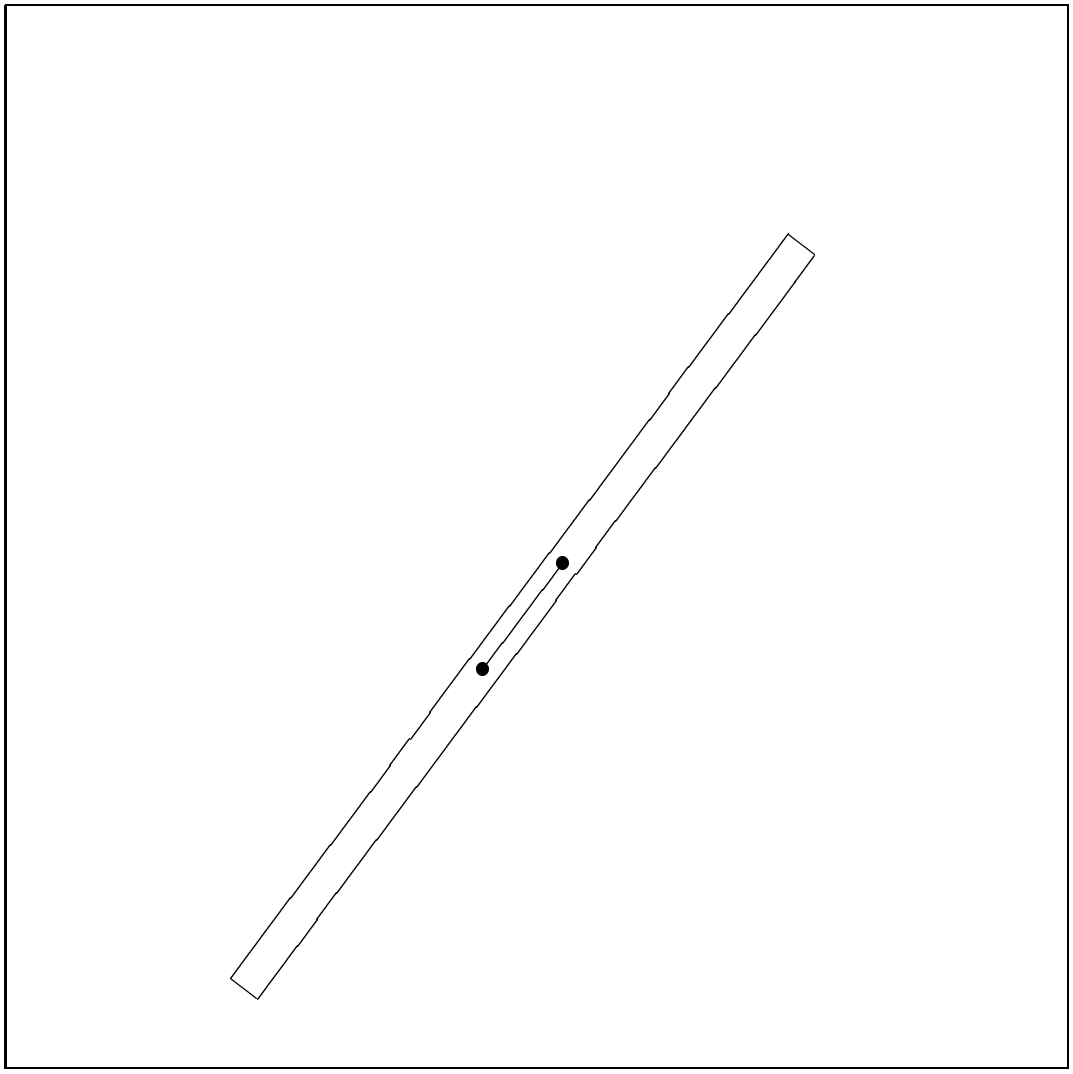


Figure 5.9: **Suchbereich zur Linienverlängerung**
Kandidatenmengenbildung durch assoziativen Zugriff auf alle Linien
mit einem Endpunkt im Suchbereich und etwa korrekter Orientierung

nungsfunktion ϕ der entstehenden Produktion gibt man am besten algorithmisch an:

```
PROCEDURE prolong(  VAR      line_set   :  Set of Instances;
                    VAR    L_line_set   :  Set of Instances );
Begin
   calculate_regression( line_set );
   While not col_set( line_set ) do
      line_set := line_set \ worst_element( line_set );
      calculate_regression( line_set );
   end While;
   L_line_set := overlapping_components( line_set );
end;
```

Die
dabei aufgerufenen Funktionen *col_set*, *worst_element*, *calculate_regression* und *overlapping_components* sind im Anhang B dokumentiert. Die Prozedur *prolong* kann nun zur Niederschrift einer Produktion in folgender Form verwendet werden:

$$p_{prol}: \{Linie, ..., Linie\} \stackrel{prolong}{\longrightarrow} \{L\_Linie, ..., L\_Linie\}$$

Sowohl auf der rechten wie auch auf der linken Seite der Produktion stehen Mengenkonfigurationen. Das besondere ist hier, daß die Prozedur *prolong* nicht nur die Zielkonfiguration konstruiert, sondern auch die zu prüfende Ausgangskonfiguration verändert, indem sie nicht hinreichend passende 'Ausreißerinstanzen' daraus entfernt. Als Verarbeitungselemente kommen zunächst Instanzen des Symbols *Linie* in betracht. Der Suchbereich orientiert sich in Lage und Richtung aber nur an dieser einen 'triggernden' Instanz. Es hat wenig Sinn, einen 1000 Pixel langen Bereich nach einer 10 Pixel langen Linie zu konstruieren. Daher wird man die Instanzen des Symbols *L_Linie* auch als Verarbeitungselement für diese Produktion eintragen. Sie liefern genauer orientierte Suchbereiche. In den *line_set*, mit dem man dann die Prozedur *prolong* aufruft, dürfen diese Instanzen aber nicht mit hineingenommen werden. Man erhielte sonst wieder eine rekursive Produktion. Sie dienen nur zur Konstruktion des Suchbereichs, und damit der gezielten Steuerung des Suchprozesses im Potenzmengenraum. Man kann Produktionen, die auf Mengen als Ausgangskonfiguration arbeiten, nur dann verwenden, wenn ein solches Kriterium zur gezielten Suche nach immer besseren Kandidatenmengen zur Hand ist. Das muß nicht unbedingt eine Gerade im Bild sein. Auch andere analytische Konturverfolgungs Modelle wie Kreis, Parabel, Ellipse oder Spline bieten sich im 2D Bildraum an. Denkbar sind Verallgemeinerungen auf das n-Dimensionale, insbesondere quadratische Formen in der 3D Szene oder in 3D Szenenfolgen (also 4D). Dabei werden die Parameter der analytischen Form aufgrund des triggernden Verarbeitungselements zunächst grob berechnet. Diese

erste Näherung dient der Konstruktion eines entsprechenden Untervolumens des Attributbereichs. Die Menge der Instanzen mit Attributwerten in diesem Bereich ist die erste Näherung für die Ausgangskonfiguration. Durch Fehlerquadratminimierung ergibt sich eine bessere Bestimmung der Parameter der Form. Die Ausgangskonfiguration wird durch Entfernen von Ausreißern verkleinert und die Paramterschätzung verbessert, bis alle Fehler eine vorher festgelegte Schwelle unterschreiten. Die entstehende Form wird als Instanz eines höheren Nichtterminals eingetragen. Solch eine Instanz kann wieder als Verarbeitungselement für dieselbe Produktion verwendet werden. Erst wenn sowohl die Menge als auch die Form konstant bleiben, ist ein maximales Element gefunden.

Im Rahmen der im Abschnitt 5.2.2 beschriebenen modellvergleichenden Produktionen können die in diesem Abschnitt vorgestellten Verfahren verwendet werden. Bereits in [LUE-86-2, FUE-88] sind strukturorientierte Verfahren beschrieben, die in ihrer wichtigsten Produktion ein Brückenmodell mit einer *Menge* von unterführenden und überführenden Teilen vergleicht, die Position und Orientierung durch Mittelung gewinnt, und die Ausgangsmenge durch Assoziativzugriff nach Suchbereichen bestimmt, die aus den Attributwerten des triggernden Verarbeitungselementes berechnet werden. Wenn man die KG so konstruiert, daß sie nur Terminale und ein Startsymbol enthält, sowie eine einzige modellvergleichende Produktion, die dieses Startsymbol reduziert aus einer Menge von Terminalen, so hat man kein eigentlich syntaktisches Verfahren mehr vor sich, sondern einen fehlerquadratsummenminimierenden Modellvergleich, wie er in der Mustererkennung Standard ist.

5.3 Komplexitätsabschätzung mit statistischen Methoden

Das in Kapitel 6 gegebene Beispiel für eine 3D Koordinaten Grammatik mag zur Motivation und Anschauung auch der in diesem Abschnitt folgenden eher statistischen Betrachtungen dienen. Im Anhang D.2 sind einige Versuche mit Daten dokumentiert, die mit dem Zufallsgenerator unter kontrollierten Bedingungen erstellt wurden. Vergleicht man die dort ermittelten Zahlen mit den Statistiken im Kapitel 6 so wird deutlich, daß die Rechenkomplexität nicht nur von der Anzahl der gegebenen Terminale und von der algorithmischen Komplexität der Grammatik abhängt, sondern in der Praxis ganz empfindlich von der Verteilung der Attributwerte. Der Entwickler ist gut beraten, der Abschätzung dieser Verteilung einige Aufmerksamkeit zu widmen. Im folgenden werden einige Faustregeln für den Anwender von KGs aufgestellt.

5.3.1 Erste Faustregel: Mit der Ordnung der Symbole fallende Statistik

Für nicht rekursive Grammatiken kann man sich leicht ein Ausgabemodul programmieren, das während der Laufzeit deutlich werden läßt, ob eine implementierte KG für eine gegebene Menge von Terminalen T_{given} Schwierigkeiten bzgl. des Aufwandes macht oder nicht. Man gibt einfach zu jedem Symbol $s \in V$ die jeweils aktuelle Anzahl von Instanzen dieses Symbols, die der Parser kumuliert hat, aus. Dabei reiht man die Symbole nach der Ordnung $\mathcal{O}$ (wie im Abschnitt 5.1.2 definiert) nebeneinander an. Im Falle der RBB-Grammatiken kann man davon ausgehen, daß eine Kante zwischen zwei Symbolen s_1 und s_2 im Produktionsnetz der Grammatik in der Regel eine 'Teil-von' Relation in den zugehörigen Objektkonzepten bedeutet. Es sollte also in der Tat mehr Instanzen mit Symbolteil s_1 geben als solche mit Symbolteil s_2. Dies gilt um so mehr, als das Bedingungsprädikat, welches die zu s_2 führende Produktion einschränkt, in der Regel mehr Wissen über die interessierenden Objekte einbringt. Instanzen mit dem Symbolteil s_2 sollten also mehr Relevanz haben. Die Wahrscheinlichkeit, daß eine Instanz auf eine Störung zurückzuführen ist und nichts mit dem gemeinten Objekt zu tun hat, sollte mit der Ordnung abnehmen. Diese Überlegungen motivieren die folgende Faustregel:

- Wenn der Parser alle möglichen Kumulationen durchgeführt hat, sollte die Anzahl der Instanzen des Symbols s mit $\mathcal{O}(s)$ in der Regel abnehmen. Ist in der Statistik irgendwo erkennbar, daß es mehr Instanzen eines Nichtterminals gibt als Instanzen seiner Vorgänger im Produktionsnetz, so liegt der Verdacht nahe, daß die entsprechenden Produktionen zu schwache Prädikate haben. Hier sollte man versuchen, mehr Wissen einzubringen.

- Wenn die Verarbeitungselemente zufällig aus der Menge aller jeweils aktuellen Verarbeitungselemente gezogen werden, so sollte das Wachstum der Anzahl der Instanzen eines Symbols mit seiner Ordnung abnehmen. Dies um so mehr, je größer die noch verbleibende Menge der Verarbeitungselemente ist. Insbesondere darf, solange die Menge der Verarbeitungselemente noch nicht fast leer ist, die Menge aller Instanzen eines Nichtterminals nicht schneller anwachsen als die Mengen der Instanzen seiner Vorgänger im Produktionsnetz. In diesem Falle kann man den Prozeß abbrechen und die entsprechenden Prädikate untersuchen. Man muß nicht abwarten, bis die Menge der Verarbeitungselemente leerläuft. In vielen Fällen wird das dann praktisch gar nicht mehr möglich sein, weil die Kapazitäten an Rechenzeit und Speicherplatz gesprengt werden.

Es ist möglich, diese Vorgehensweisen teilweise zu automatisieren. Die Relativvolumina der einzelnen Produktionen lassen sich ja leicht über die Schwellwerte

der zugehörigen Prädikate beeinflussen. Man kann also Formeln angeben, mit denen in Abhängikeit vom Verhältnis der Anzahlen der Instanzen eines Symbols und seiner Vorgänger die entsprechenden Schwellwerte gelegentlich neu bestimmt werden. Solche 'Lernregeln' kann man so einrichten, daß es völlig ausgeschlossen wird, daß das Verfahren vom Rechenaufwand her Probleme macht. Dafür kommt dann allerdings die Semantik 'ins Schwimmen'. Während bei Verwendung fester Schwellwerte und Verfahrensparameter klar definiert ist, was etwa Parallelität von Linien jeweils bedeutet, so ist bei Verwendung von Lernregeln diese Bedeutung abhängig von der Vorgeschichte, also von den präsentierten Daten. Man weiß im konkreten Fall nur: Das Verfahren hat sich entschieden, diese und jene Ergebnisse zu produzieren. Was das bedeutet, kann man nicht sagen. In der seriösen Statistik wird gemeinhin dagegen verlangt, daß die Konfidenzintervalle vor dem Test festgelegt werden. Danach erfolgt die Berechnung, und es resultiert die Annahme oder das Verwerfen der Hypothese. Wer die Konfidenzintervalle den Daten anpaßt, produziert bedeutungslose Zahlen. Im Grunde kann man diesen Vorwurf fast allen Lernverfahren machen. Für die im Kapitel 6 angegebene KG wurden daher die Verfahrensparameter fest gewählt, und zwar so, daß sich dafür jeweils eine Begründung angeben läßt. Diese Argumentationen finden sich im Anhang C.

5.3.2 Zweite Faustregel: Relative Dichte kleiner als 1

Betrachtet wird eine BBB-Produktion

$$p: \ (s_1, s_2) \ \overset{\pi}{\underset{\phi}{\longrightarrow}} \ (s_3)$$

Das Bedingungsprädikat π habe das Relativvolumen $V_{rel}(p)$ (wie im Abschnitt 2.3.1 unter Punkt 6 definiert). Wenn nun S_1 und S_2 durch einen Zufallsprozeß erzeugte Mengen von Instanzen der Symbole s_1 und s_2 sind, wobei die Verteilung der Attributwerte im Attributraum D bekannt ist, so kann ein Erwartungswert für die Kardinalität der zulässigen Ausgangskonfigurationen, und damit eine Abschätzung für die Anzahl der zu betrachtenden Ersetzungen, bestimmt werden. Insbesondere setzt man für gleichverteilte Attributwerte in der Menge S_2 - wie etwa in [LED-2] motiviert - die Anzahl der Instanzen, die mit einer gegebenen Instanz $s_1 \in S_1$ zusammen π erfüllen, als Poisson-verteilt an mit Parameter $\lambda = |S_2|V_{rel}(p)$. Die Wahrscheinlichkeit, daß es zu einer gegebenen Instanz bzgl. dieser Produktion k zulässige Partner gibt, ist also

$$P(Anz = k) = \frac{1}{k!} e^{-\lambda} \lambda^k.$$

Für verschiedene k sind diese Ereignismengen disjunkt. Die Wahrscheinlichkeit z. B., daß es höchstens k Partner gibt, ist mithin $\sum_{j=0}^{k} P(Anz = j)$. Nicht alle

zulässigen Ausgangskonfigurationen führen auf verschiedene Zielkonfigurationen. Eine schon betrachtete Instanz von s_3 muß nicht noch einmal erzeugt werden. Die Anzahl der Ersetzungen ist also nicht streng gleich der Anzahl der Ausgangskonfigurationen, sondern u. U. kleiner, was aber vernachlässigt werden kann. Unter diesen Voraussetzungen ist der Erwartunswert dafür also λ. Wenn man mithin dafür sorgt, daß $\lambda < 1$ gilt, so ist die Faustregel 1 erfüllt. Es gibt durch diese Produktion im Mittel weniger Instanzen des Symbols s_3 als des Symbols s_1.

Nach der Definition in Abschnitt 2.3.1 ist die Berechnung des Relativvolumens einer Produktion aufwendig. Für viele Prädikate π ist es aber in trivialer Weise bestimmt. Das Prädikat *apa* (für antiparallel) aus der Beispielproduktion aus Abschnitt 2.1.3 hängt z. B. nur von einem Attribut, der Orientierung, ab. Zu einer beliebigen Orientierung des triggernden Verarbeitungselementes ist der Attributbereich, in dem die Orientierung des Partners liegen muß, damit *apa* erfüllt ist (im folgenden Suchbereich genannt), immer gleich groß. Orientierungsattributbereiche haben ja keinen Rand. Man kann sich also darauf beschränken zu untersuchen, wie groß der Suchbereich ist, relativ zu einer festen Orientierung. Gelten z. B. zur Orientierung 0^o alle Winkel zwischen 30^o und 149^o im Attributbereichbereich bis 179^o als antiparallel, so ist das Relativvolumen dieses Prädikats $\frac{2}{3}$.

Für Nachbarschaften in Bildkoordinatenräumen, wie das Prädikat *adj* aus der Beispielproduktion aus Abschnitt 2.1.3, ist die Berechnung des Relativvolumens schwieriger, weil der Suchbereich kleiner wird, wenn das triggernde Verarbeitungselement am Bildrand liegt. Wenn man diesen Effekt vernachlässigt oder sich mit einer relativ engen oberen Abschätzung des Relativvolumens zufrieden gibt, so kann auch hier ein beliebiges (allerdings nicht am Rand gelegenes) Verarbeitungselement gewählt werden. Die Fläche des zugehörigen Suchbereichs, relativ zur Bildgesamtfläche, liefert dann eine gute Schätzung (und obere Grenze) für das Relativvolumen. In der Beispielproduktion hat der Bildkoordinatenraum eine Fläche von $150^2 = 22500$ Quadratpixel. Wenn zwei Punkte als benachbart gelten, solange ihre euklidische Distanz 20 Pixel unterschreitet, so hat der Suchbereich eine Fläche von $20^2\pi = 1256.6$ Quadratpixel. Man bestimmt also das Relativvolumen als etwas kleiner als 0.0558.

Bei Attributen, die unabhängig voneinander verteilt sind, kann man die sich so errechnenden Volumina miteinander multiplizieren, wenn die Prädikate durch ein $\wedge$ miteinander verknüpft sind. Nimmt man dies für Lage und Orientierung vernünftigerweise an[2], so kommt man im Beispiel auf eine gute Schätzung und obere Schranke für das Relativvolumen von $\frac{2}{3}0.0558 = 0.0372$. Das Inverse davon ist ungefähr 27. Ist also die Anzahl der Linien kleiner als 27, so ist die Kardinalität der Partnermenge für jedes Element Poisson-verteilt mit Parameter $\lambda < 1$. Das ist dann auch der Erwartungswert für die Kardinalität der Menge der Winke-

[2]und vernachlässigt das Prädikat $int_{2d}(...) \in D_2$

linstanzen, die sich damit reduzieren lassen. Und somit ist Faustregel 1 erfüllt. Diese Produktion erzeugt nicht mehr Zielinstanzen als Ausgangsinstanzen.

5.3.3 Konsequenzen für den Wissenserwerb

Durch diese Betrachtungen ergeben sich einige Konsequenzen für den Konstrukteur einer KG:

- Die Dimension eines Attributes ist von ausschlaggebender Bedeutung für die einschränkende Kraft der darauf definierten Prädikate und damit für die Brauchbarkeit einer KG im Hinblick auf den zu erwartenden Rechenaufwand. Die lineare Verkleinerung eines Maximalabstandes wirkt im Eindimensionalen linear, im Bild quadratisch, im Räumlichen kubisch usw. auf das Relativvolumen, das in höherdimensionalen Räumen meist sowieso schon wesentlich kleiner sein wird. Allerdings sollte man bedenken, daß in der Praxis die Gleichverteilungsannahme um so stärker verletzt ist, je höher die Dimension des Attributbereiches ist. Trotzdem ist hierin ein wesentliches Argument für das Bevorzugen von 3D Verfahren gegenüber 2D Bildverarbeitung zu sehen. Es ist einfach sehr unwahrscheinlich, daß eine - etwa durch Fehlkorrespondenz im Stereoprozeß - fälschlich im Raum irgendwo erzeugte Instanz mit anderen in der Weise benachbart und 'verwinkelt' ist, wie es ein bestimmtes gegebenes Modell verlangt (siehe Abschnitt D.2 im Anhang).

- Wenn dem Anwender möglichst viele *unabhängige* Attribute zu seinen Objekten einfallen, die in gewissen Relationen stehen müssen, so sollte er diese 'Constraints' auch verwenden. Die entsprechenden Relativvolumina sind dann miteinander zu multiplizieren. Wenn ein Verfahren bzgl. der genannten Faustregeln Schwierigkeiten macht, so sollte er gezielt nach solchem Wissen suchen. Allerdings ist zu bedenken, daß Unabhängikeit eine sehr starke Forderung ist, die oft schwer zu erfüllen ist. So ist z. B. die Orientierung nicht unabhängig von beiden Endpunktkoordinaten einer Linieninstanz, sondern vielmehr vollständig bestimmt. Die Abhängigkeitsverhältnisse der Attribute untereinander können sehr kompliziert werden, so daß die Berechnung der Relativvolumina schwierig wird. Man muß sich dann mit groben Abschätzungen behelfen, oder Testläufe mit statistisch relevanten Datensätzen durchführen.

- Die Gleichverteilung ist ein *best case*. Im Falle eher zutreffender Verteilungen mit ausgeprägten lokalen Häufungen, sollte man sicherheitshalber nicht von der Mächtigkeit der Gesamtmenge mal dem Relativvolumen ausgehen, sondern von der *maximalen Dichte* der Instanzen im Attributraum mal der

Größe des Suchbereichs. Dann erhält man einen *worst case*, der erfahrungsgemäß nicht allzuweit von der tatsächlichen Anzahl der Zielinstanzen entfernt ist.

- Man kann eine einfache Argumentation bzgl. der generell hohen Rechenkomplexität von KGs aufstellen: Wenn die KG und mit ihr die Bedingungsprädikate und damit die Realativvolumina fest definiert sind, und wenn die Anzahl der gegebenen Terminale n wächst, wobei die Attributverteilung sich nicht ändert, so wächst der Erwartungswert für die Anzahl der Zielkonfigurationen auf jeder Stufe mit, also mindestens exponentiell mit der Ordnung $\mathcal{O}$ für nicht rekursive Grammatiken und mindestens exponentiell mit der Tiefe der Reduktion sonst. Mit den Schwellwerten in den Prädikaten können dem nur konstante (wenn auch sehr kleine) Faktoren entgegengehalten werden. Das bedeutet, daß durch Definition besonders restriktiver Prädikate mit besonders kleinen Relativvolumina das Problem des unverhältnismäßigen Rechenaufwandes lediglich hinausgeschoben werden kann zu etwas größeren n.

  Das klingt zunächst vernichtend. Es kann dem aber entgegengehalten werden, daß in der Bildverarbeitung üblicherweise mit der Anzahl der betrachteten Instanzen auch das Bild selbst wachsend zu denken ist, etwa weil man die Rechenkomplexität eines Verfahrens nur auf einem Ausschnitt testen und dann extrapolieren möchte auf das sehr viel größere Bild. Dann bleibt die *Dichte* der Instanzen im Attributraum gleich, und das Relativvolumen wird in dem Maße kleiner, in dem der Attributraum wächst. Man sollte das für jede Produktion prüfen. Es macht zum Beispiel Sinn zu sagen: 'Die Winkelproduktion reduzierte auf dem 150^2-Pixel Auschnitt aus 17 Linien 9 Winkel. Also wird sie auf dem ganzen Bild von 1500^2-Pixel aus den dort vorhandenen 1700 Linien etwa 900 Winkel reduzieren.' Die gleiche Aussage wäre fahrlässig für eine Produktion, die Parallelenpaare sucht, solange sich das Bedingungsprädikat 'parallel' nur auf die Orientierung stützt, einen Attributbereich also, der nicht mit dem Bild wächst. Dann ist quadratisch zu rechnen, und es ergibt sich durch den Faktor 100 bei den Linien und ein Faktor 10 000 bei den Parallelenpaaren. Das sind 90 000.

Bereits im Abschnitt 5.2.1 wurden Überlegungen dieser Art angestellt. In der Tat läßt sich die Entscheidung über die mehr oder weniger hierarchische Zerlegung eines Objektes in einem Nachbarschaftsgraphen bei der Konzipierung einer KG nicht von diesen statistischen Abschätzungen trennen. Hier verläßliche Kriterien aufzustellen erscheint zur Zeit noch kaum möglich. Man kann nur Beispiele angeben, die funktionieren (wie im nächsten Kapitel durchgeführt), aber keine Garantie für deren Optimalität in Rechenzeit und Speicherbedarf übernehmen.

Chapter 6

Beispiel eines 3D Verfahrens

Als Beispiel für eine praktisch anwendbare KG wird in diesem Kapitel ein implementiertes Verfahren zur Erkennung von Transportern des Typs VW-Bus Doppelkabiner in Bildfolgen des sichtbaren Spektralbereichs vorgestellt. Das Verfahren gliedert sich in vier Teilschritte: Terminalextraktion, 2D Parser, Stereoanalyse und 3D Parser. Abschnitt 6.4 beschäftigt sich dabei mit der Natur der Ausgangsdaten und damit, wie aus diesen die Terminale des 2D Parsers gewonnen werden. Dieser selbst ist Gegenstand des Abschnitts 6.3. Abschnitt 6.2 beschreibt, wie durch stereoskopische Berechnung aus den reduzierten Instanzen des Startsymbols des 2D Parsers die Terminale für den 3D Parser erzeugt werden. Dieser selbst ist Gegenstand des Abschnitts 6.1. Die Konzipierung beginnt mit einem Modell des gesuchten Objekts, und schreitet von da hinunter zu Problemen der Bildverarbeitung. Sie bedient sich wesentlich der Hilfsmittel aus dem Abschnitt 5.2. Zur Darstellung der Ergebnisse benötigt man Hilfsmittel aus dem Abschnitt 5.1, von denen die wichtigsten in einem mit WEEK[1] bezeichneten Programmpaket bereits implementiert sind. Zur Entwicklung des Verfahrens und Einstellung der Verfahrensparameter wurde die Bildfolge BF1 (siehe Anhang F) verwendet. Um die Leistungsfähigkeit und die Grenzen des Verfahrens aufzuzeigen, wurde das Verfahren an weiteren Bildfolgen getestet. Die Ergebnisse werden im Abschnitt 6.5 präsentiert. Darüberhinaus wurden mit dem Verfahren Experimente mit Mengen von Instanzen durchgeführt, die durch einen Zufallsgenerator erzeugt wurden. Das ist im Anhang D.2 dokumentiert.

6.1 Modellvergleich und 3D Parser

Es liegt eine Detektions- bzw. Lokalisationsaufgabe vor. Das Verfahren soll entscheiden, ob Fahrzeuge eines bestimmten Typs (hier VW-Bus Doppelkabiner)

[1] **W**issens **E**rwerbs und **E**rklärungs **K**omponente

in der Szene vorhanden sind, und wenn ja, wo sie sind, wie sie orientiert sind und auch, welcher Grad von Vertrauen jeder dieser Aussagen über Lage und Orientierung zukommt. Es wird dabei davon ausgegangen, daß die Geometrie der gesuchten Objekte bekannt ist. Solches geometrisches und strukturelles Modellwissen kann zum Beispiel einfachen CAD Modellen entnommen werden. Das hier verwendete Modell ist im Anhang C gemäß den Definitionen und Notationen des Abschnitts 5.2.2 angegeben. Abbildung 6.1 zeigt eine perspektivische Darstellung. Insgesamt wurde dieses Modell auf das eigentlich Charakteristische an der räumlichen Gestalt eines solchen Fahrzeugs hin entworfen. Es besteht aus 16 Punkten oder 10 Flächen. Dies unterscheidet den gewählten Ansatz zur Repräsentation von 3D Wissen von solchen, die sehr detailreiche Untergliederungen vornehmen (wie etwa [FAUG-93-1]). Details, wie z. B. die Räder, sind weggelassen. Auch die unteren Kanten der Seitenflächen sind nicht modelliert, einerseits, weil sie von den Radkästen unterbrochen sind, was komplexer zu modellieren wäre, und andererseits, weil sie bei Fahrzeugen im Gelände sowieso häufig verdeckt sind.

Gemäß den in Abschnitt 5.2.2 vorgestellten Prinzipien sollen die Instanzen des zugehörigen Symbols *light_truck* durch einen fehlerquadratsummenminimierenden Modellvergleich gefunden werden. Als Modell dient also die Liste mit den sechzehn Punkten. Der zugrundeliegende Raum ist der dreidimensionale, euklidische Raum. Als Transformationen werden die Rotationen und Translationen zugelassen[2]. Ein wesentlicher Vorteil der 3D Modellierung liegt im Ausschluß der Skalierungen. Durch assoziativen Zugriff nach Suchvolumina wird die Komplexität der Aufzählung möglicher Ausgangskonfigurationen auf ein erträgliches Maß herabgesetzt. Wichtig ist es dabei, eine gute Ausgangsnäherung für die Transformation des Modells zu haben, damit die Suchvolumina nicht zu groß ausfallen müssen. Das wurde in diesem Beispiel erreicht, indem die beiden prägnanten, unten offenen Seitenflächen zusammengefaßt werden zu einem 'Hilfsnichtterminal', das mit *E_structure* bezeichnet ist. Diese Vorgehensweise entspricht einem teilweisen Durchführen der Konstruktion aus dem Abschnitt 2.3.1 zum Ersetzen einer RBB-Produktion durch mehrere BBB-Produktionen. Auch wenn das triggernde Verarbeitungselement für die modellvergleichende Produktion eine der anderen Flächen ist, richtet sich gemäß den im Abschnitt 5.2.2 erläuterten Prinzipien die Ausgangsnäherung der Transformation nach dieser *E_structure*, weil davon auszugehen ist, daß sie damit etwa doppelt so genau ist. Überschlagsweise kann man wie folgt abschätzen: Doppelte Genauigkeit in der Ausgangsnäherung bedeutet halber Schwellwert beim Suchvolumen im dreidimensionalen Fall, also ein Achtel des Volumens. Damit verringert sich die Mächtigkeit der Kandidatenmengen - gleichverteilte Attributwerte vorausgesetzt

[2]An dieser Stelle sind leicht zusätzliche Einschränkungen möglich, etwa dergestalt daß Fahrzeuge mit allen vier Rädern auf dem Boden stehen. Das war aber im vorliegenden Fall gar nicht notwendig.

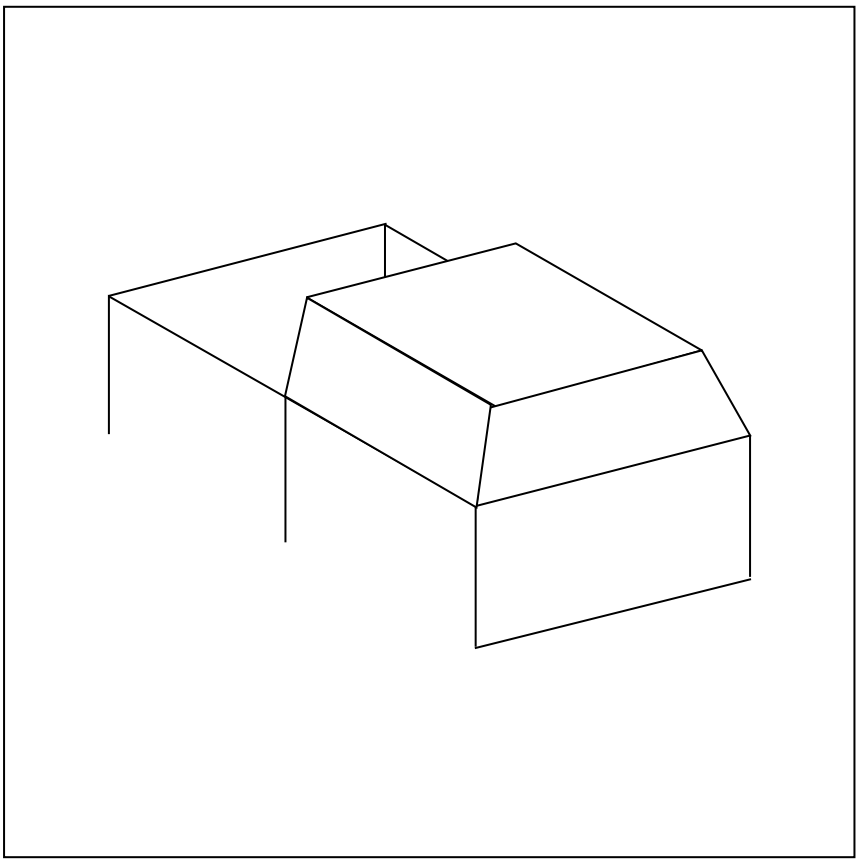


Figure 6.1: **3D CAD Modell des Transporters**

- um diesen Faktor. Die Anzahl der Kombinationsmöglichkeiten, die auf ihre Eignung als Ausgangskonfiguration geprüft werden müssen, sinkt um den Faktor 8^{-k}, wobei k hier die Gesamtzahl der Korrespondenzen angibt. Daß die Praxis nicht ganz so krasse Zahlen liefert, liegt daran, daß die Attributwerte eben nicht gleichverteilt sind. Trotzdem sind solche Abschätzungen als Anhaltspunkt sinnvoll und brauchbar.

