Maschinelle Erkennung von handgeschriebenen, mathematischen Ausdrücken

Diplomarbeit

Alexander Schulz-Heyn

Betreuer:

Prof. Dr. Alex Waibel
Dipl.Inform. Stefan Manke

Bearbeitungszeitraum: 1.Januar-30.Juni 1997

Institut für Logik, Komplexität und Deduktionssysteme
Universität Karlsruhe

angefertigt an den

Interactive Systems Laboratories (CS)
Carnegie Mellon University, Pittsburgh, Pennsylvania U.S.A.

Zusammenfassung

Man ist gewohnt mathematische Ausdrücke bestehend aus Integralen, Brüchen, Exponenten, Indizes, etc. mit einem Stift zu schreiben. Diese Form der Eingabe erlaubt es, mathematische Sonderzeichen und deren zwei-dimensionale Beziehungen auf einer Schreibfläche zu notieren. Heutige Computereingabesysteme für mathematische Ausdrücke benutzen weniger intuitive Eingabegeräte, wie Tastatur und Maus, aus Mangel eines maschinellen Erkennungssystems, welches automatisch und fehlerfrei Stiftbewegungen in eine eindeutige Repräsentation konvertieren kann. In dieser Arbeit wird ein auf einem neuen Verfahren basierendes Erkennungssystem vorgestellt, welches in der Lage ist, Stiftbewegungen als mathematische Ausdrücke zu interpretieren.

Mit diesem System wird dem Benutzer eine vertraute Eingabemodalität mittels Stift und drucksensitiven Bildschirm für mathematische Ausdrücke zum Zweck der Weiterverarbeitung in Anwendungen der Textdarstellung oder der Computeralgebra bzw. eines Taschenrechners angeboten. Die zu niedrige Erkennungsrate des Systems verhindert zwar eine vollständige Ablösung der Tastatur/Maus, die heute als Haupteingabegeräte gelten, jedoch zeigt der im Rahmen dieser Arbeit entwickelte Prototyp, daß eine intuitive und schnelle Eingabe von mathematischen Ausdrücken möglich ist.

Das Problem der maschinellen Erkennung von mathematischen Ausdrücken fällt in die Problemklasse der Erkennung von handgeschriebenen Texten. Zusätzlich jedoch sollten die geometrischen Verhältnisse der Wortgruppen berücksichtigt werden und ein Sprachmodell auf Eingaben mathematischer Ausdrücke abgestimmt werden. Das in dieser Diplomarbeit behandelte Problem der maschinellen Handschriftenerkennung für mathematische Ausdrücke wird mittels eines Parsers für probabilistische, attributierte Grammatiken gelöst, wobei die Attributsregeln aus lernfähigen Methoden bestehen. Zur Ermittlung der Erkennungsleistung des Gesamtsystems wurde ein Datensammelprojekt zur Gewinnung von Test- und Trainingsdaten durchgeführt und ein auf dieser Methode basierendes Programm implementiert.

Hiermit erkläre ich, daß ich diese Diplomarbeit selbständig verfaßt und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Karlsruhe, 30.6.1997

Alexander Schulz-Heyn

Danksagung

Ich möchte allen Personen danken, die mir bei der Fertigstellung dieser Diplomarbeit mit Ideen und Vorschlägen geholfen haben. Dank geht an meine Betreuer Prof. Dr. Alex Waibel und Dipl. Inform. Stefan Manke. Große Hilfe in programmtechnischen und konzeptionellen Fragen erhielt ich von Matthias Denecke, Marsal Gavalda und Ralph Groß. Mit allen hatte ich viele nützliche Diskussionen über Grammatiken, Parser, Datenbanken und Erkennungsalgorithmen.

Einen besonderen Dank geht an all diejenigen Personen, die für wenig Entgelt handschriftliche Daten spendeten und mir so wertvolles Material zur Verfügung stellten.

Meine Eltern ermöglichten mir das Studium der Informatik. Ihnen möchte ich diese Abschlußarbeit widmen.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einleitung

1.1 Einführung

Um vom Menschen intuitiv verwendete Eingabemethoden in Benutzerschnittstellen zu integrieren, müssen die von dem Eingabegerät (wie z.B. Mikrofon, Kamera, Stift/drucksensitiver Bildschirm, etc.) gelieferten Primärdaten (z.B. Sprachsignal, Bildfolgen, Stiftbewegungen, etc.) zunächst interpretiert werden. Diese Aufgabe versuchen Erkennungssysteme zu lösen. Allerdings werden oftmals unkorrekte Interpretationen berechnet, die den gewonnenen Vorteil der intuitiven Eingabemodalität gegenüber Eingabgeräten, die eindeutige Daten produzieren, aber weniger bedienungsfreundlich sind, abschwächen, da der Benutzer diese Fehlinterpretationen berichtigen muß.