Abbildung 6.2 zeigt den Nachbarschaftsgraphen des Modells, wie er sich durch Einführung der *E_structure* ergibt. Da die anderen Flächen etwa die gleiche Größe und Form haben wie die beiden Teile der *E_structure*, werden hier dieselben Symbole *U_structure* verwendet. Insbesondere kann auf diese Weise eine Objektteilfläche auch dann gefunden werden, wenn gar nicht alle ihre Konturen in den Bildern erscheinen. Das ist in der Praxis sehr häufig der Fall. Da man nicht wissen kann, welche Teilkontur fehlt, wird die *U_structure* in der modellvergleichenden Produktion p_2 als Viereck mit den Rotationen als Symmetrie betrachtet (siehe Abschnitt 5.2.3). Daneben wird es in der Produktion p_1, die die *E_structure* zusammensetzt, als spiegelsymmetrisch betrachtet, da man an beiden 'Schenkeln' einen Partner anlagern kann. Das kleine, sich so ergebende Produktionsnetz, das in der Abbildung 6.3 dargestellt ist, kann für sich allein über all diese Details naturgemäß wenig aussagen.

Es ergibt sich die folgende KG: *light_truck* ist Startsymbol, und es gibt noch ein weiteres Nichtterminal *E_structure*. Das Symbol *U_structure* dient in der 3D Analyse als Terminal. Der Attributraum umfaßt sechs kartesische 3D Ortsattribute zur Abspeicherung von Positionen in der Szene mit einer Auflösung von einem Zentimeter pro Voxel, ein Richtungsvektorattribut (normiert auf tausend Einheiten Länge) zur Abspeicherung von Oberflächennormalen und einen Rollwinkel um die Normale in Grad:

$$D = \begin{pmatrix} ver_1 & \text{begrenzter 3D Bereich} \\ \vdots & \vdots \\ ver_6 & \text{begrenzter 3D Bereich} \\ norm & \text{3D Orientierung} \\ roll & \text{2D Orientierung} \end{pmatrix}$$

Zulässig sind alle Permutationen der ersten sechs Ortsattribute. Für die Produk-

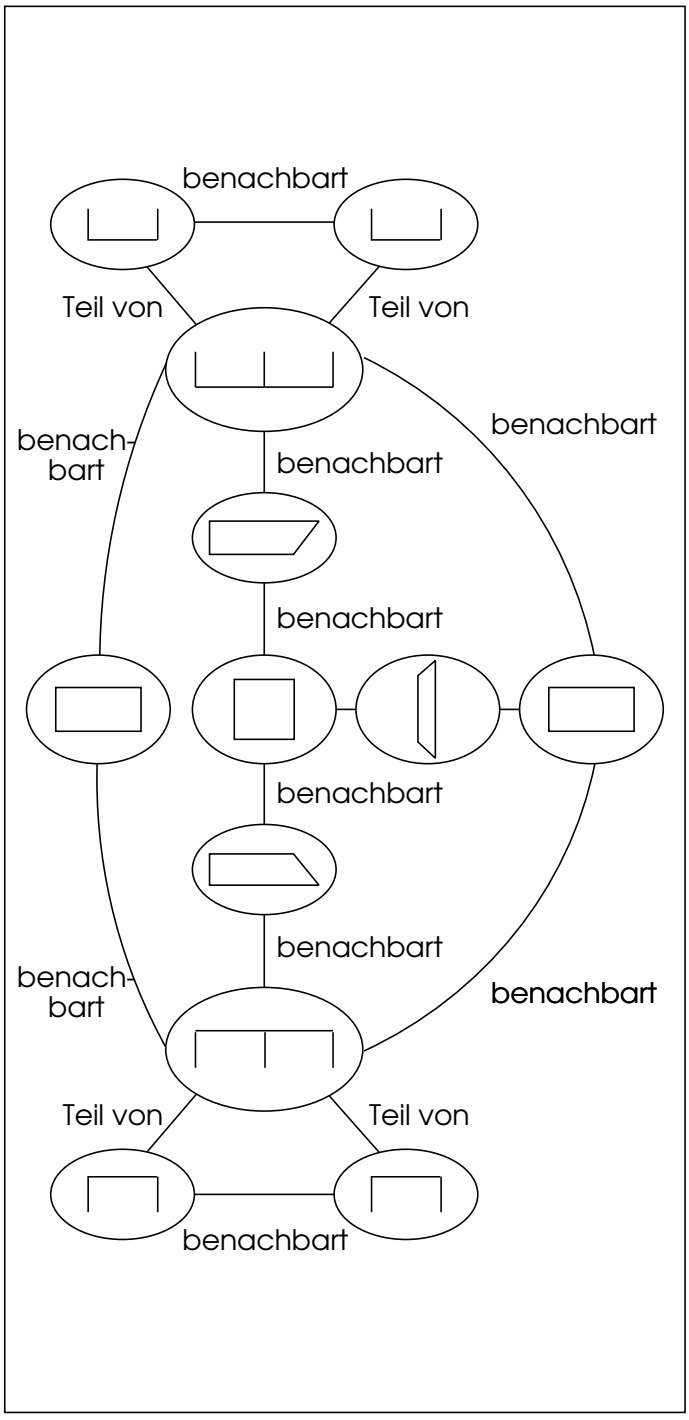


Figure 6.2: **Nachbarschaftsgraph des Transporters**
der Knoten für das Fahrzeug selbst ist nicht dargestellt; er ist mit allen Knoten außer den Teilen der E Strukturen durch 'Teil von' Kanten verbunden.

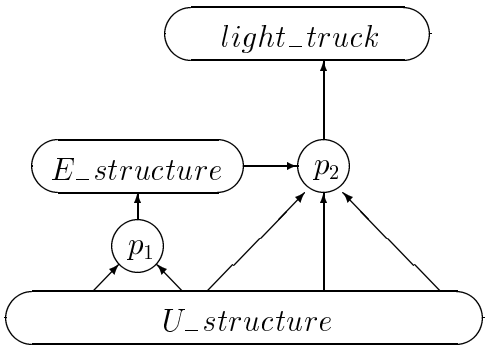


Figure 6.3: **3D Produktionsnetz des Transporters**

$$p_1 : (U\_structure, U\_structure) \xrightarrow[\substack{ver_1 = ver_{1,1} \\ ver_2 = ver_{1,2} \\ ver_3 = \frac{1}{2}(ver_{1,3} + ver_{2,2}) \\ ver_4 = \frac{1}{2}(ver_{1,4} + ver_{2,1}) \\ ver_5 = ver_{2,3} \\ ver_6 = ver_{2,4} \\ norm = c\frac{norm_1 + norm_2}{|norm_1 + norm_2|}}}{\substack{adj_\infty(ver_{1,3}, ver_{2,2}) \\ \wedge\, adj_\infty(ver_{1,4}, ver_{2,1}) \\ \wedge\, prl(norm_1, norm_2)}} (E\_structure)$$

tion

wird, entsprechend der Spiegelsymmetrie, nur die Vertauschung (1,4)(2,3) zugelassen, und zwar simultan in beiden Instanzen der Ausgangskonfiguration. Wegen der Spiegelsymmetrie in der Zielinstanz *E_structure* muß für jede der beiden entstehenden Produktionen eine *U_structure* Instanz nur einmal als Verarbeitungselement in die Warteschlange eingetragen werden. Diese Produktion ist den Beispielproduktionen aus den Abschnitten 2.1.3 und 5.1.3 sehr ähnlich. Solche einfachen Gruppierungsproduktionen verwenden kaum Modellwissen. Die Statistik der durch sie erzeugten Instanzen richtet sich wesentlich nach den im Abschnitt 5.3 beschriebenen Faustregeln. Es sind demnach Probleme mit rasch mit der Tiefe der Reduktion wachsendem Rechenaufwand zu erwarten, wenn mit steigender Zahl von Ausgangsinstanzen der Attributraum immer dichter belegt wird. Wenn dadurch die Zahl der Instanzen zu groß wird, muß Modellwissen eingebracht werden. Die nächste Produktion erreicht dies durch die im Abschnitt 5.2.2 erläuterten Methoden.

$U\_structure$	12997
$E\_structure$	25427
$light\_truck$	22

Table 6.1: **Statistik der reduzierten 3D Instanzen aus BF1**

$$p_2 : (E\_structure, U\_structure, U\_structure, U\_structure) \xrightarrow[(ver_1, norm, roll) = ls\_sqr\_tr]{template\_matching} (light\_truck)$$

Im Anhang B sind $template\_matching$ und ls_sqr_tr definiert. Die Tabelle 6.1 zeigt, wie an dieser Stelle die 'kombinatorische Explosion' gebrochen wird. Sie zeigt die Anzahlen der Instanzen nach Leerlaufen der Warteschlange, aufgeschlüsselt nach Symbolen, wie sie sich aus der Bildfolge BF1 (siehe Anhang F) ergeben.

Es hat in der vorliegenden Aufnahmesituation keinen Sinn, nach allen Teilen des zu identifizierenden Objektes zu suchen. Die Aufnahmepositionen liegen näherungsweise auf einer Geraden. Es bleibt also eine Hälfte des Fahrzeugs auf jeden Fall verdeckt. Es genügt, wie das Beispiel deutlich macht, wenn nur einige charakteristische Teile gefunden werden. Die Produktion p_2 geht also nicht von dem gesamten Nachbarschaftsgraphen des Fahrzeugs aus, sondern beschränkt sich auf die $E\_structure$ Instanz selbst und die direkt benachbarten Flächen. Auch diese drei Flächen müssen nicht alle gefunden werden, sondern es genügen zwei. Die Abbildung 6.4 zeigt eine Ansicht einer solchen Konfiguration aus dem Beispieldatensatz und die daraus reduzierte Startinstanz. Es wird deutlich, daß eine $E\_structure$ Instanz der Spiegelsymmetrie des Fahrzeugs entsprechend zweimal als Verarbeitungselement eingetragen werden muß. Sie kann entweder linke oder rechte Seite des Fahrzeugs sein. Die entsprechenden Korrespondenzen sind in den Tabellen C.1 und C.2 im Anhang aufgeführt. Zu jeder gefundenen Startinstanz des Symbols $light\_truck$ wird die Fehlerquadratsumme und die Anzahl der gefundenen Teile abgespeichert. Nach dem Leerlaufen der Warteschlange kann dann unter den Instanzen, die so nahe beieinanderliegen, daß sie nicht verschiedene Fahrzeuge meinen können, das 'beste' anhand dieser Bewertungsgrundlagen ausgewählt oder eine Mittelung vorgenommen werden. Die in Abbildung 6.4 wiedergegebene Startinstanz hat die niedrigste Fehlerquadratsumme im vorliegenden Beispieldatensatz. In der Abbildung 6.5 ist sie schwarz dargestellt.

Eine Überlagerung eines Ausgangsbildes mit den gefundenen Startinstanzen kann zur Beurteilung des Ergebnisses herangezogen werden. Dabei werden die dem Bild zugehörigen Kameraparameter zur Rückprojektion verwendet. Abbildung

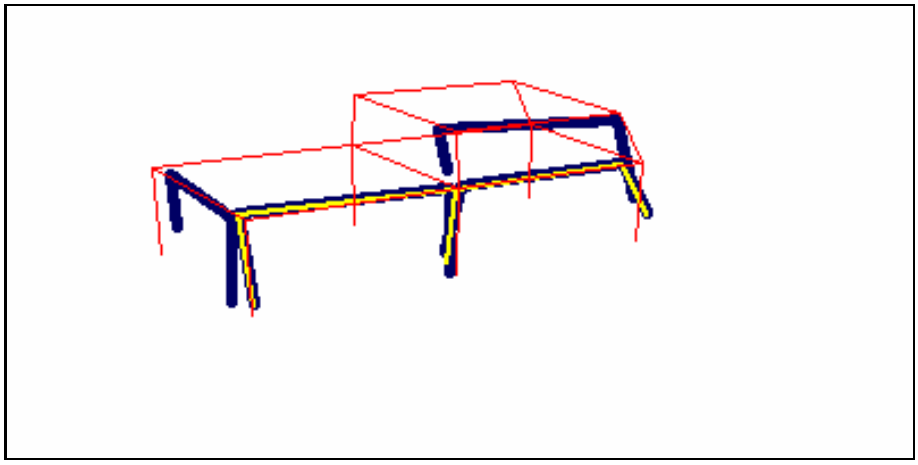


Figure 6.4: **Graphische Darstellung der beteiligten Instanzen**
light_truck (rot), *E_structure* (gelb) und *U_structure* (blau)

Figure 6.5: **Rückprojektion der gefundenen Startinstanzen**
Die Farben kodieren die Bewertung: schwarz = beste Instanz
dann rot, dann grün und lila sind die schlechtesten

6.5 zeigt dies für Bild Nummer acht aus dem Beispieldatensatz BF1. Alle *light_truck* Instanzen kommen auf die 'richtige' Stelle, referieren also auf das einzige Fahrzeug. Bei allen wurden außer dem *E_structure* Teil noch jeweils zwei Flächen gefunden. Als Bewertung kann also nur die Fehlerquadratsumme herangezogen werden. Einige der in diesem Sinne schlechteren Instanzen sind falschherum orientiert. Ab einer mittleren Bewertung sind alle Instanzen offensichtlich als semantisch korrekt anzusehen. Die Instanz mit der optimalen Fehlersumme ist nicht unter den subjektiv 'am besten zum Bild passenden' Instanzen. Sie erscheint leicht um die Rollachse verdreht. Eine Mittelung etwa über alle rot eingezeichneten Instanzen (das sind die besten fünf) käme aber der tatsächlichen Lage und Rotation des Fahrzeugs sehr nahe.

Die Abbildung 6.6 zeigt den Reduktionsbaum der bestbewerteten Startinstanz. Hier werden sowohl die im Abschnitt 4.1 erläuterten Vorteile als auch die Probleme des Abarbeitens der Warteschlange nach Bewertungskriterien deutlich. Die beste Startinstanz wird bereits nach einem kleinen Bruchteil der Rechenzeit gefunden, weil die Bewertung so gewählt wurde, daß sie mit der Ordnung der Instanzen aufsteigt. Dadurch gewinnt die Suche einen *depth first* Charakter. Weil hier bereits die *E_structure* Instanz sehr gut bewertet ist, kommt man fast unmittelbar nach ihrer Entstehung weiter zur Startinstanz. Die Bewertung der *E_structure* Instanz resultiert aber nicht aus einer guten Bewertung ihrer Vorgänger im Reduktionsbaum, sondern aus der engen Nachbarschaft und parallelen Ausrichtung ihrer Vorgänger. Die wird erst explizit, wenn diese Ausgangskonfiguration geprüft wird, und das ist erst dann der Fall, wenn eine der beiden beteiligten *U_structure* Instanzen als Verarbeitungselement den Regelindex dieser Produktion zugewiesen bekommt. Da diese Instanzen nur mäßig bewertet sind, wird eine große Menge anderer Alternativen vorher geprüft.

Bezüglich der im Abschnitt 5.3 angesprochenen Verteilung der Instanzen im Attributraum wird aus den in der Abbildung 6.7 abgebildeten Ansichten deutlich, daß die Annahme einer Gleichverteilung hier noch mehr als in der Bildverarbeitung mit 2D KGs verletzt ist. In der Tat ergibt eine Gleichverteilung der *U_structure* Instanzen in der Szene nur sehr wenige vereinzelte *E_structure* Instanzen. Dies wird durch die Experimente im Anhang D.2 belegt. Die Anzahl der *E_structure* Instanzen bei aus realen Daten resultierender Verteilung muß empirisch bestimmt werden. Wie aus Tabelle 6.1 deutlich wird, gibt es aus der Bildfolge BF1 heraus etwa doppelt soviele Instanzen des Symbols *E_structure* wie Instanzen des Symbols *U_structure*. Der Übersichtlichkeit halber sind die Instanzen des Symbols *E_structure* nicht eingeblendet. Statt dessen sind den in blau dargestellten 12997 *U_structure* Instanzen in gelb die vier *U_structure* Instanzen aus dem Reduktionsbaum aus Abbildung 6.6 überlagert. In grün ist die daraus resultierende beste Startinstanz dargestellt. So wird deutlich, wie die KG aus der recht 'chaotisch' angeordneten, umfangreichen Terminalmenge die 'wenigen richtig zueinander passenden' auswählt.

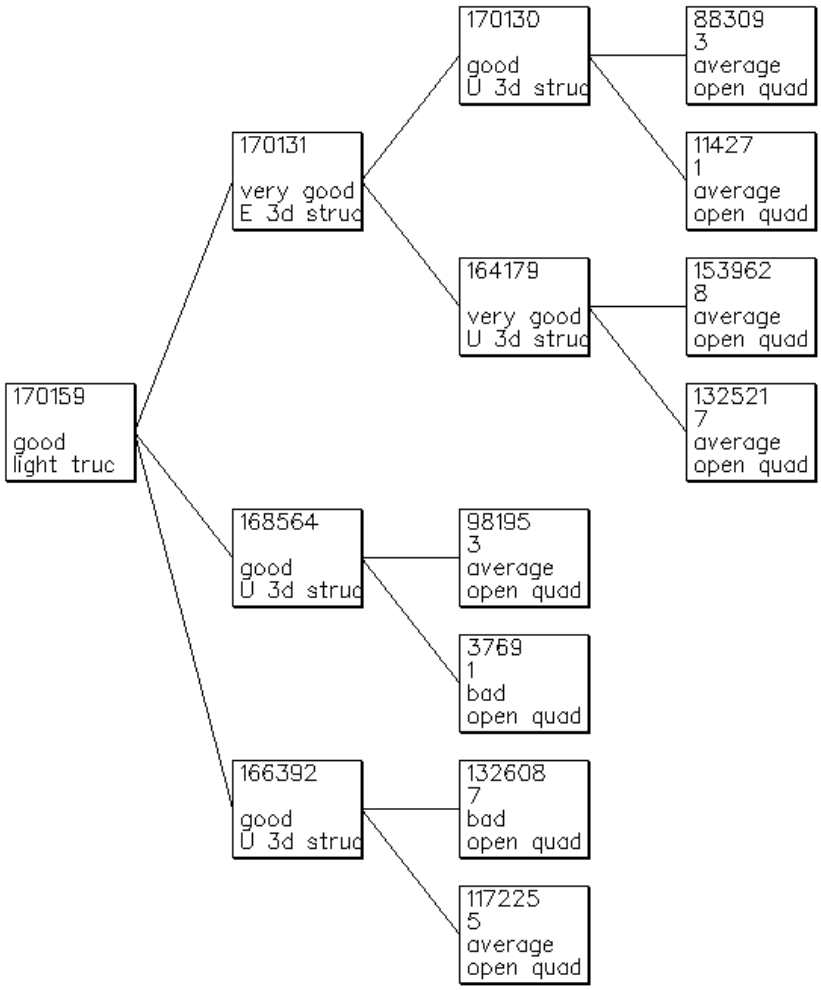


Figure 6.6: **Reduktionsbaum der besten Startinstanz**
Aufgeführt sind der Instanzenindex, der Symbolteil
und die Hilfsattribute Bewertung und Bildnummer
(zum Symbol *open_quad* siehe Abschnitt 6.2).

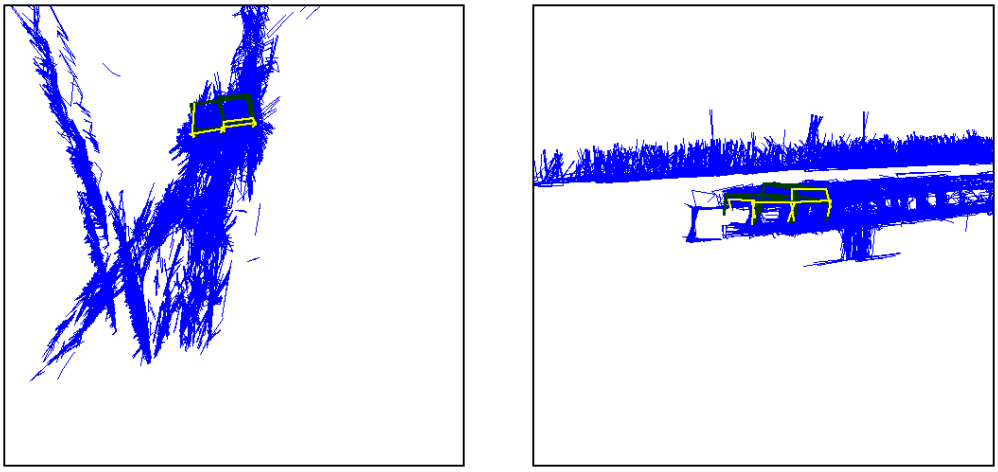


Figure 6.7: **Verteilung der Attributwerte in der Szene**
links in der Aufsicht, rechts projiziert mit den Parametern des 8. Bildes

6.2 Gewinnung der 3D Primitive durch Stereoansatz

Als Terminale dienen dem 3D Parser in diesem Beispiel annähernd rechteckige Flächen im Raum, denen eine Seite fehlt, und die daher mit $U\_structure$ bezeichnet sind. Diese entstehen durch Stereokorrespondenz aus annähernd parallelogrammförmigen 2D Strukturen (offenen Vierecken) in verschiedenen Bildern einer Bildfolge (siehe Abbildung 6.8) unter Verwendung der Funktion int_{3d}. Berücksichtigt werden dabei auch einige geometrische Bedingungen, die sich aus dem Fahrzeugmodell ergeben, und die unter dem Prädikat $geom\_const$ zusammengefaßt sind. Im Anhang B sind int_{3d} und $geom\_const$ definiert. Ordnet man diesen Strukturen das Symbol $open\_quad$ zu und definiert einen entsprechenden Attributraum, so kann auch dieses Verfahren als BBB-Produktion einer KG geschrieben werden:

$$(open\_quad, open\_quad) \quad \overset{\substack{geom\_const \\ int_{3d}}}{\longrightarrow} \quad (U\_structure)$$

Man kann diese einzelne Produktion (es gibt hier keine Symmetrien) für sich als KG mit einem Terminal und einem Startsymbol betrachten oder der im vorigen Abschnitt betrachteten 3D KG hinzufügen. Dann wird aus derem Terminal ein Nichtterminal, und es gilt $T = \{open\_quad\}$. Der Attributraum muß

entsprechend erweitert werden. Wenn man genug Speicherplatz hat, sollte man diesen Aufwand nicht scheuen. Mit der Erklaerungskomponente kann der Reduktionsbaum nämlich dann bis zu den *open_quad* Instanzen in den einzelnen Bildern durchgeführt werden. So wird deutlich, welche Bilder aus der Folge wie an der Entstehung der bestbewerteten *light_truck* Instanz beteiligt sind. Im vorliegenden Falle war dies möglich, so daß der in Abbildung 6.6 dargestellte Reduktionsbaum diese Information liefern kann. Konzeptionell sollte der Stereoprozeß aber getrennt von der 3D Analyse betrachtet werden, um der Modularität des Ansatzes willen.

Die Bildfolge enthält relativ wenig Bilder, und die Abstände zwischen den einzelnen Kamerastandpunkten sind vergeichsweise groß (siehe Abbildung 6.15b)). In [BAKER] ist sehr schön geschildert, wie man Fehlkorrespondenzen vermeiden kann, wenn man mit so vielen Bildern und so engen Abständen zwischen den Kamerastandpunkten arbeitet, daß sich die Tiefe als stetige Verschiebung durch Bildstapel ergibt. Zur Fehlkorrespondenzvermeidung könnten im übrigen Farb- oder Texturattribute herangezogen werden. Natürlich können als Terminale des 3D Parsers auch anders (z. B. durch aktive Methoden) gewonnene Flächensegmente dienen. Prinzipiell erlaubt der Ansatz, alle Arten von Sensoren oder anderen Evidenzquellen (das beinhaltet z. B. Landkarten) auch zu berücksichtigen. Das vorliegende Stereoverfahren wurde hauptsächlich gewählt, weil es sich eben gerade durch die vielen Fehlkorrespondenzen eignet zur Demonstration der Robustheit von solchen sich auf Topologie und Geometrie von Modellen abstützenden 3D Parsern. Es muß ein extrem konstruierter Sonderfall vorliegen, damit diese Störungen sich zur *räumlichen Gestalt* des gesuchten Fahrzeugs zusammensetzen lassen. Die Abbildung 6.7 zeigt alle aus den acht Bildern der Bildfolge BF1 erzeugten 3D Terminale aus verschiedenen Blickwinkeln in blauer Farbe. Insgesamt werden aus den 163774 *open_quad* Instanzen 12997 dreidimensionale *U_structure* Instanzen generiert.

6.3 2D Parser

Die offenen Vierecke sind das Startsymbol *open_quad* einer einfachen 2D KG. Das zugehörige Produktionsnetz ist in Abbildung 6.9 dargestellt. Auch diese Produktionen könnten im Prinzip mit den in den vorangehenden Abschnitten beschriebenen Produktionen vereinigt werden zu einer KG, so daß Reduktionsgraphen vom Startsymbol *light_truck* bis hinunter zu den terminalen kurzen Linienstücken betrachtet werden könnten. Aus Aufwandsgründen wurde darauf verzichtet.

Für sich betrachtet folgt die 2D KG im wesentlichen den in den Kapiteln 2 bis 5 dargestellten Prinzipien. Die Produktionen p_2 und p_3 sind bekannt aus dem Ab-

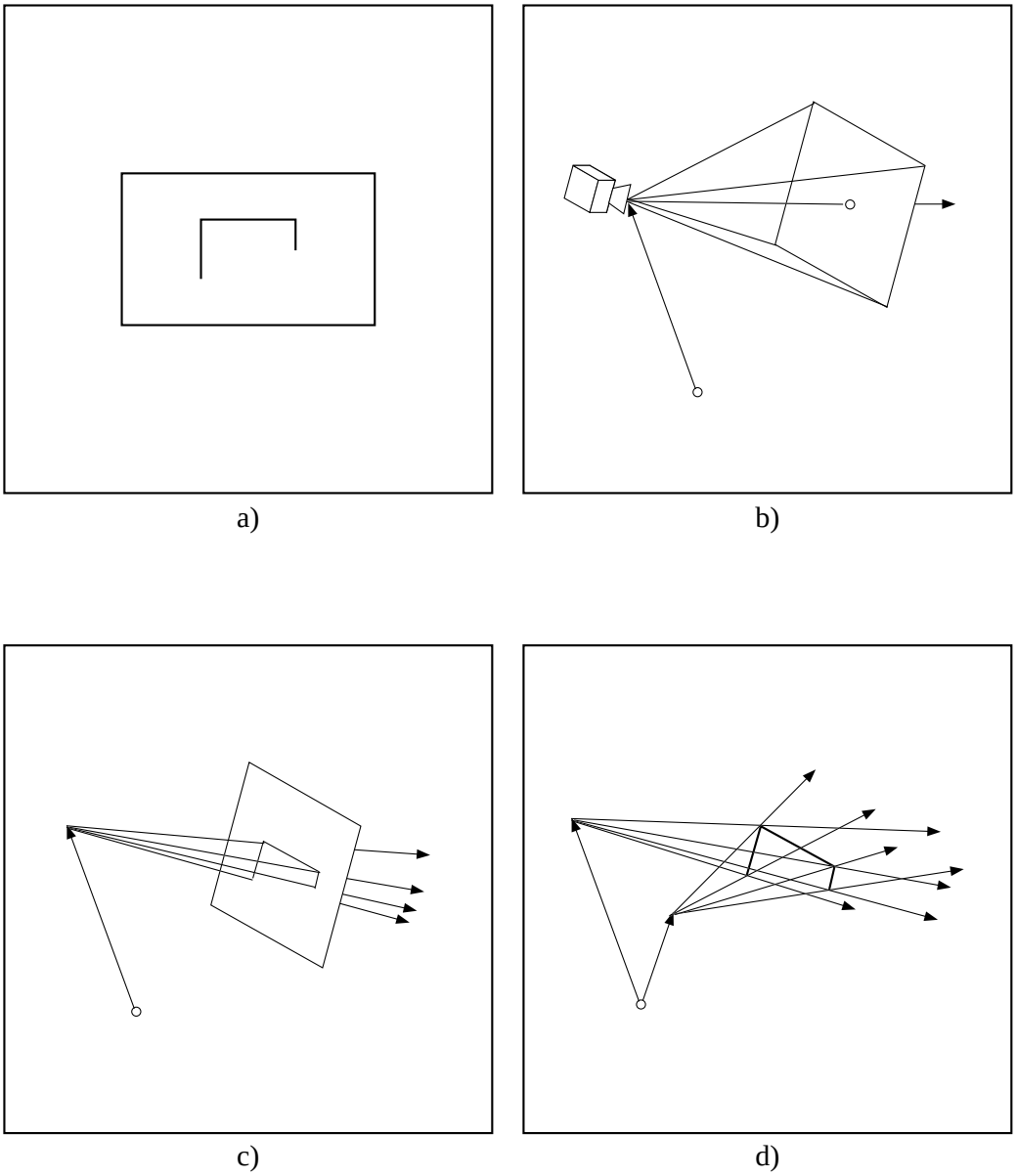


Figure 6.8: **Stereoansatz zur Erzeugung der 3D Terminale**
a) *open_quad* Instanz in einem Bild
b) Kameramodell in der Szene
c) Durch inverse Projektion der 2D Instanz erzeugte Strahlen
d) Durch Schnitt korrespondierender Strahlen erzeugte 3D Terminal Instanz

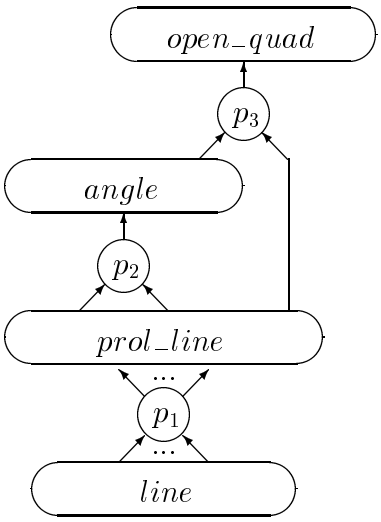
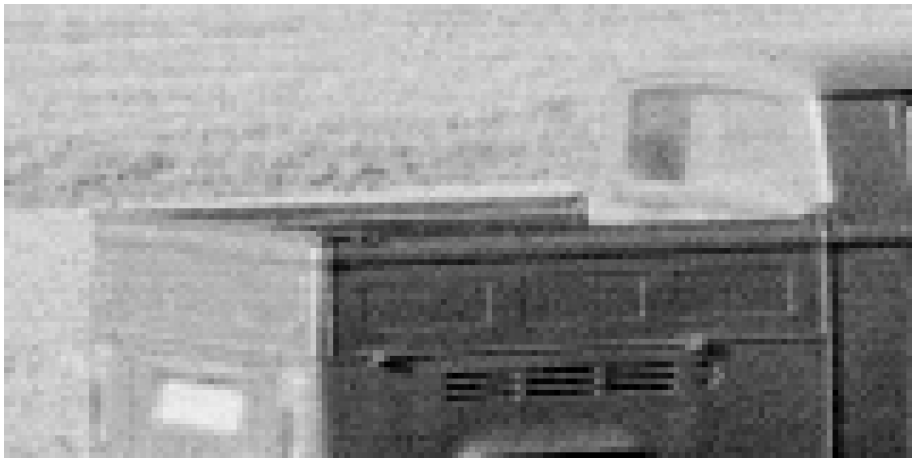


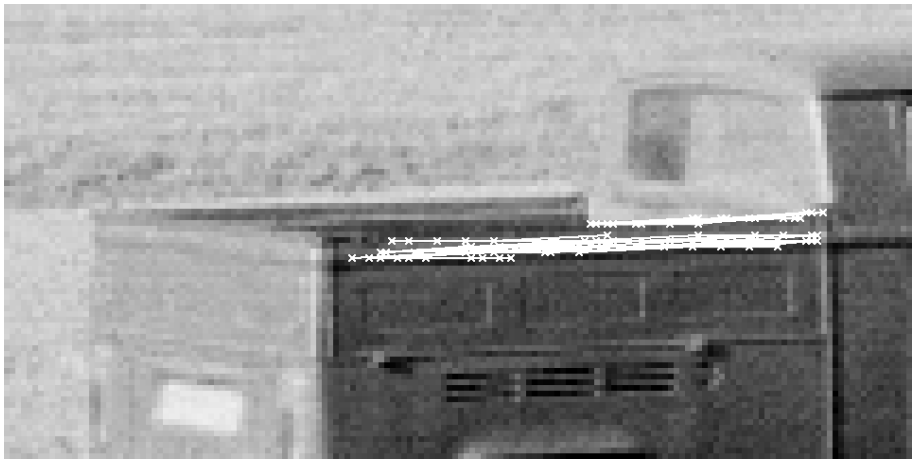
Figure 6.9: **Im Beispiel verwendetes 2D Produktionsnetz**

schnitt 2.2.3. Sie wurden lediglich ergänzt um die ihren Symmetrien entsprechenden Familien gemäß Abschnitt 5.2.3. p_1 ist eine Linienverlängerungsproduktion wie im Abschnitt 5.2.5 dargestellt. Abbildung 6.10a) zeigt einen Bildauschnitt, an dem sich die zugrundeliegende Bildverarbeitungsproblematik exemplarisch verdeutlichen läßt: Die Richtung des Grauwertgradienten entlang einer Objektaußenkontur kann sprunghaft das Vorzeichen wechseln, wenn nämlich der Hintergrund von heller auf dunkler wechselt. Viele wichtige Konturen sind u. U. auch eher durch streifenförmige Strukturen als durch Grauwertkanten gegeben. Mit der somit notwendigen Linienverlängerungsproduktion handelt man sich neue Probleme ein. Die Oberkante der vorderen Ladebordwand z. B. hebt sich heller vor der Innenseite der hinteren Ladebordwand und dunkler vor der Führerhausrückwand ab. An der Stelle, wo der Gradient das Vorzeichen wechselt, gibt es einen Versatz um etwa drei Pixel aus der Kollinearität heraus. Andererseits wurde das Bild aber vor dem Abtasten und Digitalisieren nicht tiefpaßgefiltert[3], so daß durchaus parallele Kanten im Bild erscheinen, die enger beieinander liegen. Betrachtet man die in Abbildung 6.10c) in gelber Farbe dargestellte Instanz des Symbols *prol_line* (die Kreuze an den Enden markieren die Pixelkoordinaten, mit denen diese Instanz attributiert ist), so kann man sich die Oberkante der Ladebordwand kaum besser dargestellt denken. Sieht man sich aber ihre Vorgänger im Reduktionsbaum in der Abbildung 6.10b) an, so stellt man

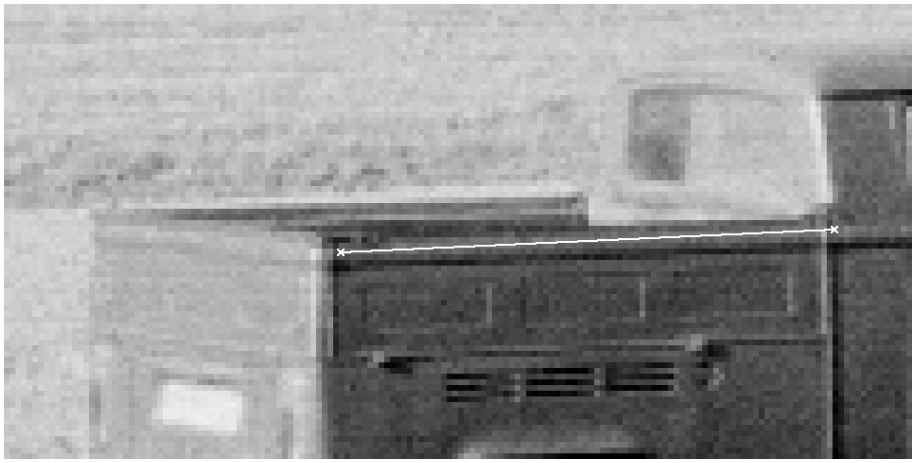
[3]Die Methoden der klassischen Signalverarbeitung verlangen gemäß dem Abtasttheorem eine Tiefpaßfilterung mit etwa der halben Abtastrate als Grenzfrequenz, um Schwebungserscheinungen, wie sie etwa bei den Motorlüftungsschlitzen unter der Ladefläche in diesem Beispiel auftreten, zu vermeiden. Hier erzeugt die Abtastung Struktur, die in der analogen Vorlage nicht enthalten ist.



a) Vergrößerter Auschnitt aus einem Ausgangsbild



b) Vorgänger der in c) gezeigten Instanz



c) *prol_line* Instanz an der oberen Ladebordwandkante

Figure 6.10: **Problematik der Konturverfolgung**

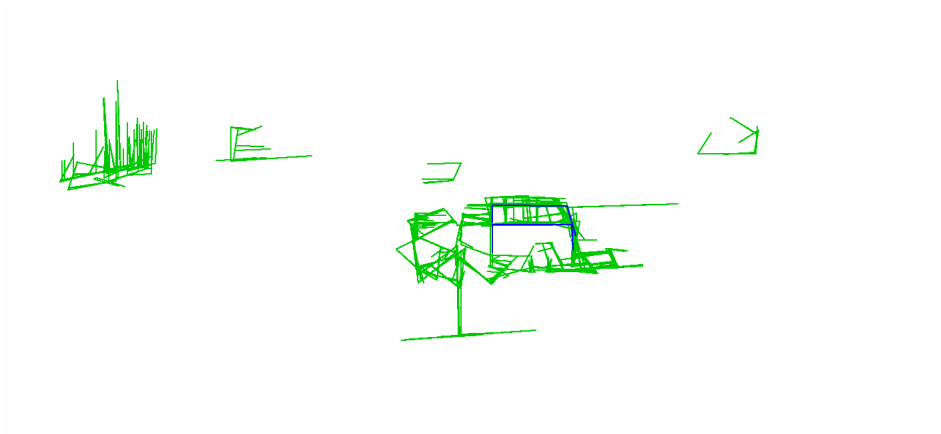
Figure 6.11: **Problematik der 2D Verarbeitung**
Der dargestellten 2D Instanz entspricht keine Struktur in der Szene

fest, daß hier offenbar Instanzen aus verschiedenen Konturen in eigentlich inkorrekter Weise in eine Mittelung einbezogen wurden. Man kann diese Instanz nur *top down* finden, indem man einen Reduktionsgraphen einer sehr guten Fahrzeuginstanz untersucht. Daß sie so gut ist, ergibt sich nicht aus ihren Vorgängern, sondern aus ihren Nachfolgern! Solche Problematik findet man nicht nur in der Linienverlängerung, sondern sie wohnt allen 2D Verfahren inne, wie das Beispiel in Abbildung 6.11 zeigt.

2D KGs können sich nur auf zweidimensionale Nachbarschaft und Geometrie stützen. Sie setzen daher unter Umständen Konturen zusammen, die in der Szene keine Entsprechung haben. Die Abbildung zeigt, dem Ausgangsbild überlagert, ein offenes Viereck, welches aus der Vorderkante des Fahrzeugs, dem Schatten des Fahrzeugs und einer weit hinter dem Fahrzeug liegenden waagerechten Struktur gebildet wird. Es wäre extrem aufwendig, eine 2D Grammatik zu entwerfen, die solche Fehler verläßlich vermeidet und nur semantisch korrekte Konturen zuläßt. Hingegen haben die Untersuchungen an diesem Beispiel gezeigt, daß es einfach ist, aus den vielen eingetragenen 3D Terminalen durch eine KG, wie sie im Abschnitt 6.1 erläutert ist, diejenigen herauszufiltern, die sich zu einem solchen Fahrzeug zusammenfügen lassen, und dann die beteiligten Strukturen entlang der Reduktionsgraphen zurückzuverfolgen. Im vorliegenden Beispiel kommt man auf etwa ein Tausendstel der insgesamt erzeugten Liniensegmente, die als 'richtige' Fahrzeugkonturen übrigbleiben. In Abbildung 6.12c) sind alle erzeugten offenen Vierecke aus dem dritten Bild der Folge in grün dargestellt, und die aus dem Reduktionsbaum der besten Hypothese und aus diesem Bild in blau.

a) Ausgangsbild: Bild Nr. 3 aus BF1

b) Terminale aus diesem Bild



c) Resultat der 2D Verarbeitung

Figure 6.12: **Resultate des 2D Parsers anhand von Ausgangsbild nr. 3** in c) sind die beiden *open_quad* Instanzen aus dem Reduktionsbaum aus Abbildung 6.6 farbig hervorgehoben

Gemäß den Gedanken im Abschnitt 5.3 kann man sich eine Statistik der Instanzen aufgeschlüsselt nach den Symbolen machen. Im Ausgangsbild Nr. 3 der Bildfolge BF1 gibt es z. B. nach Leerlaufen der Warteschlange 11293 Teminale, 8047 Instanzen des Symbols *prol_line*, 3474 Instanzen des Symbols *angle* und 8043 Instanzen des Startsymbols *open_quad*. Abbildung 6.12c) zeigt diese Instanzen graphisch dargestellt. In b) sind die Terminale und in a) ist das Ausgangsbild selbst zu sehen. Mit der Gewinnung der Terminale aus den Bildern beschäftigt sich der folgende Abschnitt.

6.4 Gewinnung der Ausgangsdaten aus den Bildern

In der Abbildung 6.13 sind drei der acht Ausgangsbilder des Beispieldatensatzes BF1 zu sehen. Einzelheiten über die Modalitäten der Messungen sind dem Anhang F zu entnehmen. Die Standorte, von denen aus die Bilder aufgenommen worden sind, liegen in etwa auf einer geraden Linie und sind jeweils etwa zwei Meter voneinander entfernt. Abbildung 6.15 dokumentiert die Aufnahmesituation mit einer Skizze und einem Foto. Die Vermessungen am Ort konnten nur eine erste Näherung für die äußeren Kameraparameter ergeben, die dann jedoch als Eingangsdaten für etwas genauere Berechnungen genommen werden konnten. Hierfür gibt es Standardverfahren (siehe etwa [KONE]).

Im nachhinein wurde in die Bildfolge der kleine Baum aus einer anderen Bildfolge so hineinkopiert, wie er erscheinen müßte, wenn er zwischen Kamera und Fahrzeug gestanden wäre. So konnte zusätzlich demonstriert werden, daß das Objekt selbst dann gefunden werden kann, wenn es in keinem der Bilder ganz zu sehen ist, sondern überall partiell verdeckt ist.

Der Schwerpunkt liegt auf der symbolischen Verarbeitung bzw. darauf, zu demonstrieren, daß solche syntaktischen Verfahren auch mit größeren Eingangsdatensätzen und mit einem hohen Anteil an Störinformation noch funktionieren. Es wurde daher bewußt ein einfaches Extraktionsverfahren gewählt, das die Konturen im Bild durch sehr viele kurze Instanzen symbolisch beschreibt. Eine Konturausdünnung oder Skelettierung zwischen Kantenoperator und Konturbeschreibung, wie sie von den meisten Autoren als Zwischenschritt vorgeschlagen wird, um die nachfolgende symbolische Verarbeitung nicht mit einer Flut von Mehrdeutigkeiten zu belasten, ist nicht nötig[4]. Direkt aus den Grauwertbildern werden die Instanzen der terminalen Symbole gewonnen, indem die Grauwertgebirge an einer fixen Anzahl äquidistanter Schwellwerte binarisiert werden. Die entstehenden Konturen werden dann durch Polygonzüge aus kurzen geraden Lin-

[4]Was nicht heißen soll, daß man sie nicht einfügen könnte, wenn es darum geht, Rechenzeit und Speicherplatz zu sparen

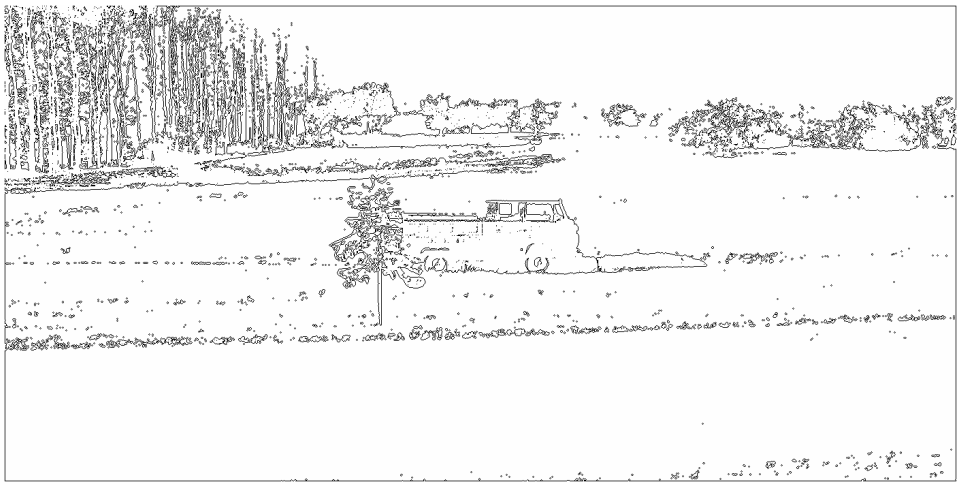
Ausgangsbild 2

Ausgangsbild 5

Ausgangsbild 8

Figure 6.13: **Beispielbildfolge BF1**

a) Ausgangsbild 1

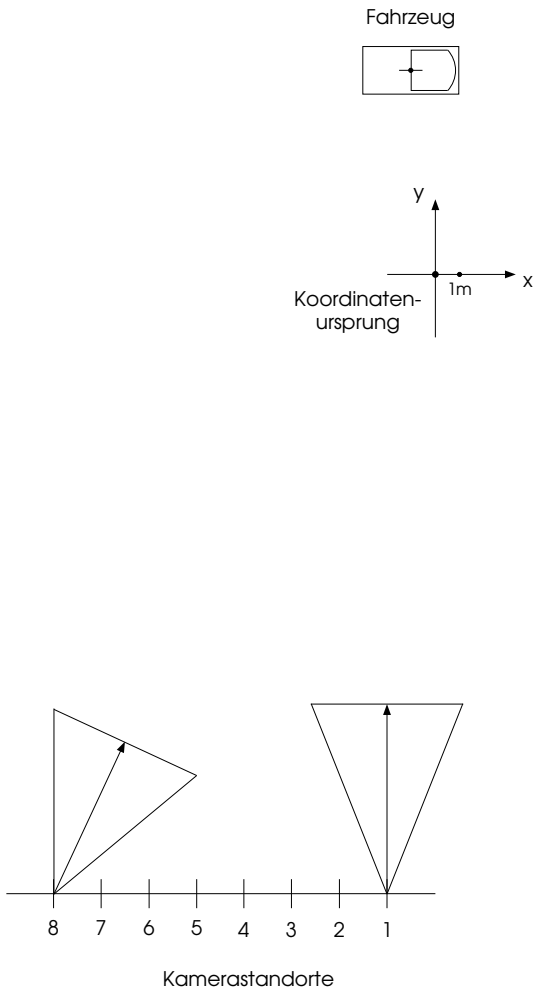


b) Binärkontur mit einem mittleren Grauwert als Schwelle

c) Rekonstruktion nach Polygonapproximation (12 Schnitte)

Figure 6.14: **Konstruktion der Terminale**

a) Fotographische Dokumentation



b) Skizze

Figure 6.15: **Aufnahme der Bildfolge BF1**

ienstücken approximiert (vgl. [LUE-86-1]). Mit vorliegender Parameterwahl werden sie maximal 20 Pixel lang. Pro Bild finden sich in der Bildfolge BF1 insgesamt ungefähr 20 000 solche Kontursegmente bei 12 Schwellwerten. Die Abbildung 6.14 dokumentiert das Verfahren anhand des ersten Ausgangsbildes. Man kann sich den datenreduzierenden Charakter solcher Verfahren verdeutlichen, indem man die durch die Polygone umschriebenen Segmente in dem der Schwelle entsprechenden Grauwert einfärbt und so das Bild 'rekonstuiert'. Übrigens ist man in der symbolischen Verarbeitung keineswegs auf gerade Linienstücke beschränkt, und kann natürlich auch runde Kontursegmente oder flächige Segmente verwenden. Es muß lediglich eine Beschreibungsmöglichkeit durch entsprechende Attribute gegeben sein.

6.5 Abschätzung der Leistungsgrenzen

Zur Verfahrensentwicklung stand nur die Bildfolge BF1 zur Verfügung. Um den Wert des Verfahrens abzuschätzen ist es erforderlich, das Verfahren - einschließlich der eingestellten Verfahrensparameter, wie sie im Anhang C dokumentiert sind - an anderen Bildfolgen zu testen. Man gewinnt so Anhaltspunkte, unter welchen Bedingungen bzgl. Objektentfernung, Wetter, Beleuchtung, Abbildungsqualität usw. das Verfahren noch bestimmungsgemäß funktionieren müßte. Seriöse Zahlen (Falschalarmraten und Entdeckungswahrscheinlichkeiten) können nur gegeben werden, wenn eine aus einer konkreten Aufgabenstellung heraus plausible a priori Wahrscheinlichkeit auf der Menge der Bildfolgen gegeben ist, und eine hinreichend große Stichprobe aus dieser Menge vorliegt. Solche Tests sind derzeit nicht möglich.

Statt dessen wurden zwei weitere Bildfolgen erstellt (siehe Anhang F) und zum Verfahrenstest verwendet. Abbildung 6.17 zeigt drei Bilder aus der Bildfolge BF2. Hintergrund und Vordergrund enthalten hier wesentlich mehr Struktur als in der Bildfolge BF1. Insbesondere gibt es andere Fahrzeuge und Objekte mit ähnlichen geometrischen Eigenschaften, wie sie das gesuchte Objekt aufweist. Insofern erscheint es bemerkenswert, daß keine falschen Instanzen des Startsymbols reduziert werden können. Korrekte Instanzen des Startsymbols *light_truck* werden aus dieser Bildfolge reduziert. Man muß allerdings gewisse Anpassungen am Modell vornehmen, da hier ein Fahrzeug neuerer Bauart vorliegt. Dieses hat in Fensterhöhe und Seitenwandhöhe andere Abmessungen. Das ist im Anhang B unter den Prädikaten *template_matching* und *geom_const* dokumentiert. Einer der Verfahrensparameter - der Schwellwert bzgl. Parallelität bei der Linienverlängerung - hat sich beim Test als ungeschickt gewählt herausgestellt (siehe Anhang C). Die beste Startsymbolinstanz ist einem Ausschnitt aus dem Bild 5 in der Abbildung 6.16 mit den entsprechenden Kameraparametern überlagert.

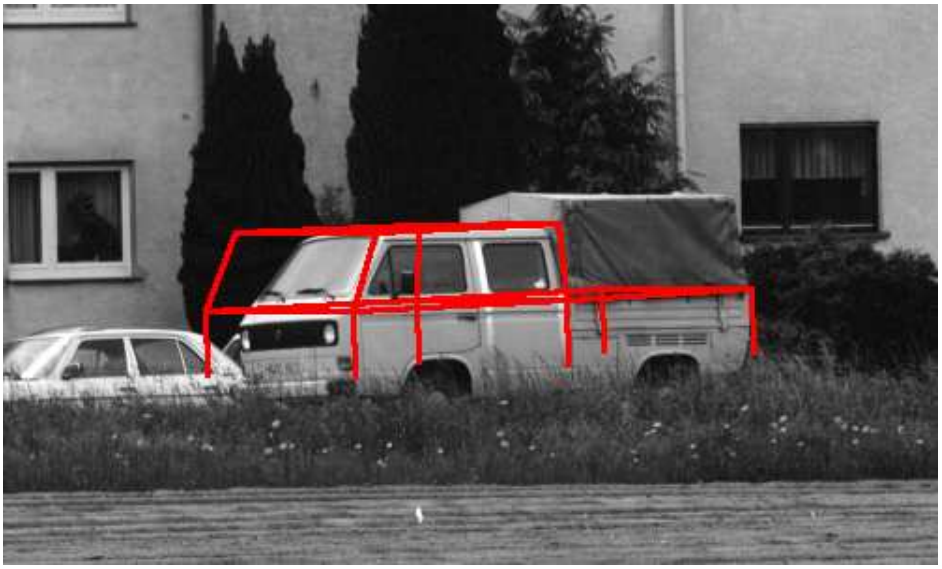


Figure 6.16: **Ergebnis mit Bildfolge BF2**

Die Bildfolge BF2 markiert in etwa die Grenze dessen, was durch das vorgestellte Verfahren noch korrekt unterschieden werden kann. Es erscheint keineswegs zwingend, daß unter diesen Bedingungen alle notwendigen Kantensegmente hinreichend sicher gefunden werden. Sind die Verhältnisse bzgl. Beleuchtung, atmosphärischer Streuung, Objektentfernung, Auflösung usw. wesentlich schlechter, so wird das Fahrzeug i. a. nicht mehr gefunden werden können, weil wichtige Segmente ausfallen. Dies demonstriert die Bildfolge BF3, von der drei Bilder in Abbildung 6.18 abgebildet sind. Insbesondere der Verlust an geometrischer und radiometrischer Auflösung durch etwa 350 Meter leicht dunstige Luft zwischen Kamera und Fahrzeug sind fatal. Diese Resultate sind zusammenfassend bereits in [MICH-96] publiziert.

Ausgangsbild 2

Ausgangsbild 5

Ausgangsbild 8

Figure 6.17: **Beispielbildfolge BF2**

Ausgangsbild 2

Ausgangsbild 5

Ausgangsbild 8

Figure 6.18: **Beispielbildfolge BF3**

Chapter 7

Diskussion

Exemplarisch wurde eine KG vorgestellt und diskutiert, die das Modell eines Fahrzeugs im Raum in seine Bestandteile zerlegt. Ein Parser wurde implementiert, der aus einfachen Primitivinstanzen im Raum Instanzen dieses Fahrzeugkonzeptes zusammensetzt. Insbesondere erscheint die hohe Robustheit in Anbetracht der Komplexität des Datenmaterials bemerkenswert. Der Aufbau solcher Systeme, ihr Wissenserwerb und die Erklärbarkeit ihrer Funktionsweise wird durch die mathematische Fundierung durch Koordinaten Grammatiken wesentlich erleichtert. Eine Betrachtungsweise im Zusammenhang mit (bzw. eine Einbindung von) statistischen Methoden wurde im Abschnitt 5.3 angedeutet. Der Ansatz reiht sich ein in die zahlreichen Versuche einer mathematischen Fundierung der Mustererkennung bzw. des Computersehens, ohne deren Allgemeinheitsanspruch insbesondere auch für Gebiete der Wahrnehmungspsychologie und Biologie zu teilen. Er sieht sich eher auf der Seite der Ingenieurwissenschaften.

Die vorgestellte Methode ist eher ungewöhnlich und nicht einmal besonders naheliegend. Man findet in der neueren Literatur kaum ähnliche Arbeiten. Die Grundlagen, auf die sie sich abstützt (etwa [ANDS-68-1, ANDS-68-2, MILG, ANDS-77]), sind eher alt. Ein wesentlicher Einwand liegt im großen Bedarf an Speicherplatz und Rechenaufwand begründet. Im Abschnitt 7.2 werden daher die von den heute sonst gängigen, veröffentlichten Ansätzen abweichenden Besonderheiten gerechtfertigt. Dazu wird im Abschnitt 7.1 zunächst beispielhaft ein solcher Standardansatz skizziert. Der Abschnitt 7.3 zieht dann noch Vergleiche mit einigen der bekanntesten anderen syntaktischen oder strukturorientierten Verfahren, und Abschnitt 7.4 erläutert den Zusammenhang mit den aus der KI für das Computersehen vorgeschlagenen Strukturen. Die vorliegende Arbeit versteht sich als Beitrag zu den eher mathematischen Grundlagen der Mustererkennung und des Computersehens, und muß daher auch an den auf diesem Gebiet erschienenen nichtsyntaktischen Ansätzen gemessen werden. Das

geschieht im Abschnitt 7.5. Natürlich gibt es inzwischen eine lange Tradition auf dem Gebiet der syntaktischen Mustererkennung, auf der die vorliegende Arbeit fußt. Der Abschnitt 7.6 umreißt kurz diese historischen Grundlagen.

7.1 2D Modellvergleichs Verfahren

Üblicherweise (z. B. in [BUN-89]) wird zum Erkennen von 3D Objekten auf Bildern etwa folgender Weg vorgeschlagen: Die Bilder werden mit einem Kantenoperator gefiltert. Das Ergebnis wird durch einen Schwellwert binarisiert und die enstehenden Konturbereiche verdünnt. Schließlich wird eine Liste mit geraden Konturstücken erstellt, die symbolisch beschrieben sind. Von dieser Liste wird ein Histogramm nach der Orientierung erstellt, oder man sucht darin (z. B. mit einem 2D Parser) nach auffälligen Winkeln oder anderen Gruppierungen. So erhält man Hinweise, wie das 3D Modell des gesuchten Objektes projiziert werden soll. Dann werden die Kanten des projizierten Modells mit den gefundenen Linien verglichen, und es ergibt sich ein Vertrauensmaß in diese Projektion. Die Parameter der Projektion erlauben es, zusammen mit den Kameraparametern, die Lage dieser Objekthypothese im Raum zu bestimmen. Nicht umsonst liegt hierbei immer hohe Betonung auf guten Vorverarbeitungsverfahren. Aus dem Bild soll möglichst ohne Informationsverlust eine Liste von maximal einigen hundert Linien erzeugt werden. Daraus ergeben sich dann einige wenige Hypothesen, die verfolgt werden können. Die Verfolgung einer einzelnen Hypothese ist recht aufwendig, denn die Projektion muß einen Algorithmus für Selbstverdeckung beinhalten, und wenn mit Teilstrukturen verglichen wird, hat man ein Untergraph Isomorphie Problem vor sich, das mit wachsender Größe des Modells sehr rasch sehr viel Aufwand mit sich bringt. Nur durch geschickt gewählte Heuristiken lassen sich diese Probleme umgehen. In nicht kooperativer Umgebung bekommt man, auch mit sehr ausgefeilter Vorverarbeitung, so viele Konturlinien, daß es weiterer Hinweise bedarf (z. B. liefern Flughöhe, Blickwinkel und das Wissen das Fahrzeuge aufrecht stehen erhebliche Einschränkungen). In [GROC] wurde ein Verfahren beschrieben, was recht zuverlässig einen solchen 2D Modellvergleich anwendet, wenn der ungefähre Ort im Bild bekannt ist. Bei den dort verwendeten Daten handelt es sich um Infrarotbilder, auf denen sich durch hot spot Detektion gut wenige verläßliche Hinweise auf Fahrzeuge finden lassen. Das gilt allerdings nur für bestimmte Aufnahmebedingungen und Betriebszustände der Fahrzeuge.

7.2 Rechtfertigung der KG

Dem FIM ist die Erforschung besonders komplexer Probleme der angewandten Mustererkennung übertragen, die so ohne weiteres nicht mit den Aufgabenstellungen aus dem industriellen Robotersehen und der Qualitätskontrolle oder der semiautomatischen Straßenverkehrsführung vergleichbar sind. Z. B. ist die Sensorik ganz anders und wesentlich vielfältiger. Meist ist die Umgebung eher destruktiv als kooperativ. Es hat sich gezeigt, daß ein Ansatz mit mathematisch

sauber fundierten, statistischen Methoden zwar für viele Spezialfälle (etwa das hotspot tracking in Infrarotbildfolgen) gute, in der Anwendung brauchbare Resultate, liefert. Der allgemeine Durchbruch zu echtem Computersehen ist mit diesen Methoden aber bisher nicht gelungen. Man suchte daher nach Alternativen.

Natürlich landete man bei der KI. An dieser Disziplin fällt vor allen Dingen die weitgehend heuristische Vorgehensweise auf. Kaum ist ein Problem angedeutet, schon ist auch ein KI Programm in den Rechner getippt, was die Sache 'löst'. Oft sind die Ergebnisse verblüffend oder gar bizarr (wie etwa im berühmten ELIZA Fall). Heute beschäftigt sich seriöse KI Forschung mit den Ursachen dafür, die eigentlich weniger im Programm als in der Psyche des Benutzers zu suchen sind. Besonders interessant sind die Mißverständnisse. Die Semantik von Datenstrukturen und Programmen rückt in den Brennpunkt der Aufmerksamkeit. Nach Auffassung des Verfassers begeht man einen Fehler, wenn man sich dabei wieder an die Tafel stellt, allerlei Kästchen und Kreise anzeichnet, sie mit Worten wie 'Wissen' oder 'Benutzer' versieht und durch allerlei Pfeile verbindet. Gefragt ist vielmehr Mathematik. Die Semantik der formalen Sprachen - sprich Modelltheorie - ist ein inzwischen weit ausgebautes - wenn auch etwas ungeliebtes - Fachgebiet in der Mathematik, an dessen Methoden man sich als Informatiker orientieren kann. Man sollte Definitionen und Formeln nicht scheuen. Sie sind auch und gerade in der Praxis hilfreich.

Erste Voraussetzung für semantische Untersuchungen ist eine klare Syntax, und im Gegensatz etwa zum Compilerbau fehlt diese auf dem Gebiet der Mustererkennung (des Computersehens) meist. Man darf sich daher nicht wundern, wenn 'strukturbasierte', also heute meist heuristisch programmierte, KI Verfahren auf Daten, an denen sie noch nicht getestet wurden, gelegentlich eigenartige Resultate zeigen. Abhilfe kann nur eine klare Sprache schaffen.

Der Abschnitt 7.3 zeigt einige Versuche auf, eine Syntax für automatisches Sehen zu definieren. Nach Auffassung des Verfassers scheitern diese Versuche an einem wesentlichen Mißverständnis: Man schuf syntaktische Strukturen, die sich an der Topologie und Geometrie gerasterter Bilder orientieren. Bildverarbeitung und Sehen im Sinne von Wahrnehmung, also von Bildverstehen, sind aber verschiedene Tätigkeiten. Die syntaktisch zu fassende Struktur ist eher die Struktur der zu erkennenden Gegenstände oder Konstellationen von Gegenständen. Damit variiert der Raum, in den die 'Vokabeln' und 'Sätze' der anzustrebenden Syntax einzubetten sind, von Anwendung zu Anwendung. So erscheint es nur folgerichtig, die sonst in der Syntax verwendeten Relationen *Konkatenation* (in Zeichenketten) oder *Nachbarschaft* (in Graphen oder Feldern) zu verallgemeinern zu einem Prädikat π, welches erst in der konkreten Anwendung zu spezifizieren ist. Die syntaktische Struktur wird auf Mengen definiert. Dieser Ansatz zeugt von der Überzeugung, daß Wahrnehmung eher ein konstruktiver Prozeß ist als ein passives Durchlaufen verschiedener Filter. Die (vorverarbeit-

eten) Sensordaten sind als *ungeordneter* Haufen von Instanzen gegeben, aus dem der Wahrnehmungsprozeß zueinander passende Teile heraussucht und in einer Art Puzzlespiel zusammensetzt. Was wie zusammenpaßt, läßt sich aus der Aufgabenstellung ableiten, wenn diese nur klar genug formuliert ist.

Eine berechtigte Frage mag lauten, warum man die durch die Topologie und Geometrie des Bildes gegebene Ordnung auf den Sensordaten zunächst gänzlich verwirft, um sie dann durch den Parser wieder zusammenzusetzen. Zugegeben ist dieses Vorgehen radikal, aber man sollte bedenken, daß das, was der Parser zusammensetzt, eine Szene von Objektteilen und Ojekten ist, die zueinander in komplexen Beziehungen stehen. Wieviel hat diese Struktur gemeinsam mit der Struktur im Bild? Anzunehmen ist, daß das von Anwendung zu Anwendung, von Sensor zu Sensor stark variiert. Insbesondere bei projizierenden Sensoren wird die Topologie der Szene im allgemeinen zerstört. Durch partielle Verdeckung können zusammenhängende Objekte als getrennte Segmente erscheinen. Andererseits können benachbarte Segmente des Bildes in der Szene weit auseinander liegen. Arbeitet der Sensor im sichtbaren Spektralbereich, so dominiert das reflektierte Licht die Strahlungsintensität, mit der eine Oberfläche im Bild erscheint. Dann ist der Grauwert mehr von der Beleuchtung als von der Albedo der Fläche abhängig, und die hat man im allgemeinen nicht unter Kontrolle, so daß man aus dem Grauwert allein gar nichts schließen kann. Man kann in solchen Bildern Figur und Hintergrund nicht trennen. Man kann nur lokale Kontursegmente extrahieren. Wie diese zusammenzusetzen sind, kann erst spezifiziert werden, wenn man weiß, was man sucht. Der Graph, der die Kontursegmente in ihren Nachbarschaften repräsentiert, wird mit dem Nachbarschaftsgraph der Oberflächensegmente eines gesuchten Gegenstandes zunächst sehr wenig gemeinsam haben.

7.3 Die KG im Vergleich mit anderen syntaktischen Verfahren

Es bestehen erhebliche Unterschiede zur klassischen Syntax der Zeichenketten. Abschnitt 7.3.1 verdeutlicht diese Unterschiede. Für Zwecke der Mustererkennung kann man eine 'semantische Ebene' zur syntaktischen Analyse hinzufügen, und gewisse eher numerische Attribute dort deutlicher repräsentieren als in der Syntax. Das wird in Abschnitt 7.3.2 diskutiert. Man kann versuchen, die Einschränkung auf 'eindimensionale' Zeichenketten zu verallgemeinern auf 'höherdimensionale' Mengen. Damit beschäftigt sich Abschnitt 7.3.3. Es gibt eine Reihe von sehr praktischen syntaxorientierten Verfahren, von denen als Beispiele die PDL nach Shaw in Abschnitt 7.3.4 und die Plex Grammatiken mit NABEs in Abschnitt 7.3.5 vorgestellt werden.

7.3.1 Stringgrammatiken

Klassische Stringgrammatiken arbeiten **ersetzend**, d. h. ihre Produktionen verwandeln eine Symbolkette in eine andere, indem sie eine Teilkette herausschneiden und sie durch eine andere ersetzen. Statt Symbolkette sagt man auch **Wort**. Den hinteren Teil des Wortes nach dem herausgeschnittenen Teil muß man dann übrigens, entsprechend der normalerweise unterschiedlichen Länge der herausgenommenen und eingesetzten Teilsequenz, umnumerieren. Die Grammatik insgesamt geht von dem Wort aus, das nur aus einem einzelnen Startsymbol besteht, und wendet solange passende Ersetzungen an, bis das Wort nurmehr aus Terminalen besteht. Dabei wird das Wort - üblicherweise aber nicht unbedingt immer - zunehmend länger. Die Menge aller so erzeugbaren Worte ist dann die **Sprache** zu der Grammatik. Alle anderen Worte gehören nicht dazu. Im allgemeinen Fall ist mit der Frage, ob ein gegebenes Wort durch eine gegebene Grammatik erzeugt wird oder nicht, ein nichttriviales Problem gegeben. Es gibt keinen Algorithmus, der dies löst (siehe [AHO]). Schon in [CHOM] wird daher eine Hierarchie von Grammatiken angegeben. Die zugehörigen Einbettungen und der Nachweis der Äquivalenz mit den entsprechenden Hierarchien aus dem Bereich der Automatentheorie und der Theorie der rekursiven Funktionen stellen Hauptresultate der theoretischen Informatik dar und finden sich etwa in [AHO, SCHÖ]. Im einzelnen unterscheidet man:

- Typ 0: Allgemeine Grammatiken (unrestricted grammars): Diese haben die Mächtigkeit von Turingmaschinen oder allgemein rekursiven Funktionen. Damit sind sie 'zu mächtig'. Es stellen sich die üblichen Wort- und Halteprobleme: Es kann kein Algorithmus angegeben werden, der definitiv entscheidet, ob ein Wort **nicht** zu einer solchen Sprache gehört. Es kann ein Algorithmus angegeben werden, der die Zugehörigkeit verifiziert. Man kann aber keine obere Schranke für die Zahl der dafür erforderlichen Rechenschritte aus der Länge des Eingabewortes abschätzen.

- Typ 1: Kontextsensitive Grammatiken (context sensitive grammars): Richtiger wäre eigentlich die Bezeichnung 'monoton', denn hier verlangt man, daß das Wort auf der rechten Seite nicht kürzer sein darf als das auf der linken Seite. Damit kann ein einfaches Verfahren angewandt werden, welches die Sprache, nach der Länge der Worte sortiert, auflistet. Man hat einen Algorithmus, der in vorhersagbarer Rechenzeit entscheidet, ob ein Wort darin ist oder nicht. Andersherum läßt sich jede primitiv rekursive Funktion als kontextsensitive Grammatik kodieren. Dies ist also die Mächtigkeit dieser Struktur. Das entsprechende Automatenmodell ist die nichtdeterministische, linear begrenzte Maschine (linear-bounded automaton). Für die Praxis ist das noch zu mächtig. Es fallen Verfahren darunter, die NP-vollständig sind, also Komplexitätsprobleme aufwerfen.

- Typ 2: Kontextfreie Grammatiken (context free grammars): Gemeint ist, daß es in einer kontextfreien Produktion keinen Wortteil gibt, der von der linken in die rechte Seite einfach kopiert wird. Dieser wäre dann der 'Kontext' der vorausgesetzt werden muß, damit die restliche Produktion anwendbar ist. Auch Produktionen, die einen solchen Teil (z. B. ein einzelnes Nichtterminal) nach links oder rechts 'durchschieben', fallen in diese kontextsensitive Kategorie. Kontextfreie Grammatiken kann man dagegen immer so umschreiben, daß die linke Seite aller Produktionen nurmehr aus einem Nichtterminal besteht. Der Reduktionsgraph wird dann baumförmig. Durch Übergang auf Normalformen, die bestimmte Rekursionszyklen vermeiden usw., können kumulative Tabellen-Parser konstruiert werden, die von niedriger polynominaler Komplexität (nach der Länge des zu parsenden Wortes) sind (siehe [EARL]). Das Äquivalent aus der Automatentheorie ist der Stapel Automat (pushdown automaton). Dies ist also die allgemeinste für größere Datenmengen praktisch handhabbare Komplexitätsklasse. Leider gilt: Gewisse eigentlich leicht zu beschreibende Sprachen - etwa $\{a^n b^n c^n;\ n = 1, 2, ...\}$ gehören nicht dazu.

- Typ 3: Rechts lineare Grammatiken (right linear grammars): Hier besteht die rechte Seite aller Produktionen entweder nur aus einem Terminal oder aus einem Terminal links und einem Nichtterminal rechts. Die zugehörige algorithmische Mächtigkeit ist die der endlichen Automaten (finite-state automaton) oder regulären Mengen. Solche Automaten scannen das Eingabewort von links nach rechts und halten spätestens am Wortende; sie sind also linear in ihrer Komplexität.

Damit ist üblicherweise in der Literatur die tiefste Stufe an Komplexität erreicht. Was darunter liegt, wird nicht mehr als 'syntaktisch' angesehen, weil es an Rekursivität mangelt. Für die theoretische Untersuchung von Mustererkennungsaufgaben erscheint es aber geboten, noch einen Schritt tiefer zu gehen:

- Typ 4: Nicht rekursive Grammatiken: Darunter sind Grammatiken zu verstehen, bei denen keine Produktion rekursiv wiederholt auf dieselbe Stelle im Wort angewendet werden kann (auch nicht 'über Umwege'). Es sollen also keine Generierungen der Form $A \overset{+}{\mapsto} \alpha A \beta$ möglich sein. [1] Da es immer nur endlich viele Produktionen gibt, ist dann sowohl die Tiefe der Reduktionsbäume als auch die Länge der Worte a priori beschränkt. Jede solche Grammatik kann ersetzt werden durch ein Lexikon, welches die Sprache auflistet. Die Komplexität ist 'constant time' - ein 'table look up' tut's.

[1] Diese Form der Grammatik ist übrigens nicht zu verwechseln mit den cycle-free grammars in [GONZ]. Dort werden lediglich *überflüssige* Produktionszyklen der Gestalt $A \overset{+}{\mapsto} A$ verboten.

Damit ist man streng genommen eigentlich nicht mehr bei den syntaktischen Strukturen, wenn man definiert: **Syntaktisch** ist eine Methode dann, wenn sie mit endlichem Vokabular und endlichem Produktionssatz Elemente unendlicher Mengen (Sprachen) unterscheiden kann in 'well formed' und 'non well formed'. Häufig wird man daher lieber von 'strukturorientierter' Mustererkennung sprechen, und dies wie folgt präzisieren: **Strukturorientiert** ist eine Methode dann, wenn sie mit kleinem Vokabular und kleinem Produktionssatz Elemente sehr großer (z. B. kombinatorisch beschriebener) Mengen unterscheiden kann in 'well formed' und 'non well formed'. Was damit gemeint ist, sieht man am besten im Abschnitt 3.1. Bei den nicht rekursiven String Grammatiken tritt die Kombinatorik nicht so zu Tage, und eine Auflistung der Sprache in einem Lexikon erscheint möglich. Dagegen ist es definitiv unmöglich, alle Bildfolgen aufzulisten, die den Schluß auf das Vorhandensein eines Transporters gemäß der im Abschnitt 6 definierten KG zulassen, obwohl diese Grammatik ein sehr kleines Vokabular und vergleichsweise wenige Produktionen enthält, endlich attributiert ist und nicht rekursiv.

Im folgenden werden noch einige wichtige weitere Unterschiede zwischen der KG und der Stringgrammatik nach Chomsky aufgeführt: Insbesondere das Umnumerieren des ganzen nachfolgenden Strings bei einer Ersetzung, aber auch das Ausgehen von dem einstelligen Wort (g) unterscheidet die Stringgrammatiken erheblich von der Koordinaten Grammatik. Es handelt sich um völlig verschiedene Begriffe. Natürlich kann man eine eindimensionale KG schreiben und die Koordinate dann als Position in der Zeichenkette auffassen. Das liefert aber nur für isometrische Grammatiken eine Einbettung (siehe Abschnitt 2.3.3). Ansonsten können zwei verschiedene Instanzen einer KG durchaus an einer Stelle stehen, und es kann andererseits Lücken geben. Jede nicht isometrische Produktion einer Stringgrammatik andererseits verschiebt die Stellung aller nachfolgenden Symbole, die in der Produktion gar nicht auftauchen. Das läßt sich in einer KG so ohne weiteres nicht formulieren.

Klassische kontextfreie Stringgrammatiken wurden für Mustererkennungsaufgaben schon sehr früh vorgeschlagen (siehe z. B. [NARA-64]). Definitionen von Grammatiken zur Klassifikation von Chromosomen sowie ein interessanter stochastischer Parser dafür finden sich in [FU-72]. Diesen frühen syntaktischen Ansätzen ist gemeinsam, daß das Durchschauen der Arbeitsweise der Grammatiken, schon wenn diese noch gar nicht so umfangreich sind, recht mühsam sein kann. Es ist nicht klar, wie man aus einem Konzept für ein zu erkennendes Objekt die entsprechende Sprache entwickelt, und vor allem nicht, wie dann aus der Sprache die passende Grammatik zu konstruieren ist. Einmal programmiert, läßt sich kaum noch etwas an so einem System warten oder verändern.

7.3.2 Attributierte Grammatiken

Eine besondere Form der Erweiterung des Begriffs der (üblicherweise kontextfreien) Stringgrammatiken ist die Definition der *attributierten* Grammatik etwa in [KNUT, MAHN]. Durch eine zusätzliche 'semantische' Ebene wird die Ausdruckskraft der syntaktischen Struktur erhöht. Die Bezeichnung stammt daher, daß man die zugeordneten Bedeutungen 'Attribute' nennt. In [KNUT] ist erstmals mathematisch untersucht, wie die entsprechenden Attributierungsstrategien definiert werden müssen, damit keine zirkulären oder inkonsistenten Attributierungen auftreten können. Man spricht daher auch von 'knuthscher Semantik'. Diese für die Zwecke des Compilerbaus weitverbreitete Begriffsbildung dient der semantischen Analyse bereits erzeugter Reduktionsbäume. Dabei wird dann untersucht, ob Variablennamen schon definiert sind, ob Zuweisungen kompatibel sind usw. In [MAHN] findet sich eine etliche Seiten lange Literaturliste zu diesem Thema.

Gelegentlich wurde versucht, auch mit dem Ansatz der knuthschen Semantik Mustererkennung zu betreiben. Der Klassiker auf diesem Gebiet ist [TSAI]. Darin werden wiederum Chromosomen klassifiziert. Dadurch, daß man den Symbolen, die für die geraden Teile der Kontur eines Chromosomarmes stehen, das Attribut 'Länge' zuordnet, wird der syntaktische Teil der Analyse viel einfacher und vor allen Dingen einleuchtender und anschaulicher. Wichtige numerische Information wird in den Attributen abgelegt, die Grammatik verliert ihre Rekursivität und gewinnt gleichzeitig an Durchschaubarkeit und Aussagekraft für den Benutzer. Insofern ist der Ansatz dem der KG verwandt. Der logische Teil der Information verbleibt aber in der Konkatenation der Symbole. Der Reduktionsbegriff der attributierten Grammatik gilt nicht für eine Symbolmenge, sondern für eine Symbolstruktur - üblicherweise eine Kette, die z. B. eine Kontur beschreibt. Dies geht deutlich aus folgendem Zitat aus [TSAI] hervor:

- "Given an input pattern for classification, after preprocessing, all necessary primitives and their attributes are extracted according to some prespecified procedures. ... The next step is to transform the primitive set into a structural representation, such as a string, a tree, or a graph, by assigning symbols to primitives, selecting concatenating directions, and any other prespecified relations. The resulting representation is then analyzed syntactically by using the syntactic rules of the attributed grammars, while the semantic computation is performed simultaneously to obtain all required nonterminal (subpattern) attributes according to the semantic rules. ... "

Für die Anwendung einer attributierten Grammatik müssen die Symbole schon in den richtigen Zusammenhang gebracht sein, bevor die syntaktische Analyse stattfinden kann. Das mag bei mikroskopischen Aufnahmen von gefärbten Schnitten noch einfach sein. Bei Aufgaben, wie der im Kapitel 6 beschriebenen Detektion

und Lokalisation von bestimmten Fahrzeugen im sichtbaren Spektralbereich in natürlicher Umgebung, ist aber genau dieses 'in den richtigen Zusammenhang bringen' der Primitive der eigentlich schwierige Teil der Analyse.

Im Übrigen war die Attributierung von Knuth als zusätzliche semantische Ebene gedacht. So hat ein Bezeichner in einer Programmiersprache eine Bedeutung - es handelt sich um eine Konstante, Variable, Funktion usw. von einem bestimmten Typ. Nach der syntaktischen Analyse - die sollte aus Komplexitätsgründen maximal kontextfrei sein - wird die semantische Konsistenz geprüft. Dabei ist ganz wichtig, in welchem Kontext welcher Bezeichner verwendet wird. Man arbeitet auf dem schon erzeugten Reduktionsbaum und vermeidet so den durch die Kombinatorik verursachten Aufwand. Einem Symbol, das für ein gerades Kontursegment steht, seine Länge als Bedeutung zuzuweisen, bedeutet dagegen schon eine ziemliche Deformation dessen, was man gemeinhin unter Semantik versteht. Natürlich kann man auf diese Weise sehr einfach eine etwas unübersichtliche kontextsensitive Grammatik, die $\{a^n b^n c^n;\ n = 1, 2, ...\}$ erzeugt, verwandeln in eine triviale Grammatik, die nur das Wort abc erzeugt. Das n setzt man einfach als Attribut und fragt ab, ob es auch überall gleich groß ist. Was man damit hauptsächlich erreicht, ist ein Durcheinanderwerfen und Verbiegen der Begrifflichkeiten. Das ist nicht sonderlich hilfreich.

7.3.3 'Höherdimensionale' Grammatiken

Die Zeichenkettengrammatik ist für die Beschreibung von Strukturen in Sprachen konstruiert worden. Dabei bezeichnet 'Sprache' eine Menge von Worten. Und ein Wort ist ein n-Tupel von Symbolen aus einem endlichen Alphabet. Man kann nun auf die Idee kommen, ein solches Wort sei doch eine Art eindimensionales Muster, man brauche den Begriff der Zeichenkettengrammatik nur zu verallgemeinern auf mehrdimensionale, insbesondere zweidimensionale Muster, also auf Felder, Bäume und Graphen von Symbolen, um dann damit strukturorientierte Mustererkennung zu machen. In der Tat gibt es in der Literatur über syntaktische Methoden der Mustererkennung (etwa in [FU-82, GONZ, ROSE-90]) die dementsprechenden Begriffsbildungen der 'array grammars', 'tree grammars', 'web grammars' usw. Dabei ist es eigentlich nicht gerechtfertigt, Strukturen wie Bäumen oder Graphen die Dimensionszahl Zwei zuzuordnen, bloß weil man sie zweidimensional veranschaulicht oder weil sie Nachbarschaftsbeziehungen in Bildern codieren. Im einzelnen werden folgende Strukturen diskutiert:

- Zweidimensionale Feld Grammatiken haben einige Aufmerksamkeit beansprucht, weil man, nicht zuletzt durch die Physiologie der Nervengewebe beeinflußt, leicht auf die Idee kommen kann, zweidimensional indizierte Automatennetze, bei denen jeder Automat auf einem beschränkten Fenster aus dem Bild arbeitet, seien die natürliche Lösung für Probleme

des Computersehens. Man kann allerdings hier nur eine 'Symbolkachel' durch eine andere gleicher Größe ersetzen. Das Analogon dazu wäre bei den Stringgrammatiken die isometrische Grammatik. Dieses Gebiet ist so 'pixelnahe', daß eher eine Verwandschaft mit gewissen Skelettierungs- oder nichtlinearen Glättungsalgorithmen aus der morphologischen Bildverarbeitung ins Auge springt, als mit den Methoden der strukturorientierten Mustererkennung. Hier hat man es mit Prozessen zu tun, die sich flächig über eine 'Retina' ausbreiten. Unter Fachleuten für ikonische Bildverarbeitung gelten solche, meist hochgradig nichtlinearen und rückgekoppelten Filter, als ziemlich fragwürdig, weil instabil.

In [ROSE-89-1, NAKA-89, NAKA-95] werden theoretische Untersuchungen bzgl. lokaler KGs angestellt, die äquivalent sind zu isometrischen Feldgrammatiken der Dimension $d \geq 2$. Es ergeben sich eine Reihe interessanter, theoretischer Resultate. Es werden aber keine möglichen Anwendungsbeispiele diskutiert.

- Baum Grammatiken unterscheiden sich von String Grammatiken in folgenden wesentlichen Punkten: Man ersetzt Unterbäume in Bäumen durch andere Unterbäume. Die Konkatenation ist ersetzt durch die Vater-Sohn Relationen. Mit einem Knoten muß man alle direkten und indirekten Nachfolger ersetzen. Das führt zu unübersichtlichen Möglichkeiten. Man beschränkt sich daher auf die expandierenden Baum Grammatiken, bei denen die Nichtterminale immer nur als Blätter auftauchen. Man hat solche Systeme für die Analyse von Blasenkammeraufnahmen vorgeschlagen [FU-73], da sie für diese Problemklasse besonders geeignet erscheinen. Durch eine Konstruktion analog den Präfixnotationen läßt sich eine Einbettung in die kontextfreien Stringgrammatiken erreichen.

- Netz Grammatiken (web grammars) arbeiten auf ungerichteten Graphen. Das wirft ein schwieriges Problem auf. Wenn man einen Untergraphen aus einem Graphen durch einen anderen ersetzen will, muß spezifiziert werden, mit welchen Knoten des Rest Graphen (host web) die Knoten des einzusetzenden Graphen verbunden werden sollen. Zur Beschreibung z. B. der Menge aller möglichen organischen Verbindungen in Strukturformeldarstellung kann man versuchen, solche Grammatiken als Grundlage zu verwenden. Man wird allerdings in der Praxis beim Versuch danach zu parsen sehr rasch in problematische Rechenkomplexitäten hineingeraten. Das liegt aber wohl nicht am strukturorientierten und lokalen, also syntaktischen, Vorgehen, sondern in der Natur der Sache selbst.

Im Abschnitt 2.3.3 dieser Arbeit ist gezeigt, wie unter anderem auch diese Grammatiken eingebettet werden können in Koordinaten Grammatiken. Dabei sind

keine sehr komplizierten Codierungsumwege nötig. Vielmehr bleibt die algorithmische Gestalt zugehöriger Generierer und Parser erhalten. Man kann sogar an den Einbettungen die Besonderheiten dieser Strukturen deutlich machen (z. B. in wie weit sie globalen Kontext verlangen). Eine besondere Variante, die Ähnlichkeiten mit Koordinaten und Netz Grammatiken aufweist, wird unter der Bezeichnung Plex Grammatik diskutiert. Darauf geht Abschnitt 7.3.5 noch genauer ein.

7.3.4 PDL Systeme

Als Grundlage für ein strukturorientiertes Bildanalyseverfahren zur automatischen Generierung von Szenenbeschreibungen aus Blasenkammerfotos wurde Ende der sechziger Jahre in Standford am Linear Beschleuniger Zentrum von A. C. Shaw ein ebenso einfaches wie richtungweisendes System vorgeschlagen. Dieses PDL (*picture description language*) System wird auch in späteren Veröffentlichungen (z. B. in [FU-82]) gelegentlich vorgestellt. Dabei haben die als *primitive* bezeichneten Instanzen jeweils ein Kopf- und ein Fußende (*head* und *tail*), und man betrachtet die entsprechenden Möglichkeiten der Nachbarschaft zwischen diesen ausgezeichneten Punkten der Primitive als 'binäre Konkatenations Operatoren'. So steht $a + b$ für Kopf von a an Fuß von b, $a \times b$ für Fuß an Fuß, $a - b$ für Kopf an Kopf und $a * b$ für sowohl Kopf an Kopf als auch Fuß an Fuß. $\sim a$ kehrt die Richtung von a um. Der Kopf wird zum Fuß, der Fuß zum Kopf. Im System PDL werden also im wesentlichen die in Abschnitt 5.2.3 erörterten Symmetrien und die damit verbundenen Kombinationsmöglichkeiten explizit. Das ist sicher ein sonst häufig vernachlässigter Punkt. Es tut sich aber schwer mit Invarianzen über die Translation hinaus (etwa Rotation, Skalierung und Perspektiven oder gar partieller Verdeckung usw.) und mit Objekten, die durch mehr als zwei Orte im Bild zu beschreiben sind. Darüberhinaus orientiert es sich an der Topologie des Bildes, nicht an der Szene. Jedes PDL System läßt sich leicht auf natürliche Weise in eine KG übersetzen. Die Umkehrung gilt natürlich nicht. Es bleibt zu bemerken, daß Shaws Ansatz auf genial einfache Weise denselben wesentlichen Punkt, den auch die vorliegende Arbeit betont, angeht, daß nämlich Wahrnehmung ein Konstruktionsprozeß ist, der aus einem Puzzlespiel die richtigen Kombinationen herausfiltert, indem er durchprobiert, was das Wissen über die Szene an Möglichkeiten zuläßt. Die Blasenkammerbilder waren dafür ein sehr gutes Beispielfeld.

7.3.5 Plex Grammatiken mit NAPEs

Die PDL von Shaw läßt als Anlagerungspunkte an einer Instanz nur zwei Stellen zu, nämlich Kopf und Fuß der Instanz. Daß dies eine Beschränkung ist, die

gewissen Anwendungen - etwa der Generierung und dem Parsen von Formeln aus der organischen Chemie - nicht gerecht wird, war genauso rasch einzusehen, wie klar wurde, wie man hier durch Modifikation der Definition Abhilfe schaffen kann. Die Verallgemeinerung auf n Anlagerungsstellen wird in der Literatur unter NAPE geführt, was für n *attaching point entity* steht. Die Definition der Plex Grammatiken, die aus Produktionen bestehen, die solche NAPEs ersetzen, geht auf [FEDE] zurück. Das wesentliche ist in [GONZ] zusammengefaßt. Hauptsächlich wird diese Struktur eben für Formelparsing für die organische Chemie vorgeschlagen. Man findet aber auch neuere Literatur, z. B. über den Einsatz bei der Analyse von Maschinenbauzeichnungen ([COLL]). Der wesentliche Vorteil gegenüber starr an der Bildgeometrie orientierten Vefahren liegt darin, daß man sich mit der Topologie an den zu betrachtenden Objekten orientieren kann. Man ist keineswegs auf die Ebene beschränkt.

Formal hat eine Plex Grammatik die Gestalt (T, N, P, g, I), wobei die Terminale T, Nichtterminale N, Produktionen P und das Startsymbol $g \in N$ wie üblich definiert sind. Es fehlt ein Attributraum. Statt dessen gibt es die Menge der Anlagerungsstellen I. Jedem NAPE aus $T \cup N$ ist eine Untermenge der Anlagerungsstellen I zugeordnet. Schreibt man die Produktionen wieder - wie in dieser Arbeit üblich - in reduzierender Richtung auf, so steht die rechte Seite links und die linke Seite rechts: $(\Sigma \to \Lambda) \in P$, wobei rechte und linke Seite nichtleere Worte sind, und in der linken Seite nicht nur Terminale stehen dürfen. Dazu muß in der Produktion zu beiden Seiten je eine Verbindungsliste Γ und eine Anlagerungsliste Δ definiert sein. Eine Verbindungsliste zu einem Wort der Länge k ist eine k^2 Matrix mit Werten aus $\mathcal{P}(I)$. Sie gibt an, welches NAPE mit welchem anderen über welche Anlagerungsstellen verbunden ist. Wort und Verbindungsliste beschreiben also eine *Struktur* von NAPEs. Grammatiken ersetzen immer eine Teilstruktur einer größeren Struktur durch eine andere Teilstruktur. Es bleibt also noch zu spezifizieren, wie die Verbindungen der einzufügenden Teilstruktur zur Reststruktur in Abhängigkeit von der zu entfernenden Teilstruktur zu konstruieren sind. Dies spezifizieren die Anlagerungslisten. Diese beiden Listen einer Produktion müssen natürlich gleich lang sein. Somit gewinnen die Produktionen die Gestalt

$$(\Sigma, \Gamma_\Sigma, \Delta_\Sigma) \to (\Lambda, \Gamma_\Lambda, \Delta_\Lambda).$$

Man muß noch gewisse Zusatzbedingungen stellen, wie z. B.: Man kann kein NAPE an sich selbst anlagern; alle Anlagerungsstellen der NAPEs in einer Produktion müssen belegt sein, entweder in der Verbindungsliste oder in der Anlagerungsliste.

Die Einbettung einer solchen Plex Grammatik in die KGs, wie sie im Abschnitt 2 definiert sind, ist nicht ganz trivial. Im Attributraum kann man die Potenzmenge $\mathcal{P}(I)$ ansetzen. Die Verbindungsliste Γ_Σ einer Produktion kann im Prädikat π untergebracht werden. Dazu benötigt man im Attributraum noch eine Index-

menge (z. B. die natürlichen Zahlen), da in der Plex Grammatik Sprache ein NAPE natürlich mehrfach auftreten darf. Die anderen Listen und die Zusatzbedingungen muß man dann in der Funktion ϕ unterbringen. Dabei muß den neu entstehenden NAPEs ein noch nicht belegter Index zugeordnet werden, was eine globale Konsistenzbedingung darstellt. Oft wird bei den Plexgrammatiken unter den Zusatzbedingungen ohnehin globale Konsistenz gefordert in dem Sinne, daß außer den in den Listen spezifizierten Verbindungen und Anlagerungen keine weiteren existieren dürfen. Es ergeben sich also ähnliche Probleme, wie bei der Einbettung der Graphgrammatiken. Umgekehrt ist eine Einbettung der KGs in die Plex Grammatiken wohl kaum möglich. Insbesondere fehlt die Möglichkeit, einen geometrischen Attributraum darzustellen.

Trotz dieser formalen Unterschiede springt in der Praxis eine gewisse Verwandschaft ins Auge. Aus den vorangegangenen Betrachtungen dieser Arbeit, insbesondere im Abschnitt 5.2.3, geht hervor, daß in den praktisch in Frage kommenden KGs hauptsächlich Nachbarschaften verwendet werden, und die Bedingungsprädikate der Produktionen im wesentlichen nur bestimmen, welche Attribute welcher Instanzen der Ausgangskonfiguration miteinander benachbart sein sollen. In der Definition der Plex Grammatiken wird diese kombinatorische Seite der konstruktiven Wahrnehmung hervorgehoben. In ihrer reinen Form können sie überhaupt nichts anderes. Sie sind der KG wesentlich verwandter als etwa die Graph Grammatiken, z. B. weil die Anzahl und Bezeichnung der Anlagerungsstellen in der Definition festgelegt wird. Wenn man die Plex Grammatiken in der Praxis ein klein wenig erweitert, indem man den einzelnen NAPEs einen Ort in einer Szene oder einem Bild zuordnet, so kommt man zu sehr ähnlichen Strukturen. Die Beschreibung mit Plex Grammatiken hebt den topologischen Aspekt hervor, wohingegen mit Koordinaten Grammatiken der geometrische Aspekt betont wird.

7.4 Vergleich mit KI Methoden zur Mustererkennung

In diesem Abschnitt werden Unterschiede und Gemeinsamkeiten des vorgestellten Ansatzes mit Verfahren verdeutlicht, die von der KI Gemeinde für das Computersehen vorgeschlagenen werden. Als Beispiel dient dafür der Ansatz mit *semantischen Netzen* und A^*, wie in [SAGR] durchgeführt. Die Bezeichnung 'semantische Netze' kommt aus der Linguistik. Zunächst muß also klargestellt werden, daß diese Strukturen nichts mit der Knuth'schen Semantik zu tun haben, auf die sich der Abschnitt 7.3.2 bezieht.

Sagerer unterscheidet mit R. J. Brachman neben der Implementierungsebene, der logischen Ebene und der Ebene der Konzepte noch die epistemologische Ebene

und die Ebene der problemabhängigen, intensionalen Beschreibung. Dabei legt er Wert auf 'Neutralität, Adäquatheit und wohldefinierte Semantik' als Kriterien für die Qualität eines semantischen Netzes, schreibt aber in [SAGR] zur Frage der Semantik (Seite 47):

- "... In einem assoziativen Netzwerk müssen die Bedeutungen der Grundelemente und die Operationen explizit definiert werden. Bis zur epistemologischen Ebene ist diese Forderung nachprüfbar. Bei der konzeptuellen Ebene würde dies eine formal definierbare Semantik des Problemkreises erfordern, was in den meisten Fällen nicht möglich ist..."

Eine sicher wichtige und nützliche Unterscheidung ist bei Sagerer diejenige in den **intensionalen** Aspekt der Konzepte, und den **extensionalen** Aspekt der Instanzen. Wenn man etwa für das Symbol *Winkel* aus der Beispielproduktion im Abschnitt 2.1.3 die Produktion umformuliert in ein KI Konzept, welches zerfällt in zwei Teilkonzepte, zwischen denen die Relation π - in diesem Falle Nachbarschaft und Antiparallelität - besteht, so erkennt man, daß praktisch relevante Symbole und Produktionen einer KG eine Intention, eine Bedeutung haben, die der Benutzer gemeint hat. Dem steht eine Extension gegenüber. Bei den Koordinaten Grammatiken setzt man dafür *Winkel* als Startsymbol, und hat dann die Extension in Gestalt der zugehörigen Sprache $\mathcal{L}_{noisy}$. Dies entspricht dem, was man in der Mathematik unter Semantik verstehen würde. Danach ist die Extension des Konzepts *Winkel* die Menge aller Bilder, in denen sich eine Instanz dieses Symbols findet. Bei der Formulierung einer KG und damit einer formalen Sprache fällt die von Sagerer geforderte Semantik also automatisch mit ab.

In der KI hat man aber gelegentlich auch den Eindruck, daß als Extension eines Konzeptes die Menge aller zugehörigen Instanzen in einem konkreten Datensatz (Beispielbild) angesehen wird. O. Grau schreibt z. B. in [GRAU] (Seite 247):

- "Die Wissensbasis enthält eine generische Prototypenbeschreibung der zu modellierenden Objekte in Form eines semantischen Netzes. Die Interpretation weist den Merkmalen und Bildprimitiven, die in der Bildverabeitungs-Pipeline extrahiert wurden, eine Bedeutung zu. Dazu wird ein Szenengraph ebenfalls in Form eines semantischen Netzes aufgebaut."

Dann hängt die Bedeutungszuweisung nicht nur von dem Konzept und dem sematischen Netz ab, von dem es ein Teil ist, sondern auch von dem konkreten Datensatz. Ob die Intention und die Extension übereinstimmen, hängt von der Wahl des Beispiels ab.

Am augenfälligsten berühren sich die Strukturen, die hier unter der Bezeichnung KG durchgeführt sind, mit den semantischen Netzen der KI in der Definition der Nachbarschaftsgraphen in Abschnitt 5.2.1. Man könnte also die Liter-

atur, die Überlegungen und die Programme, die unter dem Stichwort 'semantische Netze' zu finden sind, an dieser Stelle einbinden, um die Konstruktion der Nachbarschaftsgraphen, die Konstruktion der Produktionsnetze und letzlich der lauffähigen KGs aus der Aufgabe heraus teilautomatisch zu unterstützen, zu systematisieren oder gar zu automatisieren, also einen Algorithmus dafür zu finden. Dieses Problem wird in der Literatur unter *grammar inference* diskutiert. Der Verfasser stellt sich auf den Standpunkt, daß hierfür in Bezug auf die KGs noch nicht genügend Erfahrung vorhanden ist. Es gibt noch keine 'Experten', die zu modellieren wären. Da aber in [NIEM-90, SAGR, BUN-85] unter der Bezeichnung 'semantische Netze mit A^*-Kontrolle' auch Verfahren zur Mustererkennung diskutiert werden und das entsprechende Programm ERNEST von Kummert und Sagerer auch durch F. Quint auf vergleichbaren Daten getestet wurde [QUIN], kann ein Verfahrensvergleich durchgeführt werden.

Beispielhaft sei hierzu zunächst der Nachbarschaftsgraph aus Abbildung 5.4 herangezogen. Die Knoten und *Teil-von* Kanten kann man direkt als semantisches Netz interpretieren. Es handelt sich dann um einen Baum mit einem nicht primitiven Konzept V als Wurzel und vier primitiven Konzepten l_1,...,l_4 als Blättern, die damit durch *Teil-von* Kanten verbunden sind. Die durch die *benachbart* Kanten im Nachbarschaftsgraphen angedeuteten Relationen, werden im semantischen Netz durch eine Testprozedur ersetzt, die überprüft, ob ein Quadrupel entsprechender l_i Instanzen ein Viereck bilden. Für die primitiven Konzepte l_i gibt es jeweils eine Prozedur, die alle ihre konkurrierenden Instanzen auflistet. Als Ziel der Analyse wird eine Instanziierung des Konzeptes V gesetzt, als Daten sei eine Liste mit n Linien Instanzen gegeben. Der Tatsache, daß es konkurrierende Instanziierungen eines Konzeptes geben kann, trägt die Kontrolle Rechnung, indem sie einen **Baum aus Suchzuständen** aufbaut. Ein Suchzustand ist ein Teil des Konzeptnetzes, dessen Knoten zum Teil instanziiert sind. Eine Kante gibt es hier vom Zustand z_1 zum Zustand z_2, wenn in z_2 genau ein Konzept, welches in z_1 noch nicht instanziiert war, eine Instanz zugewiesen bekommt, und alle anderen Instanziierungen identisch sind. Ein Blatt ist erreicht, wenn das Zielkonzept instanziiert wird. Ausgehend vom Zielkonzept expandiert die Suchkontrolle alle notwendigen primitiven Konzepte (in diesem Falle vier) und verzweigt sich dann in einem exponentiell wachsenden Baum. In diesem Beispiel werden n^4 Zustände erzeugt, die jeweils die Test-Prozedur für Vierecke aufrufen. Mangels Bewertungskriterien kann der Suchbaum nicht beschnitten werden. Das Verfahren leistet in etwa dasselbe, wie die Aufzählung aller Ausgangskonfigurationen für diese Produktion der KG.

Interessanter wird es, wenn man den alternativen Nachbarschaftsgraphen aus Abbildung 5.5 zugrunde legt. Hier werden für die *Winkel* zunächst Zwischenkonzepte expandiert. Die Frage ist dann, ob es erlaubt ist, daß hier zweimal dasselbe Konzept eingesetzt werden darf. Insbesondere ist zu klären, ob eine bereits durchgeführte Instanziierung verwendet werden kann auf einem anderen Zweig

des Suchbaums. Man sollte das Konzept nicht als Knoten in einem Graph sehen, sondern als Datenstruktur, als 'Frame' mit einem 'Slot' für eine Liste von Teilkonzepten. In solch einer Liste darf ein Konzept mehrfach stehen. Es darf nur keine direkten oder indirekten Rekursionen geben, so daß ein Konzept Teilkonzept von sich selbst ist. Die Kontrolle expandiert mithin ausgehend vom Zielknoten V zunächst das Konzept für *Winkel* W. Da dieser Knoten nicht instanziiert ist, wird er expandiert, und man gelangt zum Konzept für Linie l. Dies ist ein primitives Konzept, kann also instanziiert werden durch die Aufzählungsprozedur, und man erhält hier eine Suchbaumverzweigung mit n konkurrierenden Zuständen. Jeder dieser Zustände wird weiterverfolgt, indem jetzt der zweite Teil-von Slot des Konzeptes W expandiert wird. Dieser verweist auf dasselbe Konzept l, das bereits n-Fach konkurrierend instanziiert ist. Damit sind alle Teile des nicht primitiven Konzeptes W instanziiert. Es gibt n^2 konkurrierende Zustände, die jeweils die Prozedur zur Winkelverifikation mit einem konkreten Instanzenpaar aufrufen. Wieviele konkurrierende Instanziierungen des Konzeptes W erzeugt werden, hängt von den Daten und der Verifikationsprozedur ab. Ist diese so formuliert, daß sie das entsprechende Prädikat zur *Winkel* Produktion der KG implementiert, so gibt es genauso viele konkurrierende Zustände der Suche wie *Winkel* Instanzen beim kumulativen Parserlauf der KG. Sei $m \leq n^2$ die Anzahl dieser Suchzustände. Da der erste Teil-von Slot des Zielkonzeptes V nun instanziiert ist, springt die Kontrolle zum zweiten Teil-von Slot und damit wieder zum selben Konzept W. Das stellt sich jetzt als bereits instanziiert heraus. Es gilt also nun die m^2 konkurrierenden Zustände mit der Prozedur zu testen, die ein Paar von konkret instanziierten W Konzepten auf das Vorliegen der Voraussetzungen für das V Konzept testet. Danach ist das Zielkonzept instanziiert. Man erkennt: Im Grunde läuft dasselbe ab, wie beim kumulativen Parser ohne assoziativen Zugriff.

Erkennbar sind folgende Unterschiede:

- Das semantische Netz erlaubt keine rekursiven Teil-von Relationen und ist damit jeder wirklich syntaktischen Struktur an Ausdruckskraft unterlegen. Das spielt allerdings in der Praxis keine Rolle, da man wirklich rekursive Strukturen in der Präsenz größerer Datenmengen sowieso vermeiden muß.

- Man konzentriert sich auf Anwendungen, bei denen es viele Konzepte in komplizierter Verschachtelung, aber pro Primitivkonzept nur wenige konkurrierende Instanzen gibt (zwanzig gelten als viele!).

- Da dadurch immer noch exponentiell wachsende Suchbäume drohen und man relativ tiefe Teil-von Hierarchien zulassen möchte, liegt großes Gewicht auf der Bewertung und einer Kontrolle, die die Bewertung auf optimale Weise nutzt, um früh das Ziel zu instanziieren und nicht alle Knoten des Suchbaums expandieren zu müssen. Wenn eine Bewertung für bereits in-

stanziierte Konzepte angegeben werden kann, die das Separierbarkeitskriterium der dynamischen Optimierung (siehe [BELL]) erfüllt, so können solche Abbruchkriterien bestimmt werden. Wenn darüberhinaus eine nichttriviale, optimistische Schätzung für die Kosten des noch fehlenden Weges zur Instanziierung des Ziels angegeben werden kann, so kann der Suchbaum noch weiter beschnitten werden durch Anwendung der A^*-Kontrolle (siehe [NILS]).

- Die semantischen Netze scheinen eher geeignet im Rahmen von Expertensystemen, bei denen der Benutzer interaktiv ein Zielkonzept bestimmt. Sie bieten wohl auch Vorteile durch größere Flexibilität beim Hinzufügen neuer Konzepte usw. Die KG kann dagegen bei der Verwendung allzuvieler Nichtterminale unübersichtlich werden. Die Modifikation einer KG ist im allgemeinen mit höherem Aufwand verbunden. Dafür bietet sie Vorteile bei der Effizienz und Parallelisierbarkeit. Insbesondere wenn es viele tausend mögliche konkurrierende Instanzen eines Typs gibt, ist es wohl übersichtlicher, sie einfach als eine Menge von Zwischenergebnissen zu betrachten, statt sich einen entsprechenden Baum vorzustellen, an dem jeder Knoten ein teilweise instanziiertes, semantisches Teilnetz vorstellt.

- Die semantischen Netze haben durch die Generalisierungskanten mit ihren Vererbungsmechanismen neben den Teil-von Relationen Ausdrucksmöglichkeiten, die man so übersichtlich in einer KG nicht zur Verfügung hat.

Trotz dieser Unterschiede kann man durchaus behaupten, daß die Darstellung als semantisches Netz viele Gemeinsamkeiten mit der Betrachtung desselben Mustererkennungsproblems als KG haben kann.

7.5 Mathematische Fundierung der Mustererkennung in der Literatur

Die Literatur ist gerade in dieser Richtung sehr umfangreich geworden. Es werden weit voneinander entfernte Fächer der Mathematik zur Formulierung des 'Mustererkennungsproblems' herangezogen. Ein solcher Abschnitt kann also unmöglich auf Vollständigkeit abzielen. Stellvertretend wird hier eine kleine Auswahl diskutiert.

Probleme, wie z. B. zu entscheiden, ob ein Fahrzeug bestimmten Typs in einer gestörten Bildfolge präsent ist oder nicht, können zunächst mit stochastischen Methoden angegangen werden. Etwa mit der Entscheidungsregel nach Bayes. Zur Anwendung dieser Methoden auf die Bildverarbeitung kann man

z. B. [WINK] zu Rate ziehen. Es gibt dabei, wie der Autor einräumt, zwei Hauptschwierigkeiten. Zum einen wird der Berechnungsaufwand enorm, wenn die a posteriori Schätzungen nicht mehr analytisch integriert werden können. Zum anderen ist die Konstruktion einer angemessenen, nicht trivialen a priori Wahrscheinlichkeit auf den Eingangsdaten aus der Aufgabenstellung heraus ein schwieriges Problem. Man ist hier aber in letzter Zeit ein erhebliches Stück vorangekommen. In [HORN] werden z. B. Verfahren vorgestellt, die ein 'Lernen' von solchen Dichten aus Beispieldaten erlauben, auch wenn die Objekte in der 3D Szene beliebig gedreht werden und das durch die Projektion entstehende 'Nichtwissen' mit modelliert werden muß.

In [BENN] wird in Gestalt der 'observer mechanics' ein maß- und wahrscheinlichkeitstheoretischer Ansatz präsentiert, der sogar Parallelen zur Quantenmechanik aufzeigt. Ziel ist die Modellierung der Beziehung zwischen einem Beobachter und seiner Welt. Das vorgeschlagene Modell ist ein Sechstupel aus einem 'Konfigurationsraum' X, einem 'Prämissenraum' Y, einer Menge zu unterscheidender Konfigurationen E, einer Menge zu unterscheidender Prämissen S, einer surjektiven Funktion $\pi : X \to Y$, die 'Perspektive' genannt wird, und einem 'Interpretations Kern' η. Zu X und Y sind jeweils Mengenalgebren definiert, so daß man Maße auf den darin enthaltenen Teilmengen bestimmen kann. Natürlich ist π eine meßbare Funktion. Für eine Prämisse $s \in S$ ist $\eta(s, \cdot)$ dann ein Entscheidungsmaß. Als Beispiel wird ein 'Frosch' angegeben, der 'eßbare Fliegen' fangen muß. Die eßbaren Fliegen unterscheiden sich von den giftigen durch ihre kreisförmige Flugbahn. Für X wird die Menge aller polynomial beschreibbaren geschlossenen Kurven im 3D Raum gesetzt. Das Problem liegt in der orthogonalen Projektion π in die 2D Retina Y: Ist der Schluß η vom elliptischen Bild einer Flugbahn auf eine eßbare Fliege eine geeignete Überlebensstrategie? Kreisbahnen werden durch π zu Ellipsen, aber die Menge aller möglichen Urbilder einer Ellipse enthält sehr viel mehr Kurven, die keine Kreise sind, als Kreise. Das Maß der Kreise darin ist Null. Danach müßten fast alle Fliegen, die gefangen werden, giftig sein. Wenn aber das Maß auf X anders gewählt wird, etwa so, daß ein fester Prozentsatz (nicht Null) der Fliegen eßbar ist, so kehren sich die Verhältnisse um. Derselbe Schluß vom elliptischen Bild der Bahn auf eßbare Fliege ist jetzt fast immer richtig. Dieses etwas künstlich anmutende Beispiel macht die Hauptargumente für diesen Ansatz dennoch sehr deutlich: Durch die Projektion gehen die Informationen, die man für einen korrekten Schluß bräuchte, verloren. Erst dadurch, daß man nicht triviale a priori Verteilungsmaße auf den zu unterscheidenden Fällen einführt, werden die Probleme wieder lösbar. Solche elementaren 'observer' dienen dann unter anderem als Bausteine für Hierarchien oder auch als Elemente von dynamischen Systemen, die miteinander in Raum und Zeit interagieren. Es wird eine mathematische Semantik für sie definiert, und es wird die Beziehung zur Berechenbarkeitslehre geklärt (mit dem Ergebnis, daß der 'observer' eine Turingmaschine simulieren kann, die 'observer'-Theorie

also die der Berechenbarkeit umfaßt und erweitert.)

In [GRIM] zeigt sich das 'Mustererkennungs Problem' als ein kombinatorisches Problem: Gegeben ist ein Modell als Menge von 'features' (also etwa geraden Linienstücken), eine Menge von gemessenen Daten 'features' und ein Transformationsraum ('pose space'), der Funktionen vom Modellraum in den Datenraum enthält, auf dem eine Metrik definiert ist. Ein Funktional - z. B. die Fehlerquadratsumme - ist zu minimieren. Das kann auf analytischem oder numerischem Weg geschehen. Das Problem wird in der korrekten Zuordnung der Daten 'features' zu den Modell 'features' gesehen. Wenn m die Kardinalität des Modells ist und n die Anzahl der Daten 'features', so hat der zu durchsuchende Funktionsraum die Größe m^n. Das Problem liegt also in der Vermeidung der an sich vorliegenden exponentiellen Rechenkomplexität. Hierzu wird vorgeschlagen, sich gewisser unärer und binärer Prädikate ('constraints') zu bedienen. Es sind dies oft dieselben oder ähnliche Prädikate, wie sie auch in den Produktionen dieser Arbeit verwendet werden. Der Funktionsraum wird sukzessive baumförmig durchsucht, wobei ausgehend von der zunächst leeren Zuordnung jeweils eine weitere Zuordnung der noch nicht zugeordneten Daten 'features' nur dann vorgenommen wird, wenn alle 'constraints' mit den schon gemachten Zuordnungen erfüllt sind. Bricht diese Suche ab, ohne zu einer Zuordnung sämtlicher Daten 'features' gekommen zu sein, so wird entschieden, das gesuchte Objekt sei nicht präsent. Andernfalls kann durch Minimierung des Fehlerfunktionals für jede Zuordnung die zugehörige Transformation bestimmt werden. Indem dem Modell ein 'Null feature' hinzugefügt wird, kann toleriert werden, daß die Daten gestört sind.

In [PAVL] findet sich ein eher algebraisch-topologisch motivierter Zugang zum 'Mustererkennungs Problem'. Die Lektüre dieses Buches setzt gewisse Kenntnisse aus der mathematischen Kategorienlehre voraus, die von einem Praktiker der Mustererkennung wohl nicht mehr verlangt werden sollten. Das erscheint nicht zwingend. Wenn ein Bild eine Funktion von einer Punktmenge nach einer Menge von Grau- oder Farbwerten ist, dann wird für das Studium von Transformationen, Invarianzen und kanonischen Formen auf der Menge der Bilder die Mengenlehre ausreichen. Tatsächlich kommt auch bei Monique Pavel in der Formulierung der Aufgabenstellung nichts vor, was nicht eine Menge wäre. Sie schreibt (Seite 9):

- "The *Fundamental Problem of Image Classification* and in particular the *Fundamental Problem of Pattern Recognition* consists in deciding by automatic means whether given a set $\mathcal{I}$ of Images, a set T of transformations, and an equivalence Relation defined over the whole of $\mathcal{I}$, two images $I_1, I_2 \in \mathcal{I}$ are equivalent modulo T (*classification*), respectively whether an image $I \in \mathcal{I}$ is equivalent modulo T to a pattern $\mathcal{P}_0$ (if it exsits) belonging to a set $\mathcal{P}$ of patterns (recognition)."

Zu den syntaktischen Verfahren kommt sie im Kapitel 4. Der dort definierte

Begriff *configuration* beruht aber auf Konkatenation. Die Menge der Konfigurationen C ist eine Halbgruppe mit der leeren Konfiguration als neutralem Element. Zusammen mit der Gruppe der zugelassenen Transformationen, bzgl. denen Invarianz gelten soll, ergibt sich eine algebraisch zu betrachtende Struktur. Der so definierte Begriffsapparat umfaßt und präzisiert Strukturen der klassischen syntaktischen Mustererkennung, wie etwa die PDL. Er taugt aber nicht mehr zur formalen Beschreibung von NAPEs, Feld Grammatiken, Graph Grammatiken und Koordinaten Grammatiken. Insbesondere erscheint der Schritt von Sprachen, die 2D Bilder beschreiben, zu solchen, die 3D Szenen beschreiben, auf diesem Fundament nicht so naheliegend.

Die syntaktischen Ansätze zur Mustererkennung erscheinen vor diesem Spektrum von Maßtheorie über Schätztheorie, Kombinatorik und universelle Algebra in der gegenwärtigen Literatur etwas unterrepräsentiert. Diese Arbeit mag hier ihren kleinen Beitrag einfügen. Es ist wünschenswert, zu einer vereinheitlichten Theorie der Mustererkennung zu gelangen. Einerseits sollte daher jeder theoretische Ansatz an praktikablen Beispielanwendungen demonstriert werden, damit die Besonderheiten, Vorzüge und Nachteile leicht einsehbar sind. Und andererseits sollte gezielt nach 'Verzahnungen' gesucht werden, also nach Einbettungen der einen Struktur in die andere oder nach Parallelstrukturen in den Ansätzen. In den Abschnitten 2.3.3 und 7.4 finden sich einige Überlegungen dazu bzgl. KGs, semantischer Netze und anderer Strukturen. Auf die Notwendigkeit, kombinatorische Abschätzungen durchzuführen, wurde verschiedentlich hingewiesen (siehe Abschnitt 5.3 und Anhänge C und D). Es erscheint wünschenswert, in die Bedingungsprädikate und vor allen Dingen in die Attributberechnungsfunktionen Verfahren aus der Schätztheorie einzubeziehen. Das würde es ermöglichen, den Instanzen Wahrscheinlichkeiten als Attributwerte zuzuordnen. Damit wären die KGs einsetzbar für Aufgabenstellungen, die korrekt quantifizierte Detektionswahrscheinlichkeiten und Falschalarmraten erfordern. Bisher ist das nicht geschehen, weil unklar war, ob die entsprechenden Dichten aus der meist spärlichen Datenlage verläßlich quantifiziert werden können. Hier ergeben sich nun durch die neuere Literatur (z. B. [HORN]) vielversprechende Ansätze für weitere Arbeit.

7.6 Zur Geschichte der KGs

Als erster definiert R. H. Anderson in seiner Ph. D. Thesis [ANDS-68-2] an der Harvard University die Koordinaten Grammatik und den zugehörigen Reduktionsbegriff. In einem Sammelband desselben Jahres wird die Definition einem breiteren Publikum zugänglich ([ANDS-68-1]). Anwendung war das Parsen von Formeln in zweidimensionaler mathematischer Notation. Eine Vorverarbeitung extrahiert aus dem handgeschriebenen Inputbild die einzelnen mathematischen

Symbole und assoziiert die Koordinaten von Zentrum, oberem Rand, linkem Rand und so weiter. Der Parser läuft dann im '*top down*' Modus.

In K. S. Fus Sammelband [FU-77] wird das Verfahren noch einmal erläutert [ANDS-77]. Dabei räumt Anderson ein, daß es in der ersten Implementierung 1968 in LISP auf der damals am MIT verfügbaren Hardware Probleme mit der Rechenzeit und Effizienz des Verfahrens gab, die wesentlich durch die vielen Möglichkeiten des 'partitioning' verursacht waren. Dem entspricht die in der vorliegenden Arbeit in Abschnitt 5.2.3 erläuterte Problematik. Bei der Erkennung von mathematischer Notation kann man sich Heuristiken zu Nutze machen, die darauf beruhen, daß es eben doch eine Vorzugsrichtung von links nach rechts gibt. Das nutzt Anderson aus und 'scannt' die Notation in dieser Richtung. Auch die Reihenfolge der Produktionen ist nicht beliebig. Die Produktion, die *sin* als Sinusfunktion interpretiert, steht vor der, die dasselbe Bild als Produkt dreier Variablen $s \cdot i \cdot n$ auffaßt. Die erste gefundene Interpretation gilt als korrekt und bricht den Prozeß ab. Anderson weist auf eine mögliche Beschleunigung des Verfahrens durch Ausnutzung einer weiteren Eigenschaft der mathematischen Notation hin, wie sie durch Fu in [FU-74] im Anhang C vorgeschlagen wird. Mathematische Operatoren können einander nämlich *dominieren*. Und das drückt sich in entsprechenden geometrischen und topologischen Relationen aus, die man hinzunehmen kann, um den Parser effizienter zu machen.

Bei Anderson findet man keine exakte Definition der Koordinaten Grammatik. Die Arbeit bleibt eher anwendungsorientiert. Milgram und Rosenfeld liefern dann in den siebziger Jahren in [MILG] die ersten eher grundlagenorientierten Ansätze. In dieser Zeit scheint aber das Interesse an syntaktischen Methoden für das Computersehen nachzulassen. Die Euphorie, wie man sie noch 1969 in den hitzigen Diskussionen in [NARA-69] z. B. bei R. Narasimhan antrifft, weicht einer gewissen Ernüchterung. Offenbar hatte man keine relevanten Anwendungen vorzuweisen. Die KI wandte sich den Schachprogrammen und Expertensystemen zu.

In der Vielzahl der Veröffentlichungen zur Mustererkennung machen die syntaktischen Verfahren nur noch einen kleinen Bruchteil aus. Bei A. Rosenfeld finden sich aber gelegentlich noch Arbeiten zur Koordinaten Grammatik - z. B. Ende der achtziger Jahre in [ROSE-89-1, ROSE-89-2]. A. Nakamura entwickelt Rosenfelds Ideen zur lokalen KG aus [ROSE-89-1] weiter in [NAKA-89, NAKA-95] (siehe Abschnitt 2.3.1). Die dort verfolgten Ziele und Ansätze unterscheiden sich erheblich von denen in dieser Arbeit. Die bisher aus der BPI Arbeitsgruppe am FIM erschienen Veröffentlichungen z. B. [LUE-86-2, FUE-87, FUE-90-2, STIL-95-1, STIL-95-2, STIL-96] betonen die praktische Seite und verwenden als theoretische Grundlage Blackboardansätze, Produktionssysteme und Produktionsnetze. [STIL-97] beinhaltet Überlegungen zur Semantik von Produktionsnetzen, die übertragbar sind auf Koordinaten Grammatiken. In [MICH-96] finden sich einige Teilaspekte aus der vorliegenden Arbeit.

Chapter 8

Zusammenfassung

Die vorliegende Arbeit behandelt syntaktische Verfahren der Mustererkennung. Die Theorie der Koordinaten Grammatiken (KGs) wird präzisiert und erweitert. Es wird demonstriert, daß Anwendungen im Bereich Computersehen möglich sind und daß die Theorie der KGs hier hilfreich und nützlich sein kann. Als Beispiel wird eine KG zur Detektion und Lokalisation eines modellierten 3D Objektes in hochaufgelösten Bildfolgen beschrieben. Die Aufnahmestandorte sind bodenbasiert und simulieren einen Vorbeiflug in sehr niedriger Höhe. Wesentliche Teile des Verfahrens, wie z. B. der abschließende Modellvergleich, verwenden ein ortsfestes 3D Szenenkoordinatensystem. Die Arbeit plädiert für die Nutzung solcher syntaktischer 3D Verfahren in der Objekterkennung insbesondere dann, wenn sich schwierige Zuordnungsprobleme zwischen Modell und Daten ergeben. Die wesentlichen Resultate und ihre Implikationen für die Praxis können zu folgenden Hauptpunkten zusammengefaßt werden:

- **KGs sind eine theoretische Grundlage für die Bildverarbeitung und Szenenanalyse:** Die Theorie der KGs, wie sie von Anderson, Milgram und Rosenfeld begonnen wurde, wird präzisiert und erweitert von der syntaktischen 2D Bildanalyse hin zur syntaktischen Szenenanalyse in Attributbereichen mit unterschiedlichen Topologien und Dimensionen. Insbesondere werden modellgestützte Verfahren mit kartesischen 3D Koordinaten im Attributbereich bearbeitet. Es ist aber andererseits auch möglich, Bildverarbeitungsprobleme wie die Linienverlängerung damit anzugehen. Die KGs sind ein besonders übersichtlicher und modularer Rahmen zur Arbeit an automatischen Erkennungsverfahren.

- **Es wird eine Hierarchie der KGs bzgl. der Rechenkomplexität konstruiert:** Eine Einbettung von Stringgrammatiken nach Chomsky in die KGs wird durchgeführt. Für eingeschränkte, lokale KGs ist auch die Umkehrung möglich, wie durch Rosenfeld gezeigt wurde. Mit den attribu-

tierten Grammatiken nach Knuth in der Anwendung auf die Bildanalyse nach Fu besteht eine gewisse Verwandtschaft in den Zielen und Begriffen. Aber die KGs arbeiten auf Mengen von attributierten Symbolen und nicht auf Worten, Bäumen oder Graphen. Das impliziert erhebliche Unterschiede in der Theorie. Die folgenden Resultate im Hinblick auf die Rechenkomplexität basieren daher nicht nur auf solchen Einbettungen oder Simulationen, sondern vor allen Dingen auf der Analyse der *Produktionsnetze*, die das Zusammenspiel der Produktionen einer KG beschreiben, und der *Parser*, die zu den KGs gehören:

- Wenn eine *zyklenfreie* KG und eine Menge terminaler Instanzen gegeben ist, so ist die Frage, ob diese Menge in der Sprache zur KG liegt, entscheidbar mit polynomialer Komplexität in der Kardinalität der Menge. Die höchste Potenz kann aus der Gestalt des Produktionsnetzes bestimmt werden, und ist i. A. recht groß (exponentiell mit der Länge des längsten Weges im Produktionsnetz).
- Das analoge Wortproblem für *monotone* KGs ist ebenfalls entscheidbar, jedoch von hoher Komplexität. Es gibt NP-harte monotone KGs.
- *Uneingeschränkte* KGs können Turingmaschinen simulieren. Es gibt daher ein entsprechendes Halteproblem. Sie sind nicht entscheidbar.

In der Praxis sollte man daher monotone KGs verwenden und nach Möglichkeit Zyklen im Produktionsnetz vermeiden. Da auch dann der Grad des Polynoms hoch ist, sollte man Abschätzungen zum voraussichtlichen Aufwand in Abhängigkeit von der Dichte der Daten im Attributbereich anstellen.

- **Kumulative Parser sind eine Näherungslösung und sparen Aufwand:** Der vorgestellte Top down Parser dient nur dazu, die Kardinalität der Sprachen abzuschätzen. Der durch die Definition der KGs nahegelegte rekursive, bottom up Parser, der Mengen als Variablen übergibt und einen Suchbaum mit Backtracking aufspannt, ist sehr aufwendig. Daher wird eine akkumulierende Methode vorgeschlagen, die ohne Rekursion und Backtracking auskommt und keine Mengen als Parameter übergibt. Das ist aus zwei Gründen nur eine Näherungslösung:
  - Es kann Kumulationen geben, denen keine Reduktion entspricht.
  - Es kann passieren, daß die akkumulierende Methode in eine Endlosschleife gerät, wo eigentlich Entscheidbarkeit vorliegt.

  Für praktisch relevante KGs ist die Aufwandsreduktion aber so groß, daß diese Punkte hinzunehmen sind.

- **KI Techniken wie das Blackboard können zur Implementierung verwendet werden:** Dabei werden die Produktionen zu Wissensquellen. Das Produktionsnetz mit seinen Verbindungen zwischen den Symbolen und Produktionen wird in einer speziellen Wissensquelle - dem sogenannten Dispatcher - abgelegt. Das erhöht die Übersichlichkeit und Modularität. Die Menge der aktuell kumulierten Instanzen wird in einer assoziativ ansprechbaren Datenbasis gehalten. So kann der Suchaufwand bei der Bildung der Ausgangskonfigurationen erheblich verringert werden. Es können Bewertungs- und Vertrauensmaße definiert werden, die im Kontrollmodul zur Beschleunigung des Ablaufs eingesetzt werden. Zu einem solchen System gehören dann auch graphische Ausgabeschnittstellen sowie Werkzeuge zur Fehlersuche und zur Erklärung von Resultaten. Es werden darüberhinaus Untersuchungen zur Gestalt geeigneter Hardware angestellt. Lösungen mit massiver Parallelität sind im Experimentierstadium.

- **Die Praxis der KGs im Computersehen legt die Definition gewisser zusätzlicher Hilfsmittel nahe:** Es werden die folgenden Strukturen hierzu eingeführt:

  - Ein Dimensionsbegriff für den Attributbereich (um Bildanalyse von Szenenanalyse usw. zu unterscheiden).
  - Nachbarschaftsgraphen mit Teil-von Hierarchien (in loser Anlehnung an semantische Netze).
  - Die Darstellung bestimmter Clusterverfahren durch spezielle Produktionen (mit Mengen als Ausgangskonfiguration).
  - Ein geometrischer Modellbegriff (mit zugehörigem Vergleichsverfahren als Produktion).
  - Eine algebraische Behandlung von Symmetrien im Attributbereich (durch auf den Attributindizes operierende Gruppen).

- **Ein Beispielverfahren zur Demonstration der vorstehenden Punkte wird vorgestellt:** Darin wird eine KG zur 3D Fahrzeugdetektion und Lokalisation in Bildfolgen hoher Auflösung aus dem sichtbaren Spektralbereich verwendet. Mit den bodenbasierten Aufnahmen wird eine tieffliegende Plattform simuliert, die extreme Schrägsichten liefert. Das Verfahren gliedert sich in folgende Schritte:

  - Merkmalsextraktion durch Mehrfachschwellenbinarisierung und Polygonzugapproximation.
  - Einfache 2D KG zur Konturverlängerung und zum Auffinden von Hinweisen für mögliche Objektteilflächen.
  - Stereoberechnung zur Erzeugung von Hinweisen für 3D Objektteilflächen.

– Eine 3D KG inklusive abschließendem Modellvergleich für die eigentliche Detektion und Lokalisation.

Das Verfahren wurde an einer ersten Bildfolge entwickelt und getestet. Weitere Tests wurden mit Zufallsdaten durchgeführt, die ein sehr robustes Verhalten zeigten. Daraufhin wurden Tests mit schwierigen Bildfolgen, die bei der Verfahrensentwicklung noch nicht vorlagen, durchgeführt. Dokumentiert ist ein Bildfolge, mit der noch korrekte Ergebnisse erzielt werden, und eine, bei der das Fahrzeug nicht mehr gefunden wird. Das gibt eine ungefähre Vorstellung von den Grenzen der Leistungsfähigkeit.

Die Anzahl der erforderlichen Rechenoperationen und der Speicherbedarf sind hoch. Das resultiert wohl auch aus der Problemstellung selbst. Allerdings sind, was die Kosten von Rechenleistung und Speicher angeht, in der Zukunft rasche Fortschritte zu erwarten, so daß Verfahren wie das vorgestelle in den Bereich des Machbaren kommen. Aufbau, Test und Wartung eines praktischen Detektionssystems mit KGs ist umfangreiche Ingenieursarbeit. Lernverfahren, die automatisch solche Systeme mit Hilfe von Beispieldaten erzeugen, sind derzeit nicht vorgesehen. Denkbar sind teilautomatische Adaptivität gewisser Verfahrensparameter an aktuelle Daten und ein weiterer Ausbau der beschriebenen Hilfswerkzeuge. Ein wesentlicher Arbeitspunkt für die Zukunft ist in dem Einbau von Funktionen aus der Schätz- und Entscheidungstheorie erkennbar mit dem Ziel verbesserter Quantifizierbarkeit von Erkennungsraten. Auch muß noch an einer präziseren Fassung der Bewertungs- und Vertrauensmaße gearbeitet werden, so daß der Gewinn an Verarbeitungszeit quantifiziert und gegen den Verlust an semantischer Korrektheit bei vorzeitigem Abbruch der Suche abgewogen werden kann.

Appendix A

Tabelle der mathematischen Symbole

Lateinische Großbuchstaben:

Symbol	*Bedeutung*	*Seite*
D	Attributbereich	11
D_i	Komponente des Attributbereichs	11
E	Kantenmenge eines Graphen	72
F	Fehlerfunktional	83
G	Grammatik	11
I	Instanz	12
K	Knotenmenge eines Graphen	72
$Kand_j$	Kandidatenmenge	84
KG	Koordinaten Grammatik	11
Ko	Korrespondenzen zum Modell	82
M	Modell	82
N	Menge der Nichtterminale	11
$\mathbb{N}$	Menge der natürlichen Zahlen mit Null	
P	Menge der Produktionen	11
Pr	Menge der abgearbeiteten Ausgangskonfigurationen	56
Pa_{VE}	Partnermenge eines Verarbeitungselementes	60
Q	Menge der abzuarbeitenden Ausgangskonfigurationen	56
S	Menge von Instanzen	18
T	Menge der Terminale	11
Tr	Menge der zugelassenen Transformationen	82
V	Alphabet einer Sprache	11
VE	Verarbeitungselement einer KG	60
Z	Menge der ganzen Zahlen	

Kaligraphische Großbuchstaben:

Symbol	*Bedeutung*	*Seite*
$\mathcal{A}_p$	Menge der Ausgangskonfigurationen von p	13
$\mathcal{B}(\lambda)$	Bild eines Tupels λ	18
$\mathcal{K}$	kumulative Sprache einer KG	21
$\mathcal{L}_{noisy}$	gestörte Sprache einer KG	20
$\mathcal{L}_{pure}$	Sprache einer KG	19
$\mathcal{N}$	Nachbarschaftsgraph	78
$\mathcal{O}$	Ordnung eines Alphabets einer KG	74
$\mathcal{RG}$	Reduktionsgraph	72
$\mathcal{S}$	Fehlerquadratsumme im Modell Vergleich	163
$\mathcal{T}$	Menge der Terminale im Universum einer KG	19
$\mathcal{U}$	Universum einer KG	12
$\mathcal{V}$	Menge aller Verarbeitungselemente einer KG	60
$\mathcal{Z}_p$	Menge der Zielkonfigurationen von p	13

Griechische Buchstaben:

Symbol	*Bedeutung*	*Seite*
Γ	Wort aus V^*	12
Λ	linke Seite einer Produktion	11
Σ	rechte Seite einer Produktion	11
κ	Konfiguration	12
κ_a	Ausgangskonfiguration	13
κ_z	Zielkonfiguration	13
ϕ	Attributfunktion einer Produktion	11
π	Bedingungsprädikat einer Produktion	11
ψ	globales Konsitenzprädikat	21

Symbole der verwendeten KGs:

Symbol	*Bedeutung*	*Seite*
angle	zwei antiparallele, benachbarte *prol_line*s	111
con	Konjunktion (Nichtterminal der Erfüllbarkeits KG)	42
E_structure	zwei gleich ausgerichtete, benachbarte *U_structure*s	101
erf	Startsymbol der Erfüllbarkeits KG	42
dis	Disjunktion (Nichtterminal der Erfüllbarkeits KG)	42
go	Startsymbol der 'worst case' KG	52
light_truck	VW-Bus mit Pritsche und Doppelkabine	100
Linie	kurzes, geradliniges Konturstück	13
lit	Literal (Terminal der Erfüllbarkeits KG)	42
L_Linie	verlängertes, geradliniges Konturstück	22
nt	Nichtterminal der 'worst case' KG	52
O_Viereck	Viereck, dem eine Seite fehlt	22
open_quad	Viereck, dem eine Seite fehlt	108
prol_line	verlängertes, geradliniges Konturstück	111
te	Terminal der 'worst case' KG	52
U_structure	3D Viereck, dem eine Seite fehlt	101
Viereck	Viereck	77
Winkel	zwei antiparallele, benachbarte *Linie*n	13
Zweiflaechner	zwei Vierecke	77
$\wedge$	Terminal der Erfüllbarkeits KG	42
$\vee$	Terminal der Erfüllbarkeits KG	42
$*$	Terminal der Erfüllbarkeits KG	42

Appendix B

Definition der Prädikate und Funktionen

Wichtigste Prädikate in den Produktionen sind die **Nachbarschaften**. Normalerweise ist diese topologische Relation durch eine Metrik induziert. Sind z. B. $d_1 \in D_i$ und $d_2 \in D_j$ Werte aus intervallförmigen Attributbereichen gleicher Dimension d und $q \geq 1$ eine reele Konstante, so ist mit

$$dis(d_1, d_2) = \left[\sum_{k=1}^{d} |d_{1,k} - d_{2,k}|^q \right]^{\frac{1}{q}}$$

eine Metrik gegeben. Setzt man dann einen Schwellwert $t > 0$, so ergibt sich das Nachbarschaftsprädikat[1]

$$adj_{q,t}(d_1, d_2) \stackrel{def}{\Longleftrightarrow} dis(d_1, d_2) \leq t.$$

Sind D_i und D_j Attributbereiche gleicher Dimension d und gleicher torischer oder elliptischer Geometrie, so muß dies bei der Definition der Metrik berücksichtigt werden, indem man $dis(d_1, d_2)$ ersetzt durch die Länge der kürzesten Verbindung. Im Falle eindimensionaler Orientierungen können z. B. 359^o und 1^o als benachbart gelten. Hier bedeutet benachbart ungefähr parallel.

Die wichtigsten Funktionen, die in den Produktionen verwendet werden, sind durch **lineare Gleichungssyteme** definiert. I. a. ist die Lösung dieser Systeme nicht für jedes Argument definiert, weil der Rang entarten kann, was dann auf eine Division durch Null führt. Das muß abgefangen werden, indem die

[1] Übrigens ist an dieser Stelle eine 'Fuzzifizierung' nach [ZADE] ohne weiteres möglich, indem man den Schwellwert als Schwellwertfunktion nach dem Zugehörigkeitsintervall $[0, 1] \subset R$ auffaßt, und dann durch eine stetige, z. B. stückweise lineare 'Fuzzirampe' ersetzt. Die logischen Fuzzy-Operatoren etwa für $\wedge$ und $\vee$ werden dann in die Funktion ϕ eingebaut, und liefern ein entsprechend quantifiziertes Zugehörigkeitsattribut.

Funktion einen entsprechenden Seiteneffekt liefert, der in das Bedingungsprädikat aufgenommen wird. Darüberhinaus liefern die Funktionen i. a. reelle Werte, auch wenn ihre Argumente ganzzahlig sind. Diese Werte müssen dann gerundet werden, damit man sie als Koordinaten für neue Instanzen eintragen kann. Wenn man numerische Stabilität insbesondere dadurch herstellt, daß man kleine Nenner vermeidet (durch entsprechende Prädikate im Bedingungsteil der Produktionen), so kann auf Gleitkommadarstellungen verzichtet werden. Man kann dann die ganzzahlige Division verwenden und die Operationen auf entsprechend einfachen Prozessoren in massiv paralleler Weise durchführen. Winkelfunktionen und dergleichen können in Tabellen abgelegt werden. Die in den Beispielproduktionen dieser Arbeit verwendeten Prädikate und Funktionen werden im folgenden in alphabetischer Reihenfolge aufgeführt und definiert.

Tabelle der Prädikate und Funktionen

- *adj* steht für die Nachbarschaft zweier Punkte im Bild oder in der Szene. Wenn nichts weiter dabei steht, ist die euklidische Metrik zugrunde gelegt, also $q = 2$. Der Schwellwert $t = d_{max}$ ist Verfahrensparameter (siehe Anhang C). Dieses Prädikat ist kommutativ und reflexiv.

- *apa* steht für die Antiparallelität zweier Orientierungen im Bild, also $apa = \neg par$. Der Schwellwert $t = d_{min}$ dabei ist Verfahrensparameter (siehe Anhang C). Dieses Prädikat ist kommutativ und irreflexiv.

- *calculate_regression* minimiert die Summe der quadrierten Abstände einer gegebenen Punktmenge in der euklidischen 2D Ebene von einer Geraden in Normalform $n_1 x + n_2 y - \alpha = 0$ unter Variation der Parameter n und α. Wie leicht zu verifizieren ist, führt dies auf ein homogenes Gleichungssystem der Gestalt

$$\begin{pmatrix} -\sum_x & -\sum_y & 1 \\ \sum_{xx} & \sum_{xy} & -\sum_x \\ \sum_{yx} & \sum_{yy} & -\sum_y \end{pmatrix} \begin{pmatrix} n_1 \\ n_2 \\ \alpha \end{pmatrix} = 0,$$

  wobei $\sum_x$ für die Summe der X-Koordinaten, $\sum_{xy}$ für die Summe der Produkte aus X- und Y-Koordinaten usw. steht. Der Rang der Matrix ist höchstens zwei. Die Lösung ist nur bis auf einen Faktor zu bestimmen. Man kann z. B. n normieren. Es kann gelegentlich vorkommen, daß der Rang kleiner als zwei wird. In diesen Fällen ist eine sinnvolle Regressionsgerade nicht definiert. Die Funktion liefert einen entsprechenden Seiteneffekt. Die Punktmenge wird gegeben als Menge der Ortsattributwerte von Instanzen des Symbols *Linie*. Da einzelne Punkte Attributwert mehrerer Instanzen sein können, müssen sie entsprechend gewichtet in die Summierungen eingehen. *calculate_regression* ist invariant bzgl. beliebiger Aufzählungsreihenfolgen der Instanzenmenge.

- *col* überprüft für vier Punkte in der Ebene, ob sie annähernd auf einer Geraden liegen.

$$col(P_1, P_2, P_3, P_4) \stackrel{def}{\Longleftrightarrow} \exists a, b, c \in R \forall i \in \{1, ..., 4\} : |aP_{i,x} + bP_{i,x} - c| \leq t$$

  $t = d_{max}$ ist Verfahrensparameter.

- *col_set* überprüft, nach Bestimmung der Ausgleichsgeraden mit der Funktion *calculate_regression*, ob die Ortsattribute einer Menge von Linieninstanzen alle näher als ein Schwellwert an der Geraden liegen:

$$col\_set(S) \stackrel{def}{\Longleftrightarrow} \forall I \in S, i = 1, 2 : \ |n \cdot a(I)_{P_i} - \alpha| < t$$

  Der Schwellwert (bei normiertem n gilt $t = d_{max}$) ist Verfahrensparameter (siehe Anhang C). Um zu verhindern, daß sehr kurze, antiparallele Linieninstanzen in einer Menge sind, die das Prädikat erfüllt, wird zusätzlich paarweise Parallelität der Instanzen verlangt. Der Schwellwert hierzu ist ebenfalls ein Verfahrensparameter (siehe Anhang C). Das Prädikat *col_set* ist unabhängig von der Aufzählungsreihenfolge der Linieninstanzenmenge.

- *geom_const* beinhaltet geometrische constraints, die ein Polygonzug aus vier Punkten im Raum erfüllen muß, damit er als Umrandung einer Teilfläche einer Instanz des gesuchten Startsymbols in Frage kommt. Die Funktion int_{3d} liefert vier Punkte P_i. Diese müssen folgende Prädikate erfüllen:

  - Flächigkeit: Dazu wird die Summe der quadrierten Abstände der vier Punkte von einer Ebene $n_1x + n_2y + n_3z - \alpha = 0$ minimiert. Wie leicht zu verifizieren ist, führt dies auf ein homogenes Gleichungssystem der Gestalt

$$\begin{pmatrix} -\sum_x & -\sum_y & -\sum_z & 4 \\ \sum_{xx} & \sum_{xy} & \sum_{xz} & -4\sum_x \\ \sum_{xy} & \sum_{yy} & \sum_{yz} & -4\sum_y \\ \sum_{xz} & \sum_{yz} & \sum_{zz} & -4\sum_z \end{pmatrix} \begin{pmatrix} n_1 \\ n_2 \\ n_3 \\ \alpha \end{pmatrix} = 0,$$

    wobei $\sum_x$ für die Summe der X-Koordinaten, $\sum_{xy}$ für die Summe der Produkte aus X-und Y-Koordinaten usw. steht. Der Rang der Matrix ist höchstens drei. Die Lösung ist nur bis auf einen Faktor zu bestimmen. Man kann z. B. n normieren. Es kann gelegentlich vorkommen, daß der Rang kleiner als drei wird. In diesen Fällen ist eine sinnvolle Regressionsebene nicht definiert und damit auch kein Normalvektor. *geom_const* ist dann nicht erfüllt. Man kann auch hier einen Schwellwert $|\alpha - \sum_i P_i \cdot n| < d_{max}$ für den Fehler ansetzen. Darauf wurde in der im Abschnitt 6.2 dokumentierten Produktion verzichtet. Ist das

Vertrauen in die gemessenen Koordinaten der Punkte P_i eher gering, so kann das Prädikat *geom_const* als Seiteneffekt die Punkte auf die Regressionsebene projizieren. Diese Variante wurde anfangs implementiert und später wieder fallengelassen. Es ergeben sich dadurch keine großen Unterschiede.

- Mindest- und Maximallänge: $d_{min} \leq dis(P_{i-1}, P_i) \leq d_{max}$ für $i = 2, 3, 4$.
  d_{min} und d_{max} sind Verfahrensparameter (siehe Anhang C).
- Maximale Abweichung vom rechten Winkel: $|(P_{i-1} - P_i) \cdot (P_i - P_{i+1})| \leq t\ dis(P_{i-1}, P_i) dis(P_i, P_{i+1})$ für $i = 2, 3$. $t = cos(\alpha_{max})$ ist ein Verfahrensparameter (siehe Anhang C).

Für dis gilt hier die euklidische Distanz also $q = 2$. Es ist möglich, noch viele weitere geometrische Bedingungen hinzuzufügen. Im Extrem kann gegen eine Liste von Modellvierecken ein Fehlerquadratsummen minimierender Vergleich durchgeführt werden.

- int_{2d} berechnet den Schnittpunkt zweier Geraden in der Ebene, die durch die vier Punkte $P_1, ..., P_4$ gegeben sind, durch Lösen des entsprechenden Gleichungssytems

$$\begin{pmatrix} P_{1,y} - P_{2,y} & P_{1,x} - P_{2,x} \\ P_{3,y} - P_{4,y} & P_{3,x} - P_{4,x} \end{pmatrix} int_{2d} = \begin{pmatrix} P_{1,x}P_{2,y} - P_{1,y}P_{2,x} \\ P_{3,x}P_{4,y} - P_{3,y}P_{4,x} \end{pmatrix}.$$

  Es muß zur Wohldefiniertheit dieser Funktion die Determinante ungleich Null sein (damit der Schnittpunkt überhaupt existiert). Zur Wohldefiniertheit einer Produktion muß er innerhalb des entsprechenden Attributbereichs (der Bildgrenzen) liegen. Die Funktion liefert als Seiteneffekt einen entsprechenden Wahrheitswert, der in das Bedingungsprädikat der Produktion konjunktiv einfließen muß. Die Funktion ist kommutativ bzgl. der durch die Vertauschungen $(1, 2)$, $(3, 4)$ und $(1, 3)(2, 4)$ erzeugten Untergruppe der Indexpermutationen.

- int_{3d} berechnet den 'Schnittpunkt' zweier Geraden im 3D Raum. Das ist der Mittelpunkt auf der kürzesten Verbindung zwischen den Geraden. Diese sind wie bei int_{2d} durch die vier Punkte $P_1, ..., P_4$ gegeben. Es gilt also:

$$int_{3d} \stackrel{def}{=} \frac{1}{2}[P_1 + \lambda_1(P_2 - P_1) \ + \ P_3 + \lambda_2(P_4 - P_3)]$$

  wobei sich die Parameter wie folgt ergeben:

$$\lambda_1 \stackrel{def}{=} \frac{1}{n}[(P_4 - P_3) \cdot (P_2 - P_1)(P_4 - P_3) \cdot (P_1 - P_3) - |P_4 - P_3|^2 (P_2 - P_1) \cdot (P_1 - P_3)]$$

$$\lambda_2 \stackrel{def}{=} \frac{1}{n}[|P_2 - P_1|^2 (P_4 - P_3) \cdot (P_1 - P_3) - (P_4 - P_3) \cdot (P_2 - P_1)(P_2 - P_1) \cdot (P_1 - P_3)]$$

Die Funktion ist kommutativ bzgl. der durch die Vertauschungen $(1,2)$, $(3,4)$ und $(1,3)(2,4)$ erzeugten Untergruppe der Indexpermutationen. Zur Wohldefiniertheit muß der Nenner n ungleich Null sein. Er ergibt sich aus

$$n \stackrel{def}{=} |P_4 - P_3|^2 |P_2 - P_1|^2 - [(P_4 - P_3) \cdot (P_2 - P_1)]^2.$$

Zur Wohldefiniertheit einer Stereo-Produktion, die diese Funktion verwendet, muß das Ergebnis in den entsprechenden Attributraum fallen. Die Funktion übergibt einen entsprechenden Wahrheitswert, der in das Bedingungsprädikat konjunktiv einfließen muß. Für die Stereoproduktion muß das Objekt *vor* beiden Kameras liegen, müssen also λ_1 und λ_2 positiv sein. Die kürzeste Verbindung zwischen den Strahlen darf nicht länger sein als ein Schwellwert (dazu siehe Anhang C). Zur Konstruktion einer *U_structure* Instanz wird diese Funktion für jeden Punkt einmal - insgesamt also viermal - aufgerufen.

- *ls_sqr_tr* bestimmt die Transformation des besten Modell-Konfigurations-Vergleichs im Sinne von Abschnitt 5.2.2. Daher kommt der Name 'least squared error sum transformation'. Die Werte entstehen als Seiteneffekt bei der Bestimmung des Prädikats *template_matching*, das weiter unten beschrieben ist. Außer der Translation und Rotation wird noch die Fehlerquadratsumme und die Anzahl der gefundenen Teile als Attribut gesetzt.

- *ovl* ist auf vier Punkten P_1, P_2, P_3 und P_4 in der Ebenen definiert und überprüft, ob eine 'Überlappung' vorliegt:

$$ovl(P_1, P_2, P_3, P_4) \stackrel{def}{\Longleftrightarrow} P_2 \cdot (P_4 - P_1) > P_3 \cdot (P_4 - P_1)$$

- *overlapping_components* konstruiert nach Aufruf von *calculate_regression* eine Menge von Instanzen des Symbols *L_Linie*, die alle auf der Ausgleichsgeraden liegen. Dazu wird übergegangen zu einer Normalform mit einem Vektor senkrecht auf dem Normalvektor der Ausgleichsgeraden. Ein mit Null initialisiertes Histogramm über den skalaren Teil dieser Form wird für jedes Element der Instanzenmenge in der Regression zwischen dem kleineren und dem größeren Wert inkrementiert. Nach Abschluß dieses Prozesses stehen mit den Bereichen des Histogramms, die nicht mehr auf Null liegen, die Bereiche entlang der Geraden zur Verfügung, die durch die Instanzenmenge in der Regression 'überlappt' werden. Durch Schnitt der Geraden mit der Form für Anfangs- und Endwert eines solchen Nicht-Null-Bereichs entstehen die Endpunktkoordinaten der *L_Linie* Instanzen. Durch Wahl der Histogrammschrittweite und ein Über- oder Unterschreiten der tatsächlichen Anfangs- und Endwerte, kann entweder eine gewisse 'Mindestüberlappung' verlangt werden oder ein 'Überspringen'

kleinerer Lücken erreicht werden. Das ist ein Verfahrensparameter, auf den der Anhang C noch eingeht. *overlapping_components* ist kommutativ bzgl. beliebiger Permutationen der Aufzählung der Instanzen in der Ausgangsmenge.

- *par* steht für die Parallelität zweier Orientierungen im Bild. Es wird ein Schwellwert d_{max} gesetzt. Ist der Abstand kleiner als diese Schwelle, so gilt Parallelität. Sind also O und Q Orientierungen, so gilt:

$$par(O,Q) \stackrel{def}{\Longleftrightarrow} |O-Q| < d_{max}$$

  Dieses Prädikat ist kommutativ und reflexiv. Die Orientierungen im Bild sind ein eindimensionaler Raum mit geschlossener Topologie. Normalerweise nimmt man $Z \; mod \; 180$, um eine Anschauung als normale Winkelgrade zu ermöglichen. Der Betrag ist in diesem Raum als Betrag der kleinsten Differenz (Minimierung über alle Repräsentanten der Klassen) aufzufassen. Dieses Prädikat kann für beliebige Stellenzahlen n definiert werden, indem es dann jeweils paarweise für alle eingetragenen Orientierungen gelten soll.

- *prl* bezeichnet Parallelität der Oberflächennormalen in der Szene. Will man hier Innen- und Außenseite nicht unterscheiden, so muß die projektive Ebene als Topologie verwendet werden. Meist wird man jedoch wie im Kapitel 6 unterscheiden wollen. Dann haben die Normalvektoren die Topologie einer Kugel. Sie werden als normierte 3D Vektoren gespeichert. Zwei solche Normalvektoren n_1 und n_2 gelten als parallel, wenn das Sinusquadrat des eingeschlossenen Winkels einen Schwellwert d_{max} unterschreitet:

$$prl(n_1,n_2) \stackrel{def}{\Longleftrightarrow} 1 - n_1 \cdot n_2 < d_{max}$$

  Dieses Prädikat ist kommutativ und reflexiv. Es kann für beliebige Stellenzahlen n definiert werden, indem es dann jeweils paarweise für alle eingetragenen Orientierungen gelten soll.

- *template_matching* implementiert die im Abschnitt 5.2.2 diskutierte Funktionalität zum Vergleich zwischen einem Modell und einer Ausgangskonfiguration. Resultat ist erstens eine Transformation $t = (v_x, v_y, v_z, \alpha, \beta, \gamma)$ (Translation im Raum und starre Rotation um die drei Achsen), zweitens eine Fehlerquadratsumme, und drittens ein Wahrheitswert (ob alle Abstände ein feste Schwelle d_{max} unterschreiten).

  Die Rotation erfolgt in der Reihenfolge α um die X-Achse (Rollen), dann β um die Y-Achse (Nicken) und schließlich γ um die Z-Achse (Gieren), so daß die Drehmatrix die folgende Gestalt gewinnt:

$$\begin{pmatrix} \cos\gamma\cos\beta & \cos\gamma\sin\beta\sin\alpha - \sin\gamma\cos\alpha & \cos\gamma\sin\beta\cos\alpha + \sin\gamma\sin\alpha \\ \sin\gamma\cos\beta & \sin\gamma\sin\beta\sin\alpha + \cos\gamma\cos\alpha & \sin\gamma\sin\beta\cos\alpha - \cos\gamma\sin\alpha \\ -\sin\beta & \cos\beta\sin\alpha & \cos\beta\cos\alpha \end{pmatrix}$$

Das Modell $M = (p_1, ..., p_m)$ und die Korrespondenz K sind als Verfahrensparameter gegeben (siehe Anhang C). Die Attributvektoren der Ausgangskonfiguration $a(\kappa_a) = (a_1, ..., a_k)$ sind Eingangsparameter des Prädikates *template_matching*. Jedem Modellpunkt p ordnet die Korrespondenz eine Menge von Paaren von Indizes $K(p) = \{..., (j, i), ...\}$ zu, wobei der erste Index $1 \leq j \leq k$ eine Position in der Konfiguration meint und der zweite Index $1 \leq i \leq n$ eine Komponente des Attributbereichs. Die zu minimierende Fehlerquadrat Summe läuft über alle Modellpunkte und Korrespondenzen also

$$\mathcal{S} = \sum_{p \in M} \sum_{(j,i) \in K(p)} (t(p) - a_{j,i})^2.$$

Dabei variiert die Transformation t über alle Translationen und Rotationen. Die entsprechenden partiellen Ableitungen $\frac{\partial \mathcal{S}}{\partial v_x}$, $\frac{\partial \mathcal{S}}{\partial v_y}$, $\frac{\partial \mathcal{S}}{\partial v_z}$, $\frac{\partial \mathcal{S}}{\partial \alpha}$, $\frac{\partial \mathcal{S}}{\partial \beta}$ und $\frac{\partial \mathcal{S}}{\partial \gamma}$ werden gleich Null gesetzt und bilden so ein nichtlineares, homogenes Gleichungssystem. Lösungen des Minimierungsproblems lösen auch dieses Gleichungssystem.

Für jede Lösung $t = (v_x, v_y, v_z, \alpha, \beta, \gamma)$ führt die Addition von Vielfachen von 2π zu einem der Winkel wieder zu einer Lösung. Darüberhinaus gibt es nichttriviale Identitäten. Z. B. führt $\alpha = \pi$, $\beta = 0$ und $\gamma = 0$ auf die selbe Drehmatrix wie $\alpha = 0$, $\beta = \pi$ und $\gamma = \pi$, und nicht nur die Minima sondern auch die Maxima lösen das System. Eindeutig ist die Lösung, wenn die Winkel α und γ auf das Intervall $(-\pi, \pi]$ und β auf das Intervall $(\frac{-\pi}{2}, \frac{\pi}{2}]$ beschränkt werden, und wenn folgende zusätzliche Voraussetzungen gelten:

- Der Nickwinkel β darf nicht $\frac{\pi}{2}$ sein. Dann ist nur noch die Summe $\alpha+\gamma$ eindeutig bestimmbar, nicht aber, wie diese Summe verteilt wird auf den Gier und Rollwinkel. Wenn β nahe bei $\pm\frac{\pi}{2}$ liegt, wird die Lösung instabil. *template_matching* wird dann auf 'falsch' gesetzt, und es liegt also keine Ausgangskonfiguration vor (das Fahrzeug würde dann ohnedies 'auf der Nase' oder auf dem Heck stehen).
- Das Modell muß die Eindeutigkeit erzwingen. Man braucht mehr als zwei Modellpunkte, und diese dürfen nicht in einem Ort zusammenfallen oder auf einer Geraden liegen. In der Nähe solcher Lagen wird die Lösung instabil.
- Die Attributwerte müssen die Eindeutigkeit erzwingen. Man braucht mehr als zwei Attributwerte, und diese dürfen nicht in einem Ort zusammenfallen oder auf einer Geraden liegen. In der Nähe solcher Lagen wird die Lösung instabil.

Die geschlossene Lösung des nichtlinearen Gleichungssystems macht Schwierigkeiten. Man kann aber das System separieren in einen Teil zur

Bestimmung der Translation und einen zur Bestimmung der Rotation. Zur Bestimmung der Rotation wurde ein Iterationsverfahren nach Newton verwendet. Dabei werden die zweiten partiellen Ableitungen von $\mathcal{S}$ nach den Drehwinkeln an der aktuellen Stelle benötigt. Es entsteht ein dreidimensionales, lineares Gleichungssystem, das mit Hilfe der Adjunkten schnell zu lösen ist. Es wird abgefragt, ob der Betrag der Determinante eine Schwelle ϵ unterschreitet. Ist das der Fall, so liegt eine der oben beschriebenen Instabilitäten vor, und *template_matching* erhält den Wert 'falsch'.

Das Newtonverfahren benötigt eine Ausgangsnäherung für die Drehwinkel. Diese wird aus der *E_structure* der Ausgangskonfiguration bestimmt. Da diese von der endgültigen Lösung nicht sehr weit entfernt sein kann, wird das Verfahren nicht das Maximum sondern das Minimum finden, und es genügen wenige Iterationen. Die Anzahl der Iterationen ist ein Verfahrensparameter und in Anhang C dokumentiert.

Ist die Rotation bestimmt, so kann die Translation durch Einsetzen in den separierten Teil des Gleichungssystems berechnet werden. Rotation und Translation werden dann an die Funktion *ls_sqr_tr* weitergereicht, die die Translation in das Attribut ver_1 schreibt, und die Rotation (α, β, γ), die Fehlerquadrat Summe $\mathcal{S}$ und die Anzahl der gefundenen Teile $k' \leq k = 4$ in die entsprechenden Attribute kopiert.

Der Abstand zu jeder Korrespondenz wird bestimmt. Wenn einer dieser Abstände den Verfahrensparameter d_{max} (siehe Anhang C) überschreitet, wird *template_matching* auf 'falsch' gesetzt.

- *spin* liefert den Drehsinn eines Polygonzugs P_1, P_2, P_3 in der Ebene.

  $$spin(P_1, P_2, P_3) \stackrel{def}{=} sgn((P_2 - P_1)_x \cdot (P_3 - P_2)_y - (P_2 - P_1)_y \cdot (P_3 - P_2)_x)$$

  Bei vier Punkten impliziert $spin(P_1, P_2, P_3) = spin(P_2, P_3, P_4)$, daß keine 'Z'-Konfiguration vorliegt. Es ist allerdings dadurch nicht ausgeschlossen, daß sich das erste und das dritte Segment kreuzen. Will man solche 'entarteten' offenen Vierecke vermeiden, so kann man noch bzgl. der Orientierungsattribute $par(O_1, O_3)$ verlangen (allerdings darf man da nicht sehr stark einschränken, da die zentralperspektivische Projektion Parallelität nicht erhält).

- *worst_element* bestimmt nach Aufruf von *calculate_regression* den Punkt P_{worst} aus der Punktmenge mit der maximalen Abweichung von der Geraden $|n \cdot P_{worst} - \alpha| \stackrel{!}{=} Max$, und hat als Ergebnis die Instanz, zu der dieser Wert gehört. Nach Entfernung der Punkte dieser Instanz aus der Punktmenge muß die Regressionsberechnung korrigiert werden. Der Wert von *worst_element* hängt nicht von der Aufzählungsreihenfolge der Punktmenge ab.

Appendix C

Verfahrensparameter des Beispiel Programms

Es gibt viele Methoden, eine geeignete Einstellung der Verfahrensparameter zu finden. Z. B. wurden für das BPI-System auch Evolutionsstrategien für diesen Zweck untersucht [STEI]. Neben dem Studium solcher und anderer 'Lernverfahren' sollte man sich vor allen Dingen, wie im Abschnitt 5.3 motiviert, darum bemühen, die Schwellwerte und anderen Verfahrensparameter einerseits nach der Semantik der Symbole zu richten, und andererseits so eng zu wählen, daß es nicht zu einer Überbelastung der vorhandenen Hardware kommt. Eine KG kann als robust betrachtet werden, wenn zwischen der Mindesttoleranz, die die Semantik verlangt, und der Maximaltoleranz, über der die Anzahl der Instanzen zu rasch wächst, hinreichend viel Spielraum verbleibt. Die folgenden Überlegungen beziehen sich auf die Parameter der KG aus dem Kapitel 6.

1. Die **Anzahl der Schwellwerte**, mit denen die Grauwertbilder binärisiert werden, entscheidet darüber, wie groß der Kontrast einer Kante im Bild sein muß, damit sie als solche sicher gefunden wird. In den hier dokumentierten Beispielläufen lag dieser Parameter auf 12 Schnitten, die in gleichen Abständen zwischen dem kleinsten und dem größten, vorkommenden Grauwert liegen. Einstellungen zwischen 8 und 24 Schnitten haben sich im Laufe der langjährigen Arbeiten mit dieser Terminalextraktion im sichtbaren Spektralbereich bewährt. Im Prinzip gilt, daß die erhöhte Zahl an Terminalen, die durch Hinzunahme von mehr Schwellen erzeugt wird, das Verfahren stark belasten kann. Die im Abschnitt 5.2.5 beschriebene Linienverlängerungsproduktion wirkt dem aber zum Teil entgegen. Es wird hier ja nur nach den jeweils maximalen, kollinearen Teilmengen gefragt.

2. Die **Parameter der Polygonzugsapproximation**. Hier gibt es eine Schrittweite, eine Rückschrittweite und eine Approximationsmindestgüte

(siehe [FUE-88]); Diese Parameter entscheiden darüber, wie groß ein Segment mindestens sein muß, damit es noch approximiert wird, und wie genau Grauwertecken durch Polygonpunkte getroffen werden. Es können hier Überlegungen zur Größe der Modellflächen zusammen mit der Maximalentfernung, dem Öffnungswinkel, der Abtastung usw. eingehen. In den hier dokumentierten Beispielläufen lag die Schrittweite auf 20 und die Rückschrittweite auf 10 Pixel. Das sind, genauso wie die gewählte Einstellung zur Approximationsgüte, standard Default-Einstellungen, die sich für solches Datenmaterial bewährt haben. Wie oben gilt, daß die Anzahl der erzeugten Terminale zwar stark von diesen Parametern abhängt, aber daß die Linienverlängerungsproduktion diesen Effekt teilweise wieder beseitigt. Die Auswirkungen auf die Zahl der verlängerten Linieninstanzen sind geringer.

3. Der **Kollinearitätsparameter** d_{max} der Relation *col_set*. Der Abschnitt 6.3 erläutert die zugrundeliegende Problematik. Hier sind also Probleme der Kontrastübertragung des Objektivs (siehe [ZEISS]) und Überstrahlungseffekte durch Reflektionen im Filmmaterial zu berücksichtigen. Daraus ergibt sich, wie breit eine ideale Kante im Bild verschmieren kann. Darüberhinaus sind die Konzepte bzw. Modelle für das zu findende Objekt zu untersuchen. Die gerade modellierten Fahrzeugkanten sind nur mit einer gewissen Toleranz gerade. Auch liegen keine scharfen Knicke vor, sondern abgerundete Falze. Die dadurch hervorgerufene maximale Verbreiterung einer Kante hängt invers von der minimalen Objektentfernung ab. Es wurde hier durchgehend ein Maximalwert von 2 Pixel Abstand von der Regressionsgerade verwendet.

4. In der Relation *col_set* ist auch die Forderung nach **Parallelität** enthalten. Die im Abschnitt 6.1 dokumentierten Ergebnisse aus der Bildfolge BF1 wurden erzielt mit einer Toleranzschwelle von 2 Grad an dieser Stelle. Bei dieser Parameterwahl müssen Linieninstanzen etwa dreißig Pixel lang sein, damit ein Versatz von einem Pixel noch akzeptiert wird. Das ist unnötig eng: Beim Test des Verfahrens mit der Bildfolge BF2, der im Abschnitt 6.5 dokumentiert ist, ergab die Analyse des Ergebnisses mit Hilfe der Erklärungswerkzeuge (siehe Abschnitt 5.1), daß dieser Parameter semantisch falsch gewählt war. Auch bei einer nur zehn Pixel langen Linieninstanz sollte ein Versatz von einem Pixel noch akzeptiert werden. Die Schwelle wurde daher erhöht auf fünf Grad. Wesentliche Auswirkungen bzgl. der Rechenkomplexität sind dabei nicht festgestellt worden. Es erhöhen sich lediglich die Anzahlen der Instanzen, die in die Regression bei der Berechnung der einzelnen verlängerten Linien eingehen. Die Anzahl der verlängerten Linien ändert sich kaum. Dieser Parameter war der einzige, der sich beim Verfahrenstest als falsch gewählt herausstellte.

5. Der **Schwellwert des Nachbarschaftsprädikates** adj_∞ in den Produktionen p_2 und p_3 kann zum Teil motiviert werden durch Unzulänglichkeiten in der Bildvorverarbeitung und Linienextraktion, und zum Teil geschlossen werden aus dem Modell oder Konzept für das zu findende Objekt. Das Fahrzeug hat keine idealen Ecken, sondern die als Ecken modellierten Stellen sind rund und haben daher eine räumliche Ausdehnung. Zusammen mit einer minimalen Entfernung von einigen Metern ergeben sich daraus etwa 10 Pixel im Bild. Aus der oben dokumentierten Einstellung der Parameter der Polygonzugsapproximation ergeben sich ähnliche Werte für den *worst case*, daß eine Ecke schlechtest möglich getroffen wird. Dieser Wert wurde also als Schwelle verwendet. Da die damit erzielten Statistiken wie am Ende des Abschnitts 6.3 dokumentiert bereits die Faustregeln aus dem Abschnitt 5.3 zu verletzen beginnen, ist nicht mehr viel Raum den Parameter wesentlich größer zu wählen. Dieser Parameter ist kritisch.

6. Der **Schwellwert** für den Abstand der Strahlen in der **Stereotriangulation** hängt von den maximalen Ungenauigkeiten bei der Bestimmung der Kameraparameter ab. Für Fehler in der Kamerapositionierung ist er konstant, für Fehler in der Orientierung sollte er von der Entfernung, also von λ abhängig gemacht werden. Hinzu kommen Fehler aus der Vorverarbeitung und den vorangehenden 2D Produktionen. Schätzt man den Fehler in der Positionierung der 2D Attribute auf maximal 3 Pixel, so kommt man in den interessierenden Entfernungen auf $2cm$ bis $6cm$ Abstand der Strahlen. Dann rechnet man grob noch $2cm$ Toleranz hinzu für die Kameraparameter. Das ergibt einen Schwellwert von $8^2 = 64cm^2$. Man braucht noch etwas Sicherheit. Für den hier dokumentierten Lauf wurden daher $10^2 = 100cm^2$ gewählt. Wenn also zwei Strahlen um weniger als $10cm$ aneinander vorbeigehen, werden sie als korrespondierend akzeptiert.

   Mit dieser Einstellung ergeben sich im vorliegenden Bildmaterial aus 163774 2D *open_quad* Instanzen 12997 3D *U_structure* Instanzen. Das ergibt einen Faktor von 0.079. Demnach könnte dieser Parameter gemäß der ersten Faustregel aus Abschnitt 5.3.1 wesentlich angehoben werden. Allerdings geht aus den Überlegungen im Abschnitt 5.3.3 hervor, daß die Anzahl der *E_structure* Instanzen quadratisch mit der Anzahl der *U_structure* Instanzen wachsen wird. Es ist anzunehmen, daß diese Zahl sehr rasch mit zunehmendem Schwellwert ansteigt. Man kann versuchen, das Relativvolumen der Produktion mit den Verfahren aus Abschnitt 5.3.2 grob zu bestimmen: Stellt man sich einen Zylinder mit dem Radius des Schwellwertes um eine feste Gerade herum vor, und betrachtet die Anzahl der diesen Zylinder schneidenden Geraden aus einer gleich im Raum verteilten Geradenmenge, so sieht man, daß diese Anzahl etwa quadratisch mit dem Radius steigen wird. Nun unterscheidet man noch zwei Extremfälle: 1. Die 4 Strahlen zu einer Instanz sind völlig unabhängig voneinander. Dann führt eine Ver-

doppelung des Parameters auf $20^2 = 400cm^2$ auf einen Faktor von etwa $4^4 = 256$. Man müßte also dann mit drei Millionen *U_structure* Instanzen und mit *einer Milliarde E_structure* Instanzen rechnen. 2. Ein Strahl der *open_quad* Instanz unterschreitet den Schwellwertabstand zu einem korrespondierenden Strahl einer Partnerinstanz genau dann, wenn dies alle vier tun. Das ergibt einen Faktor 4 bei den *U_structure* Instanzen (also etwa Fünfzigtausend) und einen Faktor 16 bei den *E_structure* Instanzen (ungefähr eine halbe Millionen). Die tatsächlichen Zahlen werden dazwischen liegen. Soweit wird man nicht gehen wollen. Allenfalls eine mäßige Steigerung auf etwa $12^2 = 144cm^2$ erscheint möglich. Damit stehen zwischen Zwanzig- und Fünfzigtausend *U_structure* Instanzen und zwischen Fünfzig- und Vierhunderttausend *E_structure* Instanzen zu erwarten.

Tatsächlich stößt man bei der Konstruktion des Reduktionsgraphen der in Abschnitt 6.1 dokumentierten 'besten' *light_truck* Instanz (in Abbildung 6.5 ist sie schwarz dargestellt) auf eine Korrespondenz zweier Strahlen, die etwa $9.3cm$ aneinander vorbeigehen. Der Wert für die 15 anderen Korrespondenzen ist im Intervall bis $8cm$. 7 Abstände sind kleiner als $5cm$. Demzufolge wird die Instanz mit der kleinsten Fehlerquadratsumme nicht konstruiert, wenn der Schwellwert auf $8^2 = 64cm^2$ liegt. Die in Abbildung 6.5 rot dargestellten vier subjektiv besten Instanzen haben eine nur um etwa 10% größere Fehlerquadratsumme und werden auch mit diesem Schwellwert noch gefunden. Der Schwellwert kann also in einem geschätzten Intervall zwischen $8^2 = 64cm^2$ und $12^2 = 144cm^2$ gewählt werden.

7. In der Stereotriangulation wird eine *U_structure* Instanz nur dann konstruiert, wenn sie das Prädikat *geom_const* erfüllt. Darin sind **Grenzen** für **geometrische Abmaße** und **Winkel** aufgenommen. Diese ergeben sich aus dem Modellwissen über das zu findende Objekt. Für das Fahrzeug werden Abmaße zwischen 30cm und 300cm akzeptiert. Es treten nur rechte Winkel auf. Für die Fensterteile im Modell stimmt das nicht ganz. Da die 3D Koordinaten außerdem mit erheblichen Unsicherheiten gemessen sind, wurde für den Cosinus des Winkels das Intervall (-0.33, 0.33) zugelassen, was einer Abweichung von etwa 20^0 entspricht.

8. Das Prädikat *geom_const* beinhaltet darüberhinaus, daß **gegenüberliegende Seiten** des gefundenen Vierecks **etwa gleich lang** sein sollen. Es wird also näherungsweise Parallelogrammform gefordert, was mit dem etwa rechten Winkel zusammen einer Forderung nach etwa rechteckiger Gestalt entspricht. Diese Forderung impliziert, daß nur ein Winkel geprüft werden muß. Für die maximale Abweichung in der Länge gegenüberliegender Seiten wurde ein Faktor 0.2 gesetzt. Diese beiden Verfahrensparameter (Winkeltoleranz und Parallelogrammtoleranz) wurden bei der Verfahrensentwicklung mit der Bildfolge

BF1 so gewählt, daß sich robuste Ergebnisse ergaben. Sie haben sich auch am geänderten Modell und unter den geänderten Bedingungen (z. B. bzgl. Entfernung und Triangulationswinkel) der Bildfolge BF2 noch als brauchbar erwiesen.

Über das Wachstum der Anzahl an akzeptierten *U_structure* Instanzen als Funktion dieser beiden Parameter läßt sich aufgrund der Dimensionalität vermuten, daß es sich lokal jeweils annähernd linear verhält. Im Anhang D.2 wurden Untersuchungen mit zufällig erzeugten konvexen 3D Polygonen, gemacht die die Abmaß Bedingung erfüllten. Nur etwa jede vierzigste erfüllt dann auch die Parallelogrammbedingung und ist rechtwinklig (siehe Tabelle D.10). Wenn man die Erzeugung so ändert, daß auch die Parallelogrammbedingung erfüllt ist, ist eines von drei Vierecken hinreichend rechtwinklig (siehe Tabelle D.10). Diese Versuche wurden mit gleichverteilten Zufallspolygonen gemacht. Die Erfahrung mit den durch die Stereoproduktion aus den Bildfolgen gewonnen *U_structure* Instanzen lassen eher vermuten, daß das Wachstum hier noch stärker ausfällt (Instanzen aus Fehlkorrespondenzen sind sehr häufig und meistens ziemlich spitzwinklig, so daß sie an dieser Stelle 'ausgesiebt' werden).

9. Für die Produktionen p_1 (Konstruktion der *E_structures* Instanzen) wird die **3D Nachbarschaftsrelation** adj_q verwendet. Der Einfachheit halber ist $q = \infty$ (also nicht die euklidische, sondern die Maximumsnorm) gesetzt. Dadurch geht, streng genommen, die Rotationsinvarianz der Produktionen verloren. Die Mindesttoleranz ergibt sich auch hier aus Fehlerabschätzungen. Die im CAD-Modell und damit im Nachbarschaftgraph dargestellten Nachbarschaften sind am tatsächlichen Fahrzeug nicht genauer als auf etwa $10cm$ realisiert. Dieses Fahrzeug ist ja nicht wirklich eckig, sondern hat abgerundete Flächen. Dieser Fehler fällt allerdings weniger ins Gewicht gegenüber den Fehlern in der Tiefenbestimmung durch die Stereoproduktion. Bei einer Stereobasis von $4m$, einer Objektentfernung von $20m$ und einem Fehler in der Positionierung der 2D Attribute von 3 Pixel kann sich ein Fehler von etwa $30cm$ in der Tiefe einstellen. Man braucht noch etwas Sicherheit. Für die hier dokumentierten Läufe wurden daher $50cm$ gewählt. Unterschreitet das Maximum der Beträge der Differenzen der Koordinaten zweier Punkte diesen Wert, so gelten sie als benachbart.

   Bei der Reduktion der *E_structure* Instanz, die der in Abschnitt 6.1 dokumentierten 'besten' *light_truck* Instanz vorangeht, liegt das Maximum dieser Werte bei $29cm$. Für den zugehörigen Modellvergleich sind tatsächlich genau $30cm$ erforderlich.

10. In der Produktion p_2 (Modell Vergleich) wird die gleiche Schwelle $d_{max} =$

$50cm$ verwendet. Jedoch liegt dem Prädikat *template_matching* die euklidische Norm zugrunde.

11. In der Produktion p_1 (Konstruktion der *E_structure* Instanzen) der 3D Grammatik wird das Prädikat *prl* verwendet für die **Parallelität** zweier Oberflächennormalen. Der euklidische Abstand der normierten Vektoren im Raum liefert ein Maß für den Sinus des eingeschlossenen Winkels. Was hier mindestens zu tolerieren ist, muß ebenfalls aus den möglichen Fehlern der Stereotriangulation bestimmt werden. Die *U_structure* Instanzen sind etwa $200cm$ lang und $100cm$ hoch. Bei einem Fehler von $30cm$ in der Lage können sich leicht Fehler von 15^o in der Orientierung einstellen.

    Bei der Reduktion der *E_structure* Instanz, die der in Abschnitt 6.1 dokumentierten 'besten' *light_truck* Instanz vorangeht, stehen die Normalen in einem Winkel von 10^o zueinander. Mit einem Schwellwert von 0.15 würde sie gerade noch gefunden werden, weil nicht der euklidische Abstand für den assoziativen Zugriff genommen wird, sondern wiederum die Maximumsnorm ($q = \infty$). Der tatsächlich verwendete Schwellwert ist für den dort dokumentierten Lauf auf 0.2 gesetzt. Das neuere Modell, welches für die Analyse der Bildfolgen BF2 und BF3 im Abschnitt 6.5 verwendet wurde, hat wesentlich flachere Seitenflächen. Dieser Parameter ist damit anzupassen auf etwa 0.3. Im dort dokumentierten Lauf wurde 0.25 verwendet.

    Dem Attribut 'Oberflächennormale' kommt gemäß der Definition aus Abschnitt 5.1.3 die Dimension zwei zu. Setzt man Gleichverteilung der zugehörigen Werte an, so wird also ein quadratisches Wachstum in der Anzahl der Ausgangskonfigurationen dieser Produktion mit variierendem Schwellwert zu erwarten sein, weil das Relativvolumen quadratisch wächst. Insbesondere in den in Abschnitt 6.5 dokumentierten Läufen ergeben sich bereits wesentlich mehr *E_structure* Instanzen als *U_structure* Instanzen. Die Faustregel 1 aus dem Abschnitt 5.3.1 ist schon verletzt. Offenbar ist dies also ein gutes Beispiel für einen Schwellwert, bei dem nicht genug Spielraum verbleibt zwischen der Mindesttoleranz, die durch die Modalitäten der Messung gefordert ist, und der Maximaltoleranz, die durch den kombinatorischen Aufwand gerechtfertigt erscheint.

12. In der Produktion p_2 (Modell Vergleich) der 3D Grammatik wird die 3D Rotation durch ein Newtonverfahren iteriert. Die **Anzahl der Iterationen** ist ein Verfahrensparameter. Für die im Abschnitt 6.1 dokumentierten Ergebnisse wurde nullmal iteriert. Die Ausgangsnäherung erschien gut genug. Für die im Abschnitt 6.5 dokumentierten Ergebnisse wurde dreimal iteriert. Bei diesem neueren Fahrzeugmodell sind die senkrechten Kanten der *E_structure* wesentlich kürzer, und bei der größeren Objektentfernung in der Bildfolge BF2 und den spitzeren Triangulationswinkeln ist die Ausgangsnäherung somit instabiler. Die Ergebnisse mit null Iterationen waren

um etwa 5^o falsch, und sahen in der Rückprojektion schief aus. Zudem waren die besten falsch herum orientierten Instanzen genauso gut wie die richtig orientierten. Erst mit ein oder zwei Iterationen ergaben sich befriedigende Rotationen, und die richtig orientierten Instanzen konnten anhand der Fehlerquadratsumme von den falsch orientierten zuverlässig separiert werden. Mehr als drei Iterationen sind kaum erforderlich, da die Winkelattribute in ganzen Graden gerastert sind.

Im Anhang D sind Untersuchungen dokumentiert, bei denen die in den Beispielen verwendeteten KGs auf Instanzenmengen arbeiten, die mit einem Zufallsgenerator synthetisch erzeugt wurden. Dann variieren nicht die Schwellwerte, sondern die Anzahlen der Instanzen. Es ergeben sich dabei ähnliche Erkenntnisse.

In der modellvergleichenden Produktion p_2 der 3D Grammatik wird das in Abbildung 6.1 dargestellte Modell verwendet. Gemäß den in Abschnitt 5.2.2 eingeführten Notationen ist das Modell für das ältere Fahrzeug aus der Bildfolge BF1 gegeben durch die Tabellen C.1 und C.2. Die Punkteliste ist die gleiche. Die Korrespondenzen sind unterschiedlich je nachdem, welche Seite des Fahrzeugs durch die Produktion gesucht wird. Darüberhinaus sind gemäß den in Abschnitt 5.2.3 eingeführten Begriffen und wie im Abschnitt 6.1 motiviert für die *U_structure* Instanzen in der Ausgangskonfiguration jeweils die zyklischen Indexpermutationen (1), (1,2,3,4), (1,3)(2,4) und (1,4,3,2) als Rekombination zugelassen. Insgesamt ist dieser Modell-Vergleich also in $2 \cdot 4 \cdot 4 \cdot 4 = 128$ Korrespondenzvarianten in der verwendeten KG enthalten. Das neuere Fahrzeug aus den Bildfolgen BF2 und BF3 hat flachere Seitenwände und dafür höhere Fenster. Es ändert sich in der Z Koordinate der Wert 30 auf den Wert 60 und der Wert 160 auf den Wert 180. Alle anderen Koordinaten und alle Korrespondenzen bleiben gleich.

Name	X Koord.	Y Koord.	Z Koord.	Kommentar	Korrespondenz
rechte Seitenfläche					
p_1	220	-90	30	vorn unten	$\{(1,1),(2,1)\}$
p_2	220	-90	110	vorn oben	$\{(1,2),(2,4),(4,1)\}$
p_3	0	-90	110	mittig oben	$\{(1,3),(4,4)\}$
p_4	-200	-90	110	hinten oben	$\{(1,5),(3,1)\}$
p_5	-200	-90	30	hinten unten	$\{(1,6),(3,4)\}$
p_6	0	-90	30	mittig unten	$\{(1,4)\}$
linke Seitenfläche					
p_7	220	90	30	vorn unten	$\{(2,2)\}$
p_8	220	90	110	vorn oben	$\{(2,3)\}$
p_9	0	90	110	mittig oben	$\emptyset$
p_{10}	-200	90	110	hinten oben	$\{(3,2)\}$
p_{11}	-200	90	30	hinten unten	$\{(3,3)\}$
p_{12}	0	90	30	mittig unten	$\emptyset$
Dachfläche					
p_{13}	0	85	160	mittig links	$\emptyset$
p_{14}	0	-85	160	mittig rechts	$\{(4,3)\}$
p_{16}	210	-85	160	vorn rechts	$\{(4,2)\}$
p_{16}	210	85	160	vorn links	$\emptyset$

Table C.1: **Modell des Fahrzeugs VW-Bus 'Doppelkabiner' (rechte Seite)**

Name	X Koord.	Y Koord.	Z Koord.	Kommentar	Korrespondenz
rechte Seitenfläche					
p_1	220	-90	30	vorn unten	$\{(2,1)\}$
p_2	220	-90	110	vorn oben	$\{(2,4)\}$
p_3	0	-90	110	mittig oben	$\emptyset$
p_4	-200	-90	110	hinten oben	$\{(3,1)\}$
p_5	-200	-90	30	hinten unten	$\{(3,4)\}$
p_6	0	-90	30	mittig unten	$\emptyset$
linke Seitenfläche					
p_7	220	90	30	vorn unten	$\{(1,6),(2,2)\}$
p_8	220	90	110	vorn oben	$\{(1,5),(2,3),(4,1)\}$
p_9	0	90	110	mittig oben	$\{(1,3),(4,2)\}$
p_{10}	-200	90	110	hinten oben	$\{(1,2),(3,2)\}$
p_{11}	-200	90	30	hinten unten	$\{(1,1),(3,3)\}$
p_{12}	0	90	30	mittig unten	$\{(1,4)\}$
Dachfläche					
p_{13}	0	85	160	mittig links	$\{(4,3)\}$
p_{14}	0	-85	160	mittig rechts	$\emptyset$
p_{16}	210	-85	160	vorn rechts	$\emptyset$
p_{16}	210	85	160	vorn links	$\{(4,4)\}$

Table C.2: **Modell des Fahrzeugs VW-Bus 'Doppelkabiner' (linke Seite)**

Appendix D

Verhalten der Beispielgrammatiken bei synthetischen Eingangsdaten

Die in dieser Arbeit insbesondere im Kapitel 6 beschriebenen KGs sind an Daten entwickelt worden, die durch Abtastung und Digitalisierung von photographischen Aufnahmen entstanden sind. Man kann aber als Eingangsdaten auch Mengen von Instanzen verwenden, die mit einem Zufallsgenerator erzeugt worden sind. Es ergeben sich dabei Einsichten in ihre Robustheit, ihr Skalierungsverhalten mit ansteigenden Instanzenzahlen und in ihre Fehlerraten. Der Vorteil liegt hier in der Kontrollierbarkeit der Erzeugung der Ausgangsdaten. Insbesondere kann die Verteilung gewählt und variiert werden. Allerdings dürfen die Ergebnisse nicht überinterpretiert werden. Auf realen Daten ist das Verhalten erfahrungsgemäß wesentlich instabiler. Das läßt den Umkehrschluß zu, daß die Verteilung tatsächlich gemessener Instanzenmengen im Attributraum wesentlich komplexer ist als die Verteilung synthetisch erzeugter Instanzenmengen.

Der Abschnitt D.1 behandelt eine Reihe von Versuchen, die mit der Produktion zur Konstruktion von Instanzen des Symbols *Winkel* aus Abschnitt 2.1.3 durchgeführt wurden. Der Schwerpunkt liegt auf dem praktischen Nachweis der linearen Rechenkomplexität einer solchen Produktion auf einer hinreichend dimensionierten Spezialmaschine mit assoziativem Zugriff, im Gegensatz zu quadratischer Rechenkomplexität beim Lauf auf einer Auswahl von Standardmaschinen. Deutlich werden auch die gegenwärtig realistischen Rechenzeiten.

Der Abschnitt D.2 verwendet die Beispiel KG aus dem Kapitel 6. Deutlich wird hier die Reaktion eines solchen Produktionssystems als Ganzes auf Veränderungen in der Verteilung der Eingangsdaten. Es kann auf diese Art plausibel gemacht werden, warum es sehr unwahrscheinlich ist, daß ein Eingangsdatensatz als zur Sprache gehörig akzeptiert wird, in dem kein solches Fahrzeug

n_{Linie}	10	20	30	40	50	60	70	80	90	100
n_{Winkel}	0	0	2	9	7	10	13	16	10	14
$t_{pas,4090}$	0	0	0	0	10	10	10	10	30	30
$t_{pas,\alpha}$	nicht gemessen									
$t_{bpi,4090}$	40	60	100	150	170	220	290	300	310	350
$t_{bpi,WPP1}$	320	570	880	1140	1330	1690	1960	2150	2430	2710
$t_{c,SUN}$	nicht gemessen									

Table D.1: **Rechenzeiten für Bildgröße** 256^2 **Pixel**

n_{Linie}	100	200	300	400	500	600	700	800	900	1000
n_{Winkel}	2	5	10	26	24	48	55	86	122	138
$t_{pas,4090}$	20	80	190	310	530	790	1060	1380	1730	2160
$t_{pas,\alpha}$	nicht gemessen									
$t_{bpi,4090}$	370	730	1380	1880	2720	2940	3620	4370	5490	6120
$t_{bpi,WPP1}$	2640	5210	7650	10380	12670	15520	17980	21100	23700	26240
$t_{c,SUN}$	nicht gemessen									

Table D.2: **Rechenzeiten für Bildgröße** 1024^2 **Pixel**

enthalten ist.

D.1 Versuche mit einer 2D Produktion

Die folgenden Versuche wurden mit Beispieldatensätzen erstellt, die mit einem Programm erzeugt wurden, welches einen einfachen Standardzufallsgenerator aufruft. Die darin enthaltenen Instanzen des Symbols *Linie* sind in ihrer Position und Orientierung gleichverteilt und in der Länge zwischen 10 und 100 Pixel gleichverteilt. Position, Orientierung und Länge sind unabhängig voneinander. Abbildung D.1 zeigt einen solchen Datensatz in einem Bild mit 256^2 Pixeln.

Die Tabellen D.1 bis D.4 geben die Rechenzeiten für die Produktion aus dem Abschnitt 2.1.3 an. Eine Instanz des Symbols *Winkel* wird erzeugt, wenn sich zwei Endpunkte hinreichend nahekommen (in diesem Falle 10 Pixel), und der Schnittpunkt der zugehörigen Geraden existiert und im Bildbereich liegt. n_{Linie} ist die Anzahl der Instanzen des Symbols *Linie* und n_{Winkel} die Anzahl der Instanzen des Symbols *Winkel*. Die Rechenzeiten sind in Millisekunden angegeben, wie sie das Betriebssystem liefert.

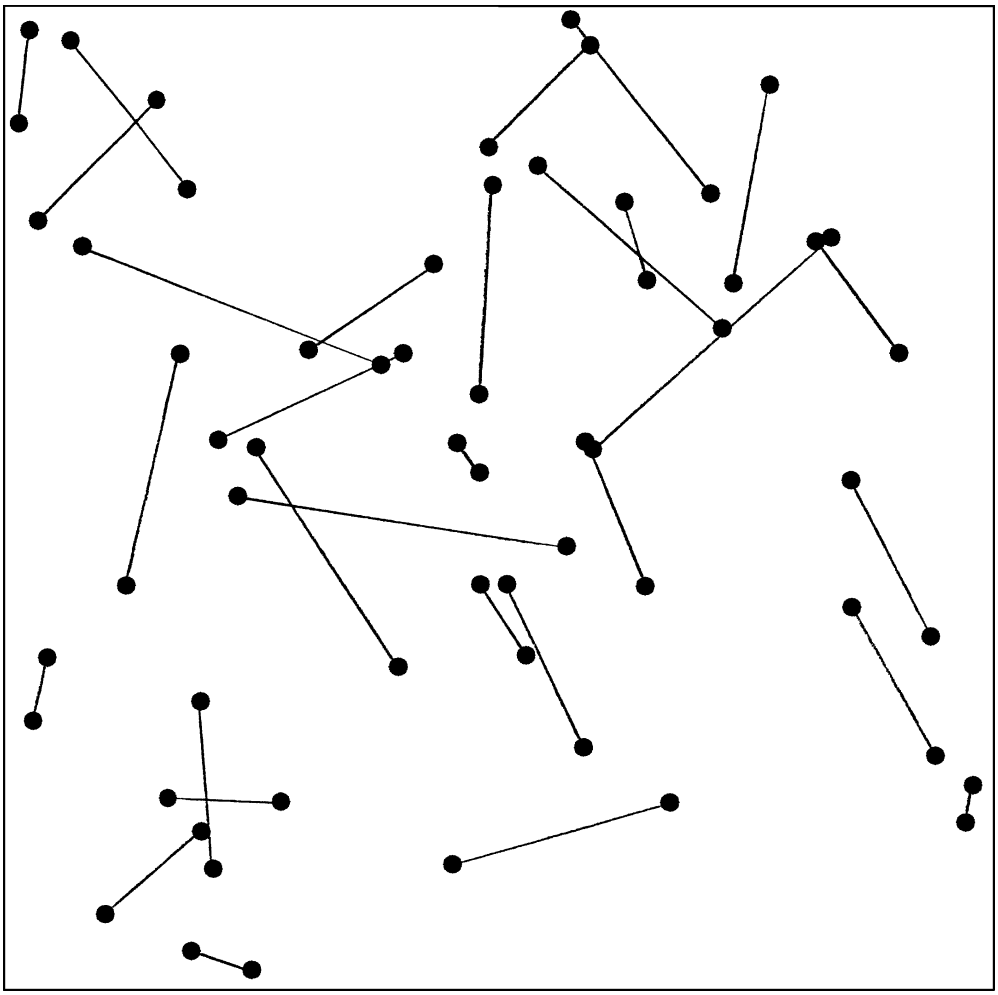


Figure D.1: **Beispiel eines synthetischen Datensatzes**
30 Linien Instanzen in einem Bild mit 256^2 Pixeln

n_{Linie}	1000	2000	3000	4000	5000
n_{Winkel}	6	34	76	125	166
$t_{pas,4090}$	2130	8820	19570	34820	54570
$t_{pas,\alpha}$	590	2480	5580	10080	15570
$t_{bpi,4090}$	9060	27180	57520	98740	150780
$t_{bpi,WPP1}$	25160	50890	75580	101200	129140
$t_{c,SUN}$	710	2910	6620	11760	18310

Table D.3: **Rechenzeiten für Bildgröße** 4096^2 **Pixel I**

Bildgröße 4096^2 Pixel:					
n_{Linie}	6000	7000	8000	9000	10000
n_{Winkel}	295	421	503	657	842
$t_{pas,4090}$	81300	108420	141790	181260	228280
$t_{pas,\alpha}$	22440	31050	39830	50480	62260
$t_{bpi,4090}$	214880	294730	391820	481700	599920
$t_{bpi,WPP1}$	150710	181700	203640	232640	266890
$t_{c,SUN}$	26340	35740	46710	60190	73460

Table D.4: **Rechenzeiten für Bildgröße** 4096^2 **Pixel II**

Im einzelnen wurden folgende Werte ermittelt:

- $t_{pas,4090}$: Rechenzeit mit PASCAL (ohne Debugger und mit Optimierung, Stand etwa 1994) auf einer VAX 4000/90 der Firma DIGITAL.
- $t_{pas,\alpha}$: Rechenzeit mit PASCAL (ohne Debugger und mit Optimierung Stand etwa 1994) auf einer VAX ALPHA der Firma DIGITAL.
- $t_{bpi,4090}$: Rechenzeit mit BPI (mit Debugger, ohne Optimierung, Benutzung der Wissenserklährungskomponenten möglich, Stand etwa 1992) auf einer VAX 4000/90 der Firma DIGITAL.
- $t_{bpi,WPP1}$: Rechenzeit mit BPI (mit Debugger, ohne Optimierung, Benutzung der Wissenserklährungskomponenten möglich, Stand etwa 1992) auf der im Auftrag des FIM entwickelten WPP1-Spezialmaschine (1990 in Dienst gestellt) mit einer VAX 3020 der Firma DIGITAL als Master.
- $t_{c,SUN}$: Rechenzeit mit C (ohne Debugger und mit Optimierung Stand etwa 1994) auf einer SUN SS20/61.

Deutlich wird die etwa quadratische Rechenkomplexität. Die Angaben des Herstellers DIGITAL über den Rechenzeitfaktor zwischen den Modellen ALPHA und 4000/90 bestätigen sich. Die inzwischen eher veraltete VAX 3200 ist laut Hersteller gegenüber der 4000/90 um eine ganze Zehnerpotenz langsamer. Das wäre gegen die aktuellen ALPHA und SUN Maschinen ein Faktor 40. Daß die Kombination WPP1/VAX 3200 trotzdem bei größeren Datensätzen hier noch mithält, liegt am assoziativen Speicherzugriff (siehe Abschnitt 4.1). Die Pascal- und die C-Version dieses Programms dienen inzwischen im FIM gelegentlich als *benchmark* Programme. In rascher Folge werden neue Rekorde für das 10000 Linien Problem gemeldet. Ohne Parallelisierung ist eine kürzlich in Betrieb genommene ALPHA mit sechs Prozessoren etwa doppelt so schnell wie die alte. Mit Parallelisierung wäre wohl eine ganze Zehnerpotenz drin. In der C-Version ergab

Bildgröße	Verfahren	Konstant	Linear	Quadratisch
256^2	4000/90 BPI	1.6	3.7	-0.0020
256^2	WPP1	-3.7	29.2	-0.0200
1024^2	4000/90 PAS	0.5	-0.035	0.0022
1024^2	4000/90 BPI	1.1	3.6	0.0025
1024^2	WPP1	-0.2	25.3	0.0011
4096^2	4000/90 PAS	-151.5	-0.6	0.0023
4096^2	4000/90 BPI	-304.4	1.2	0.0059
4096^2	WPP1	0.0	24.5	0.0002
4096^2	ALPHA PAS	0.2	0.03	0.0006
4096^2	SUN C	-11.9	-0.05	0.0007

Table D.5: **Empirische Parameter der Rechenkomplexität**

alleine schon das Umsteigen auf eine neue Betriebssystemversion (incl. neuem Compiler) einen Sprung auf etwa 54 Sekunden. Von dieser Version gibt es eine parallelisierte Variante, die auf einer SUN SS20/71 mit sechs Prozessoren 6,4 Sekunden fertig wurde. Derzeit hält ein einzelner Prozessor der Multialpha den Rekord mit 5 Sekunden, wenn man die C Variante verwendet. Mit Blick auf diese Zahlen erscheint die WPP1 zunehmend veraltet.

Einzelne Komponenten des Nachfolgemodells WPP2 wurden mit den kleineren Datensätzen getestet. Rechenzeiten konnten hier noch nicht ermittelt werden. Diese Maschine soll durch ihre SIMD-Architektur auch bei BBB-Produktionen eine lineare Performance aufweisen, für die die Partnermenge nicht assoziativ bestimmt werden kann. Darüberhinaus soll der 'Flaschenhals' zwischen Hostrechner und Spezialhardware beseitigt werden, so daß auch der lineare Faktor wesentlich herabgesetzt sein wird.

Die jeweils zehn Meßwerte jeder Zeile wurden durch Regressionsparabeln (Fehlerquadratsummenminimierung) angenähert. Es ergibt sich für jede Zeile ein konstanter, ein linearer und ein quadratischer Koeffizient. Die Tabelle D.5 zeigt diese Ergebnisse. Die Zahlen für die Beispiele in den 256^2 Bildern sind kaum verwertbar. Die Rechenzeiten sind zu kurz und die Anzahl der Instanzen zu klein. Die Parameter sind offensichtlich instabil. Betragsmäßig große negative Werte im konstanten Parameter deuten dagegen auf überquadratisches Wachstum hin. In diesen Fällen reicht wohl der Arbeitsspeicher nicht mehr, so daß zusätzlicher Aufwand bei der Verwaltung des Massenspeichers entsteht. Der hohe Wert im linearen Parameter für die WPP1-Spezialhardware erklärt sich aus einem 'Flaschenhals' in der Kommunikation mit dem Hostrechner. Im wesentlichen ist der Nachweiß erbracht, daß der quadratische Koeffizient hier zu vernachlässigen ist.

Anzahl der erzeugten *U_structure* Instanzen	40000	80000	160000	320000
Anzahl der akzeptierten *U_structure* Instanzen	1125	2314	4592	9021
Anzahl der reduzierten *E_structure* Instanzen	0	0	0	0
Anzahl der reduzierten *light_truck* Instanzen	0	0	0	0

Table D.6: **Zufällige Instanzen im Bereich** 4000^3 **Voxel**

Anzahl der erzeugten *U_structure* Instanzen	40000	80000	160000	320000
Anzahl der akzeptierten *U_structure* Instanzen	1101	2258	4417	8829
Anzahl der reduzierten *E_structure* Instanzen	0	0	3	4
Anzahl der reduzierten *light_truck* Instanzen	0	0	0	0

Table D.7: **Zufällige Instanzen im Bereich** 2000^3 **Voxel**

D.2 Versuche mit der 3D Beispiel KG

Für die 3D KG aus dem Abschnitt 6.1 wurde ein Programm erstellt, welches zufällige Instanzen des Symbols *U_structure* erzeugt. Angegeben wird dabei vom Benutzer ein 3D Intervall, in dem sich die Instanzen befinden sollen, das minimale und maximale Abmaß im Sinne des Prädikates *geom_const* (siehe Anhänge B und C) und die Anzahl der zu erzeugenden Instanzen. Es ist dafür Sorge getragen worden, daß sich alle vier Punkte in einer Ebene befinden, aber nicht auf einer Geraden. In dieser Ebene ist der Polygonzug U-förmig. Der erste und der letzte Punkt liegen also auf derselben Seite der Geraden durch den zweiten und dritten Punkt. Trotzdem wird, wie die Zahlen in den Tabellen D.6 bis D.9 zeigen, nur etwa jede vierzigste Instanz akzeptiert, erfüllt also alle Kriterien des Prädikates *geom_const*.

Die Argumentation aus dem Abschnitt 5.3 kann durch diese Versuche bestätigt werden: Die Anzahl der *E_structure* Instanzen wächst quadratisch mit der *Dichte* der *U_structure* Instanzen im Attributraum. Dies kann erreicht wer-

Anzahl der erzeugten *U_structure* Instanzen	40000	80000	160000	320000
Anzahl der akzeptierten *U_structure* Instanzen	1083	2130	4321	8760
Anzahl der reduzierten *E_structure* Instanzen	0	3	7	32
Anzahl der reduzierten *light_truck* Instanzen	0	0	0	0

Table D.8: **Zufällige Instanzen im Bereich** 1000^3 **Voxel**

Anzahl der erzeugten *U_structure* Instanzen	40000	80000	160000	320000
Anzahl der akzeptierten *U_structure* Instanzen	1040	2196	4592	9021
Anzahl der reduzierten *E_structure* Instanzen	2	12	47	227
Anzahl der reduzierten *light_truck* Instanzen	0	0	0	0

Table D.9: **Zufällige Instanzen im Bereich** 500^3 **Voxel I**

Anzahl der erzeugten *U_structure* Instanzen	40000	80000	160000	320000
Anzahl der akzeptierten *U_structure* Instanzen	12835	25141	50684	101243
Anzahl der reduzierten *E_structure* Instanzen	425	1639	6638	25165
Anzahl der reduzierten *light_truck* Instanzen	0	0	0	3

Table D.10: **Zufällige Instanzen im Bereich** 500^3 **Voxel II**

den, indem man die Anzahl der *U_structure* Instanzen erhöht und den Bereich, in dem sie sich befinden, konstant hält oder indem man das Volumen dieses Bereichs verkleinert und die Anzahl der *U_structure* Instanzen konstant hält.

Instanzen des Symbols *light_truck* konnten so nicht erzeugt werden. Deswegen wurde das Programm, welches die *U_structure* Instanzen mit dem Zufallsgenerator erzeugt, so modifiziert, daß der Teil des Prädikates *geom_const* erfüllt ist, der verlangt, daß die gegenüberliegenden Seiten der U-Struktur (als Viereck betrachtet) etwa gleich lang sind. Die Instanzen sind also in den Grenzen parallelogrammförmig, die auch die KG aus Kapitel 6 verlangt. Auf diese Weise wächst die Zahl der akzeptierten *U_structure* Instanzen um mehr als eine Größenordnung. Wenn dann etwa 100000 Instanzen auf einem Würfel von 5 Meter Kantenlänge zusammengedrängt sind, entstehen erste Instanzen des Startsymbols durch Zufall. Die Tabelle D.10 dokumentiert diese Versuche.

Den Attributbereich kann man nicht mehr wesentlich einschränken, wenn noch *light_truck* Instanzen reduziert werden sollen. Das Fahrzeug ist etwa 4,5 Meter lang. Man könnte aber die Anzahl der akzeptierten *U_structure* Instanzen noch einmal verdoppeln. In diesem Falle steht zu erwarten, daß etwa viermal so viele *E_structure* Instanzen reduziert werden und die Anzahl der *light_truck* Instanzen sprunghaft ansteigt.

Appendix E

Definition der Algorithmen Sprache für Mengen

Im Kapitel 3 wurden verschiedene Verfahren zur algorithmischen Behandlung von KGs diskutiert. Der dabei verwendete Code orientiert sich zum Teil an PASCAL, ist aber mit speziell für diese Aufgabe tauglichen Sonderkonstruktionen versehen. Daher wird hier die Syntax in Abschnitt E.1 und die Semantik in E.2 detailliert erläutert.

E.1 Syntax

Die Syntax wird nach der BNF-Konvention präzisiert. Dabei gibt es eine kleine Änderung: Die geschweiften Klammern { und } können nicht wie üblich als Teil der Metasprache verwendet werden, da sie in der Objektsprache auftreten. In ihrer Rolle werden die eckigen Klammern [und] in die Metasprache aufgenommen. Der Inhalt zwischen diesen Klammern darf also beliebig oft - auch nullmal - gesetzt werden.

$$
\begin{array}{rcl}
< executable > & ::= & < main\_head >< main\_body > \\
< main\_head > & ::= & ALGORITHM < symbol > (< para\_list >); \\
 & & VAR[< declaration >] \\
 & & [< subroutine >] \\
< main\_body > & ::= & Begin\ main[< action >]end.main \\
< para\_list > & ::= & [< symbol >:< type >;] \\
 & & [VAR < symbol >:< type >;] \\
< subroutine > & ::= & < rout\_head >< rout\_body >
\end{array}
$$

$$
\begin{array}{rcl}
< rout\_head > & ::= & PROCEDURE < symbol > (< para\_list >); \\
 & & VAR[< declaration >] \\
 & & [< subroutine >] \\
< rout\_body > & ::= & Begin\ [< action >] end < symbol >; \\
< declaration > & ::= & < symbol > [, < symbol >] :< type >;
\end{array}
$$

$$
\begin{array}{rcl}
< action > & ::= & \forall < symbol > \in < term > [< action >] end \forall < symbol >; | \\
 & & Choose < symbol > \in < term > | \\
 & & If < condition > Then [< action >;] \\
 & & Else[< action >;] endIf; | \\
 & & If < condition > Then[< action >;] endIf; | \\
 & & < symbol > (< symbol > [, < symbol >]); | \\
 & & < symbol >:< symbol > \longrightarrow < symbol >; | \\
 & & < symbol >:< symbol > \longmapsto < symbol >; | \\
 & & < symbol > := < term >; | stop; \\
< condition > & ::= & < condition > \vee < condition > | \\
 & & < condition > \wedge < condition > | \\
 & & < symbol > \in < term > | \\
 & & < symbol > \subseteq < term > | \\
 & & < term > = < term > | \\
 & & < term > \neq < term > | \\
< type > & ::= & Set\ of < KG\_Item >; | \\
 & & Pair\ of < KG\_Item >, < KG\_Item >; | \\
 & & Triple\ of < KG\_Item >, < KG\_Item >, < KG\_Item >; | \\
 & & \{< symbol > [, < symbol >]\}; | \\
 & & < KG\_Item >; \\
< term > & ::= & < term > \cup < term >; | \\
 & & < term > \cap < term >; | \\
 & & < term > \setminus < term >; | \\
 & & \{< symbol > \in < symbol >; < condition >\} | \\
 & & < symbol >; \\
< KG\_Item > & ::= & Universe \mid Production \mid Terminal \mid Nonterminal \mid \\
 & & Instance \mid Configuration;
\end{array}
$$

E.2 Semantik

Die Bedeutung der einzelnen nichtterminalen und terminalen Symbole und der Konstrukte wird im folgenden erläutert. Dabei ist vor allen Dingen zu klären, wie die Anweisungen zu verarbeiten sind. Darüberhinaus ergeben sich eine Reihe von semantischen Bedingungen (im Sinne von [KNUT]), die sinnvollerweise zusätzlich zu den syntaktischen Anforderungen abzuprüfen sind.

- Mit *executable* wird ein Algorithmus also ein Programm bezeichnet. Es gliedert sich in einen deklarativen Kopf *main_head* und einen ausführbaren Teil *main_body*.

- Ein Unterprogramm *subroutine* hat im wesentlichen die gleiche Struktur. Zur besseren Unterscheidung sind Kopf *rout_head* und ausführbarer Teil *rout_body* mit entsprechenden Schlüsselworten kenntlich gemacht.

- Mit *action* ist eine Anweisung gemeint. Die Struktur ist rekursiv. Komplexere Anweisungen setzen sich aus einfacheren zusammen. Dafür gibt es die üblichen Möglichkeiten für bedingte Anweisungen mit If und $Then$ und einem optionalen $Else$. Besonders wichtig sind hier aber die **Aufzählung von Mengen** mit dem Symbol $\forall$ und eine nicht näher spezifizierte **Auswahl aus einer Menge** mit der Funktion $Choose$. Natürlich ist ein Prozedur Aufruf möglich. Dabei wird eine Parameterliste übergeben. Die Parameterübergabe geschieht normalerweise durch den Wert. Wenn im Deklarationsteil ein VAR vor dem Parameter steht, dann wird per Referenz übergeben, und die Prozedur kann den entsprechenden Wert ändern. Die Prozedur muß vor ihrem Aufruf definiert sein. Rekursiver Selbstaufruf ist zulässig. Die Anzahl und die Typen der Parameter müssen mit der Parameterliste dort kompatibel sein. Durch das Zeichen := wird einem Symbol der Wert zugewiesen, der sich aus der Berechnung eines Terms ergibt. Darüberhinaus ist es auch möglich, mit den Symbolen $\longrightarrow$ und $\longmapsto$ KG-Produktionen durchzuführen. $\longrightarrow$ wird bei der Reduktion von Mengen verwendet und $\longmapsto$ bei der Anwendung von Produktionen auf Konfigurationen. Als spezielle Funktionen gelten $\mathcal{A}_p$ für die Menge der Ausgangskonfigurationen einer Produktion p und $\mathcal{B}(\kappa)$ für das Bild einer Konfiguration κ wie im Abschnitt 2.1 definiert.

- Mit *term* ist ein Term im Sinne der Mengenalgebra mit den Operationen $\cup, \cap, \setminus, ...$ gemeint. Ferner ist es erlaubt, Untermengen einer Menge zu bilden mit Hilfe von Bedingungsprädikaten. Als 'atomare' Termkonstruktionen gelten einfache Symbole.

- *symbol* steht für die Terminale. Hier sind beliebige (nicht leere) Zeichenketten möglich. Ausgenommen werden die Schlüsselworte $ALGORITHM$,

PROCEDURE, *VAR*, *Begin*, *end*, *main* usw.

Appendix F

Bildquellennachweis

- **Bildfolge BF1:** Diese Bildfolge wurde am 27.10.1988 um c. a. 14^{00} zwischen den Gemeinden Sulzbach und Malsch in Baden von E. Michaelsen aufgenommen. Das Objekt lag bei normalen Sichtbedingungen im direkten Sonnenlicht. Als Kamera diente eine Kleinbildspiegelreflex des Typs CONTAX mit einem $35mm$ Zeiss Distagon T Objektiv, für das ein Datenblatt vorliegt ([ZEISS]). Als Bildhauptpunkt wurde die Maskenmitte angenommen. Verwendet wurde KODAK Ektachrom 100 Diafilm mit Entwicklungsprozeß E6. Die Abtastung erfolgte durch einen OPTRONICS System c-4500 mit $40\frac{Pixel}{mm}$ mit einer radiometrischen Auflösung von einem BYTE (nach Mittelung über die drei Farbauszüge). Der verwendete Ausschnitt mißt 1400×700 Pixel. Die Bildfolge umfaßt 8 Aufnahmen. Zur Bestimmung einer ersten Näherung für die äußeren Kameraparameter wurde ein Stativ mit einer groben Winkelteilung und ein Maßband verwendet. In Abbildung 6.15 ist die Aufnahme der Bildfolge dokumentiert. Der Abstand der Bildaufnahmestandorte beträgt jeweils $2m$. Die Entfernung zum Objekt liegt bei etwa $30m$ bis $40m$. Im nachhinein wurde in die Bildfolge BF1 ein kleiner Baum aus einer anderen ähnlichen Bildfolge hineinkopiert. Dabei wurde von einer Position zwischen dem Fahrzeug und den Kamerastandorten ausgegangen, so daß das Fahrzeug in allen Bildern partiell verdeckt ist.

- **Bildfolge BF2:** Diese Bildfolge wurde am 25.05.1996 um c. a. 12^{00} bei Büchenau (Bruchsal) in Baden von E. Michaelsen aufgenommen. Die Sichtbedingungen waren bei bewölktem Himmel außergewöhnlich klar, die Beleuchtung diffus. Als Kamera diente eine Mittelformatsucherkamera des Typs MAMIA UNIVERSAL 6x9 mit einem passenden $150mm$ Objektiv vom selben Hersteller. Als Bildhauptpunkt wurde die Maskenmitte angenommen. Verwendet wurde KODAK Technical PAN mit Feinkornentwicklung. Die Abtastung erfolgte durch einen OPTRONICS Color Getter

3pro mit $40\frac{Pixel}{mm}$ mit einer radiometrischen Auflösung von einem BYTE (nach Mittelung über die drei Farbauszüge). Der verwendete Ausschnitt mißt 3200×1600 Pixel. Die Bildfolge umfaßt 8 Aufnahmen. Die Ausrichtung der Kamera ist jeweils etwa gleich. Die Aufnahmestandorte wurden jeweils 10 Schritte auseinander gewählt. Es wurde aus der Hand belichtet. Die Entfernung zum Objekt liegt bei etwa $200m$.

- **Bildfolge BF3:** Diese Bildfolge wurde am 01.02.1996 um c. a. 11^{00} bei Leimersheim in der Pfalz von E. Michaelsen und J. Hebel aufgenommen. Das Fahrzeug wurde freundlicherweise vom Pionierbrückenbatallion 330 der BW zur Verfügung gestellt. Die Sichtbedingungen waren eher dunstig, die Beleuchtung schwach und diffus. Als Kamera diente eine Mittelformatsucherkamera des Typs MAMIA UNIVERSAL 6x9 mit einem passenden $250mm$ Objektiv vom selben Hersteller. Als Bildhauptpunkt wurde die Maskenmitte angenommen. Verwendet wurde KODAK Technical PAN mit Feinkornentwicklung. Die Abtastung erfolgte durch einen OPTRONICS Color Getter 3pro mit $80\frac{Pixel}{mm}$ mit einer radiometrischen Auflösung von einem BYTE (nach Mittelung über die drei Farbauszüge). Der verwendete Ausschnitt mißt 6400×3200 Pixel. Die Bildfolge umfaßt 8 Aufnahmen. Zur Bestimmung einer ersten Näherung für die äußeren Kameraparameter wurde ein Stativ mit einer groben Winkelteilung und eine abgemessene Schnur verwendet. Der Abstand der Bildaufnahmestandorte beträgt jeweils $10m$. Die Entfernung zum Objekt liegt bei etwa $300m$ bis $400m$.

Index

Bibliography

[AHO] Aho A. V.; Ullman J. D.: *The Theory of Parsing, Translation, and Compiling*, Volume I: *Parsing*, Prentice Hall, Englewood Cliffs, N. J., 1972.

[ANDE] Anderer C. e. a.: *Sensordatenauswertung zur Zielbekämpfung*, FIM-Bericht Nr. 198 Teil 2, FIM, Ettlingen, 1989.

[ANDS-68-1] Anderson R. H.: *Syntax-Directed Recognition of Hand-Printed Two-Dimensional Mathematics*. In: Klerer M.; Reinfelds J.: *Interactive Systems for Experimental Applied Mathematics*, Academic Press, New York, 1968, pp 436 - 459.

[ANDS-68-2] Anderson R. H.: *Syntax-Directed Recognition of Hand-Printed Two-Dimensional Mathematics*, Ph. D. Thesis, Department of Engineering and Applied Physics, Havard University, Cambridge, Mass., Jan. 1968.

[ANDS-77] Anderson R. H.: *Two-Dimensional Mathematical Notation*. In: Fu K. S.: *Syntactic Pattern Recognition, Applications*, Springer, Berlin, 1977, pp 147 -177.

[BAIR] Baird H. S.; Bunke H.; Yamamoto K. (Eds.): *Structured Document Image Analysis*, Springer, Berlin, 1992.

[BAKER] Baker H. H.: *Scene Structure from a Moving Camera*. In: Blake A.; Troscianko T.: *Ai and the Eye*, John Wiley and Sons, Chichester, 1990, pp 229 - 260.

[BAU] Bausch U.; Kestner W.; Sties M.: *Vorstudie zur kontextbezogenen Interpretation von Bildern und Bildfolgen auf der Grundlage semantischer Netze*, FIM-Bericht Nr. 88, FIM, Karlsruhe, März 1981.

[BELL] Bellman R.: *Dynamic Programming*, Princeton University Press, Princeton, N. J., 1957.

[BENN] Bennett B. M.; Hoffman D. D.; Prakash C.: *Observer Mechanics*, Academic Press, New York, 1989.

[BINF] Binford T. O.; Levitt T. S.: *Model-based Recognition of Objects in Complex Scenes*. In: ARPA (Hrsg): *Image Understanding Workshop 1994*, Proceedings in zwei Bänden von einem Workshop in Montery, University of California, Morgan Kaufman, San Franzisco, 1994, 149 - 155;

[BRUN] Brunn A.; Gülch E.; Lang F.; Förstner W.: *A Multi-Layer Strategy for 3D Building Acquisition*. In: Leberl F.; Kalliany R.; Gruber M.: *Mapping Buildings, Roads and other Man-Made Structures from Images*, (Proceedings IAPR TC-7 Workshop, Graz, 1996), R. Oldenbourg, Wien, 1997, pp 11-37.

[BUN-85] Bunke H.: *Modellgesteuerte Bildanalyse*, Teubner, Stuttgart, 1985.

[BUN-89] Gmür E.; Bunke H.: *3-D Object Recognition Based on Subgraph Matching in Polynomial Time*. In: Mohr R.; Pavlidis Th.; Sanfeliu A.: *Structural Pattern Analysis*, World Scientific, Singapore, 1989, pp 131 - 147.

[BUN-94] Messmer B. T.; Bunke H.: *Erkennen und Lernen zweidimensionaler Objekte mittels Subgraph-Isomorphismus*. In: Kropatsch W. G.; Bischof H. (Hrsg): *Mustererkennung 1994*, (DAGM 94), Informatik Xpress 5, TU Wien, 1994.

[CHEL] Chellappa R.; Rosenfeld A.: *Vision Engineering: Designing Computer Vision Systems*. In: Chen C. H.; Pau L. F.; Wang P. S. P.: *Handbook of Pattern Recognition & Computer Vision*, World Scientific, Singapore, 1993, pp 805 - 815.

[CHEN] Chen C. H.; Pau L. F.; Wang P. S. P.: *Handbook of Pattern Recognition & Computer Vision*, World Scientific, Singapore, 1993.

[CHOM] Chomsky N.: *Aspects of the Theory of Syntax*, MIT Press, Cambridge, Mass., 1965.

[CLOW] Clowes M. B.: *Transformational Grammars and the Organization of Pictures*. In: Grasselli A. ed al. *Automatic Interpretation and Classification of Images* Academic Press, New York, 1969.

[COLL] Collin S.; Colnet D.: *Syntactic Analysis of Technical Drawing Dimensions*, Int. Jour. of Pat. Rec. and AI., Vol. 8, No. 5, 1994, pp 1131 - 1148.

[DENZ] Denzler j.; Niemann H.: *COBOLT: Ein System zur datengetriebenen Verfolgung bewegter Objekte in Echtzeit*. In: Paulus E.; Wahl F. M. (Hrsg): *Mustererkennung 1997*, (DAGM 97), Informatik aktuell, Springer, Berlin, 1997, pp 209-216.

[DICK] Dickmanns E. D.: *Performance Improvements for Autonomous Road Vehicles*. In: Rembold U.; Dillmann R.; Hertzberger L. O.; Kanade T. (Hrsg):

Intelligent Autonomous Systems, IAS-4, IOS Press, Amsterdam, 1995, pp 2 - 14.

[EARL] Early J.: *An Efficient Context-Free Parsing Algorithm*, Ph.D. Thesis, MIT, Cambridge, Massachusetts, 1968.

[ENGE] Engelmore R.; Morgan T.: *Blackboard Systems* Addison-Wesley, Wokingham England, 1988.

[FISC] Fischer A.; Steinhage V.: *On the Computation of Visual Events in Aspect Graph Generation*. In: Paulus E.; Wahl F. M. (Hrsg): *Mustererkennung 1997*, (DAGM 97), Informatik aktuell, Springer, Berlin, 1997, pp 156-163.

[FAUG-93-1] Faugeras O.: *Three-Dimensional Computer Vision*, MIT, Cambridge, Massachusetts, 1993.

[FAUG-93-2] Faugeras O.: *Computer Vision Research at INRIA*, Int. J. of Comp. Vis., Vol. 10, No. 2, 1993, pp 91-99.

[FEDE] Feder J.: *Plex Languages*, Inform. Sci. 3, 1971, pp 225 - 241.

[FINK] Fink B.: *Anwendung formaler Sprachen in der Bildverarbeitung*, FIM-Bericht Nr. 2, FIM, Karlsruhe, Februar 1973.

[FUE-85] Füger H.; Greif H. J.; Lütjen K.: *Wiedererkennen von stationären Objekten zur bildgestützten Navigation und zur integrierten Luftaufklärung*, FIM-Bericht Nr. 135 Teil 2, FIM, Ettlingen, Dezember 1985.

[FUE-87] Füger H.; Greif H.-J.; Jurkiewicz H. J.; Lütjen K.: *Auswahlverfahren für die wissensbasierte Bildauswertung mit dem Blackboard-basierten Produktionssystem BPI*. In: Paulus E. (Hrsg): *Mustererkennung 1987*, (DAGM 87), Informatik Fachberichte Nr.149, Springer, Berlin, 1987, pp 290-294.

[FUE-88] Füger H. et al.: *Wiedererkennen von stationären Objekten für die bildgestützte Navigation*, Abschlussbericht, FIM, Ettlingen, Dezember 1988.

[FUE-90-1] Füger H.; Lütjen K.; Jurkiewicz K.: *Kontextsensitive Bildanalyse in Luftbildern*. In: Großkopf R. (Hrsg): *Mustererkennung 1990*, (DAGM 90), Informatik Fachberichte Nr. 254, Springer, Berlin, 1990, pp 585-592.

[FUE-90-2] Füger H.; Lütjen K.; Michaelsen E.; Schwan K.: *Strukturorientierte 3D-Szenenanalyse in Bildfolgen*. In: Großkopf R. (Hrsg): *Mustererkennung 1990*, (DAGM 90), Informatik Fachberichte Nr. 254, Springer, Berlin, 1990, pp 659-666.

[FU-72] Lee H. C.; Fu K. S.: *A Stochastic Syntax Analysis Procedure and Its Application to Pattern Classification*, IEEE Trans. on Comp., Vol. C-21, No. 7, 1972, pp. 660-666.

[FU-73] Fu K. S.; Bahargava B. K.: *Tree Systems for Syntactic Pattern Recognition*, IEEE Trans. on Comp., Vol. C-22, No. 12, 1973, pp. 1087-1099.

[FU-74] Fu K. S.: *Syntactic Methods in Pattern Recognition*, Academic Press, New York, 1974.

[FU-77] Fu K. S.: *Syntactic Pattern Recognition, Applications*, Springer, Berlin, 1977.

[FU-82] Fu K. S.: *Syntactic Pattern Recognition and Applications*, Prentice Hall, Englewood Cliffs, N. J. 1982.

[FU-84] Lin W. C.; Fu K. S.: *A Syntactic Approach to 3-D Object Representation*, IEEE Trans. on Pat. An. and Mach. Int., Vol. 6, No. 3, 1984, pp. 351-364.

[GONZ] Gonzalez R. C.; Thomason M. G.: *Syntactic Pattern Recognition*, Addison-Wesley, Reading, Mass., 1978.

[GRAU] Grau O.: *Ein Szeneninterpretationssystem zur Modellierung dreidimensionaler Körper*. In: Sagerer G.; Posch S.; Kummert F. (Hrsg.): *Mustererkennung 1995*, (DAGM 95), Informatik aktuell, Springer, Berlin, 1995.

[GRIM] Grimson L.; Eric W.: *Object Recognition by Computer: The Role of Geometric Constraints*, MIT Press, Cambridge, Mass., 1990.

[GROC] Groch W. D. e. a.: *Automatisierung der Zielerkennung in Aufklährungsbilddaten des infraroten Spektralbereichs*, FIM-Bericht Nr. 196 Teil 1, FIM, Ettlingen, 1988.

[GRUB] Gruber M.; Kofler M.; Leberl F.: *Managing Large 3D Urban Database Contents supporting Phototexture and Levels of Detail*. In: Gruen A.; Baltsavias E. P.; Henricsson O.: *Automatic Extraction of Man-Made Objects from Aerial and Space Images (II)*, (Ascona Workshop der ETH), Birkhäuser Verlag, Basel, 1997, pp. 377-386.

[GRUE] Gruen A.; Kuebler O.; Agouris P.: *Automatic Extraction of Man-Made Objects from Aerial and Space Images*, (Ascona Workshop der ETH), Birkhäuser Verlag, Basel, 1995.

[HAAL] Haala N.; Hahn M.: *Data fusion for the dedection and reconstruction of buildings*. In: Gruen A.; Kuebler O.; Agouris P.: *Automatic Extraction of Man-Made Objects from Aerial and Space Images*, (Ascona Workshop der ETH), Birkhäuser Verlag, Basel, 1995, pp 211-220.

[HARA] Haralick R. M.; Shapiro L. G.: *Computer and Robot Vision*, in zwei Bänden, Addison-Wesley, Reading, Mass., 1992.

[HAYS] Hays D. G.: *Introduction to Computational Linguistics*, American Elsevier, New York, 1967.

[HENR] Henricsson O.: *Analysis of Image Structures using Color Attributes and Similarity Relations*, Diss., ETH Zürich, Inst. f. Geod. u. Photogrm., Zürich, 1996.

[HEUS] Heuser M.; Liedge C. E.: *Ein attributiertes Relaxationsverfahren zur 3D-Lageerkennung von Objekten.* In: Burkhard H.; Höhne K. H.; Neumann B. (Hrsg): *Mustererkennung 1989*, (DAGM 89), Informatik Fachberichte Nr. 219, Springer, Berlin, 1989.

[HORN] Hornegger J. M.: *Statistische Modellierung, Klassifikation und Lokalisation von Objekten*, Diss., Univ. Erlangen-Nürnberg, Technische Fakultät, Erlangen, 1996.

[IU-94] ARPA (Hrsg): *Image Understanding Workshop 1994*, Proceedings in zwei Bänden von einem Workshop in Montery, University of California, Morgan Kaufman, San Franzisco, 1994.

[KAPP] Kappenberger M.: *Eine Wissenserwerbskomponente zum Erlernen von Konzepten durch konstruktive Induktion*, Diss., Fortschrittberichte VDI, Reihe 10: Informatik/Kommunikationstechnik Nr. 146, VDI Verlag, Düsseldorf 1990.

[KASA-65] Kasami T.: *An Efficient Recognition and Syntax Analysis Algorithm for Context-Free Languages*, AFCRL-65-758, Air Force Cambridge Research Laboratory, Bedford, Massachusetts, 1965.

[KASA-69] Kasami T.; Torii K.: *A Syntax Analysis Procedure for Unambiguous Context-Free Grammars*, J. ACM. Vol. 16, No. 3, 1969, pp. 423-431.

[KIRS] Kirsch R.: *Computer Interpretation of English Text and Picture Patterns*, IEEE Trans. Elec. Comp. VOl EC.-13, 1964, pp. 363-367.

[KLUG] Kluge K.; Kanade T.; Hideki K.: *Car Recognition for the CMU Navlab.* In : Thorpe C. E. (Ed.): *Vision and Navigation - The Carnegie Mellon Navlap*, Kluwer Academic Publishers, Norwell Mass., 1990, pp 95-115.

[KNUT] Knuth D. E.: *Semantics on Context-Free Languages*, Math. Systems Theory, Vol 2, No 2, 1968, pp 127-145.

[KOLL] Kollnig H.; Nagel H.-H.: *3D Pose Estimation by Fitting Image Gradients Directly to Polyhedral Models.* In: *Fifth International Conference on Computer Vision*, ICCV'95, IEEE Computer Soc. Press, Los Alamitos, 1995, pp 569-574.

[KONE] Konecny G.; Lehmann G.: *Photogrammetrie*, Walter de Gruyter, Berlin, 1984.

[KORT] Kort A.; Pogoda A.; Steinhage V.: *Employing Aspect Hierarchies for Building Detection in Digital Images*. In: Jähne B.; Geißler P.; Haußecker H.; Herin F. (Hrsg): *Mustererkennung 1996*, (DAGM 96), Informatik aktuell, Springer, Berlin, 1996, pp. 217-224.

[KREB] Krebs B.; Korn B.; Burkhardt M.: *A 3d Object Recognition System with Decision Reasoning under Uncertainty*. In: Paulus E.; Wahl F. M. (Hrsg): *Mustererkennung 1997*, (DAGM 97), Informatik aktuell, Springer, Berlin, 1997, pp 183-190.

[LED-2] Lloyd E.: *Probability*. Aus: Ledermann W.: *Handbook of Applicable Mathematics*, Band II, John Wiley and Sons, Chichester, 1980.

[LESS] Lesser V. R.; Erman L. D.: *A Retrospective View of the Hearsay-II Architecture*, Fith International Joint Conference on Artificial Intelligence, Cambridge Mass., 1977.

[LUE-86-1] Lütjen K.: *Ein wissensbasierter Ansatz zur Wiedererkennung von symbolisch beschriebenen Objekten für die bildgestützte Navigation*, FIM-Bericht Nr. 157, FIM, Ettlingen, April 1986.

[LUE-86-2] Lütjen K.: *BPI: Ein Blackboard-basiertes Produktionssystem für die automatische Bildauswertung*. In: Hartmann G. (Hrsg.): *Mustererkennung 1986*, (DAGM 86), Informatik Fachberichte Nr.125, Springer, Berlin, 1986, pp 164-168.

[LOWE] Lowe D. G.: *Perceptual Organization and Visual Recognition*, Kluwer Academic Publishers, Norwell Mass., 1985.

[MAHN] Mahn U.: *Attributierte Grammatiken und Attributierungsalgorithmen*, Springer, Berlin, 1988.

[MAYH] Mayhew J. E. W.; Frisby J. P.: *Model Recognition from Stereoscopic Cues*, MIT Press, Cambridge, Mass., 1991.

[MARI] Marriott K.: *Constraint Multiset Grammars*, IEEE Symposium on Visual Languages, 1994, pp 118-125.

[MARR] Marr D.: *Vision*, W. H. Freeman and Company, San Franzisco, 1982.

[MICH-89] Michaelsen E.: *Objektidentifikation im extremen Tiefflug*, FIM-Bericht Nr. 203, FIM, Ettlingen, Dezember 1989.

[MICH-88] Michaelsen E.: *Neuronale Netze*, FIM-Bericht Nr. 204, FIM, Ettlingen, Dezember 1988.

[MICH-96] Michaelsen E.: *3D Coordinate Grammars*. In: Girod B.; Niemann H.; Seidel H.-P. (Eds.): *3D Image Analysis and Synthesis '96*, Infix, Sankt Augustin, 1996, pp 81-85.

[MILG] Milgram D. L.; Rosenfeld A.: *A note on 'grammars with coordinates'*. In: Nake F.; Rosenfeld A.: *Graphic Languages*, North Holland, Amsterdam, 1972, pp 187 - 194.

[MINS] Minsky M. L.: *Berechnung: Endliche und Unendliche Maschinen*, Verlag Berliner Union, Stuttgart, 1971.

[NAGE] Nagel H.-H.: *AI Approaches towards Sensor-Based Driver Support in Road Vehicles*. In: Nebel B.; Dreschler-Fischer L. (Eds): *KI-94: Advances in Artificial Intelligence*, Springer, Berlin, 1994, pp 1-15;

[NAKA-89] Nakamura A.; Aizawa K.: *Relationships Between Coordinate Grammars and Path Controlled Graph Grammars*, Int. J. of Pat. Rec. and A. I., Vol. 3, 1989, pp 445 - 458.

[NAKA-95] Nakamura A.: *Some Notes on Parallel Coordinate Grammars*, Int. J. of Pat. Rec. and A. I., Vol. 9, 1995, pp 753 - 761.

[NARA-64] Narasimhan R.: *Labeling Schemata and Syntactic Description of Pictures*, Inform. Contr., vol. 7, 1964, pp. 151-179.

[NARA-69] Narasimhan R.: *Picture Languages*. In: Kaneff S. (Edtr.): *Picture Language Machines*, Conference in Canberra 1969, Academic Press, London, 1970, pp 1 - 30.

[NIEM-83] Niemann H.: *Klassifikation von Mustern*, Springer, Berlin, 1983.

[NIEM-90] Niemann H.: *Pattern Analysis and Understanding*, Springer, Berlin, 1990.

[NILS] Nilsson N. J.: *Problem Solving Methods in Artificial Intelligence*, McGraw-Hill, New Jork, 1971.

[PAVL] Pavel M.: *Fundamentals of Pattern Recognition*, Marcel Dekker Inc., New York, 1993.

[PERN] Perner P.; Wang P.; Rosenfeld A. (Edtr.): *Advances in Structural and Syntactical Pattern Recognition* (IAPR/SSPR 96), Lecture Notes in Computer Sience 1121, Springer, Berlin, 1996.

[POPP] Popp C.: *Eine Erklaerungskomponente für ein Blackboard-System*, Diss., Fortschrittberichte VDI, Reihe 10: Informatik/Kommunikationstechnik Nr. 135, VDI Verlag, Düsseldorf 1990.

[QUIN] Quint F.: *Zur automatischen, kartengestützten Auswertung monokularer Luftbilder*, Diss., Univ. Karlsruhe, Inst. f. Photogrmtr. u. Fernerk., Karlsruhe, 1997.

[ROSE-71] Rosenfeld A.: *Isotonic Grammars, Parallel Grammars an Picture Grammars*. In: Meltzer B.; Michie D.: *Machine Intelligence 6*, pp 281 - 294, Edinburgh Univ. Press, 1971.

[ROSE-79] Rosenfeld A.: *Picture Languages*, Academic Press, New York, 1979.

[ROSE-89-1] Rosenfeld A.: *Coordinate Grammars Revisited*, Int. J. of Pat. Rec. and A. I., Vol. 3, 1989, pp 435 - 444.

[ROSE-89-2] Rosenfeld A.: *An Architecture for Picture Parsing*. In: Narasimhan R.: *A Perspective in Theoretical Computer Science*, World Scientific, Singapore, 1989.

[ROSE-90] Rosenfeld A.: *Array, Tree and Graph Grammars*. In: Bunke H.; Sanfeliu A.: *Syntactic and Structural Pattern Recognition Theory and Applications*, World Scientific, Singapore, 1990.

[SAGR] Sagerer G.: *Darstellung und Nutzung von Expertenwissen für die Bildanalyse*, Informatik Fachberichte Nr.104, Springer, Berlin, 1985.

[SCHL] Schlageter G.; Stucky W.: *Datenbanksysteme: Konzepte und Modelle*, Teubner, Stuttgart, 1983.

[SCHÖ] Schöning U.: *Theoretische Informatik - kurzgefaßt*, Spektrum, Heidelberg, 1997.

[SEST] Sester M.; Förstner W.: *Object Location Based on Uncertain Models*. In: Burkhardt H.; Höhme K. H.; Neumann B. (Hrsg.): *Mustererkennung 1989*, (DAGM 89), Informatik Fachberichte Nr. 219, Springer, Berlin, 1989, pp 457-464.

[SHAW-68] Shaw A. C.: *The Formal Description and Parsing of Pictures*, Report SLAC-84, Stanford Linear Accelerator Center, Stanford University, Stanford, California, 1968.

[SHAW-69] Shaw A. C.: *A Formal Picture Description Scheme as a Basis for Picture Processing Systems*, Information and Control, Vol 14, 1969, pp 9 - 52.

[STEI] Füger H.; Stein G.; Stilla U.: *Multi-populations evolution strategies for structural image analysis*, IEEE Conference on Evolutionary Computation (ICEC'94), Orlando, Vol I, 1994, pp 229-234.

[STIL-91] Stilla U.; Jurkiewicz K.: *Objektklassifikation mit einem blackboardorientierten Inferenzmechanismus*, FIM-Bericht Nr. 230, FIM, Ettlingen, Juni 1991.

[STIL-95-1] Stilla U.: *Map-aided Structural Analysis of Aerial Images*. ISPRS Journal of Photogrammetry and Remote Sensing, Vol 50, 1995, pp 3-10.

[STIL-95-2] Stilla U.; Michaelsen E.; Lütjen K.: *Structural 3D-Analysis of Aerial Images with a Blackboard-based Production System*. In: Gruen A.; Kuebler O.; Agouris P.: *Automatic Extraction of Man-Made Objects from Aerial and Space Images*, (Ascona Workshop der ETH), Birkhäuser Verlag, Basel, 1995, pp 53-62.

[STIL-96] Stilla U.; Michaelsen E.; Lütjen K.: *Automatic Extraction of Buildings from Aerial Images*. In: Leberl F.; Kalliany R.; Gruber M.: *Mapping Buildings, Roads and other Man-Made Structures from Images*, (Proceedings IAPR TC-7 Workshop, Graz, 1996), R. Oldenbourg, Wien, 1997, pp 229-244.

[STIL-97] Stilla U.; Michaelsen E.: *Semantic Modeling of Man-Made Objects by Production Nets*. In: Gruen A.; Baltsavias E. P.; Henricsson O.: *Automatic Extraction of Man-Made Objects from Aerial and Space Images II*, (Ascona Workshop der ETH), Birkhäuser Verlag, Basel, 1997, 43-52.

[STRA] Strat T. M.; Climenson D.: *An Overview of DARPA's Research Program in Automatic Population of Geospatial Databases*. In: Gruen A.; Baltsavias E. P.; Henricsson O.: *Automatic Extraction of Man-Made Objects from Aerial and Space Images II*, (Ascona Workshop der ETH), Birkhäuser Verlag, Basel, 1997, 3-12.

[THOR] Thorpe C. E. (Ed.): *Vision and Navigation - The Carnegie Mellon Navlap*, Kluwer Academic Publishers, Norwell Mass., 1990.

[TOMB] Tombre K.: *Structural and Syntactic Methodes in Line Drawing Analysis: To Which Extent Do They Work*. In: Perner P.; Wang P.; Rosenfeld A. (Edtr.): *Advances in Structural and Syntactical Pattern Recognition* (IAPR/SSPR 96), Lecture Notes in Computer Sience 1121, Springer, Berlin, 1996 pp 310-321.

[TSAI] Tsai W. H.; Fu K. S.: *Attributed Grammar - a Tool for Combining Syntactic and Statistical Approaches to Pattern Recognition*, IEEE Trans. SMC-10, 1980, pp 873-885.

[ULLM] Ullman S.: *High-level Vision*, MIT Press, Cambridge, Mass., 1995.

[WERT] Wertheimer H.: *Untersuchungen zur Lehre von Gestalt*, Psychologische Forschung, Vol 1, pp 47-58, 1922.

[WETZ] Wetzel D.: *Wissensbasierte Verkehrsszenenanalyse zur Fahrerunterstützung*, Diss., Infix, Sankt Augustin, 1996.

[WINK] Winkler G.: *Image Analysis, Random Fields and Dynamic Monte Carlo Methods*, Reihe *Applications of Mathematics* Nr. 27, Springer, Berlin, 1995.

[WINS] Winston P. H.: *Artificial Intelligence*, Addison-Wesley, Reading Mass., 1984.

[YNGR] Younger D. H.: *Recognition and Parsing of Context-Free Languages in time* n^3, Information and Control Vol. 10, No. 2, 1967, pp. 189-208.

[ZADE] Zadeh L. A.: *Syllogistic Reasoning in Fuzzy Logic and its Application to Reasoning with Dispositions*, UCB/ERL M84/17, University of California, Berkeley, 1984.

[ZEISS] Carl Zeiss Stiftung Oberkochen, Abteilung Photographie: *Datenblatt für Distagon T f2.8 35mm*, Oberkochen, 1985.