Der Grund schwacher Erkennungsraten hängt oftmals damit zusammen, daß das zugrundeliegende Problem komplizierter ist, als das es der Algorithmus modellieren kann und daß die Rechenleistung und Speicherkapazität heutiger Rechner der Komplexität des Problems nicht gewachsen sind. Diese Umstände zwingen den Algorithmus einige, eventuell korrekte Interpretationsmöglichkeiten der Eingabe nicht zu berücksichtigen. Die Bestrebungen einer Weiterentwicklung von Erkennungssystemen zielen in die Verringerung der Fehlinterpretationen. Um diesem Ziel näher zu kommen, werden oft Systeme verwendet, die in der Lage sind, eine Abbildung von Eingabe zur gewünschten Ausgabe selbständig zu erlernen. Diese Lernmethoden haben sich in den Einsatzgebieten bewährt, bei denen eine Aufstellung eines Regelwerks per Hand aus Mangel an Verständnis oder aufgrund der Komplexität des Problems nicht realisierbar ist.

Am Beispiel der Eingabe von mathematischen Ausdrücken wird ein ähnliches Problem sichtbar. Auch hier hat der Benutzer mit der Tastatur die Möglichkeit einen mathematischen Ausdruck exakt zu beschreiben, jedoch erzielt ein Computerstift mit einem drucksensitiven Bildschirm durch seine intuitive Eingabeart Bedienungsvorteile. Der Nachteil liegt wiederum darin, daß ein Erkennungssytem erforderlich ist, welches die Stifttrajektorie in eine eindeutige Repräsentation überführt.

Mathematische Ausdrücke werden aus Einzelzeichen zusammengesetzt, welche in Positions- und Größenverhältnissen zueinander stehen. Das Problem der Konvertierung von einer Stifttrajektorie in eine eindeutige Repräsentation des mathematischen Ausdrucks liegt darin, zunächst eine Segmentierung der Eingabe zu vollziehen, bevor die daraus ergebenen Segmente als Zeichen interpretiert und ihre geometrischen Relationen bestimmt werden können. Dieses Problem soll mittels eines maschinellen Erkennungssystems für mathematische Ausdrücke gelöst werden, welches in dieser Arbeit vorgestellt

wird. Dieses Erkennungssystem verbindet einen bedienungsfreundlichen Editor, welcher auf der Eingabe per Computerstift und drucksensitiven Bildschirm basiert, mit Anwendungsprogrammen, die mathematische Ausdrücke weiterverarbeiten.

1.2 Eingabeformen und Einsatzgebiete

Die Tastatur als Haupteingabegerät eignet sich weniger gut zur intuitiven Eingabe von mathematischen Ausdrücken als z.B. ein Computerstift mit drucksensitiven Bildschirm. Die Ursache dafür ist, daß mathematische Ausdrücke im allgemeinen zwei-dimensionale Strukturen haben, die mit vielen Sonderzeichen versehen sind, welche auf einer herkömmlichen Tastatur nicht vorhanden sind. Um den mathematischen Ausdruck mit der Tastatur einzugeben, ist der Benutzer gezwungen, umständliche Befehlsfolgen einzugeben.

Neben der Eingabe per Tastatur oder Computerstift gibt es weitere Eingabemodalitäten, welche im folgenden kurz beschrieben und ihre spezifischen Vor- und Nachteile genannt werden.

Eingabe per Tastatur

Mittels der Tastatur kann eine formale Sprache zur Beschreibung von mathematischen Ausdrücken, wie z.B. HTML [Hus97] oder Latex [Kop93], eingegeben werden. Ein spezielles Sonderzeichen, für welches es keine Taste auf der Tastatur gibt, wird durch ein Befehlswort beschrieben. Strukturelle Zusammenhänge werden ebenfalls mit Befehlen eingeleitet. Diese Befehle definieren auch die geometrischen Relationen der Teilausdrücke. Ein Beispiel für ein HTML Ausdruck sieht wie folgt aus:

$$\texttt{\∑_{n = 0}^mxⁿ}$$

Dieser Text produziert folgende Darstellung auf dem Monitor:

$$\sum_{n=0}^{m} x^n$$

Vorteile: Der Quelltext ist mit einem einfachen Texteditor lesbar und editierbar.

Nachteile: Diese Modalität verlangt spezielle Kenntnisse der formalen Sprache vom Benutzer. Außerdem bedarf es eines Kommandoaufrufs zur Visualisierung des Ausdrucks.

Eingabe per Tastatur und Computermaus

Eine Computermaus ermöglicht eine Berarbeitung des mathematischen Ausdruckes am Darstellungsmonitor. Das sogenannte "Point and Click"-Verfahren bietet dem Benutzer eine Schreibfläche und einen gesonderten Bereich zum Anklicken von Befehlen, eine sogenannte "Toolbox". Mittels der Computermaus bestimmt der Bediener die Stelle der Eingabe, indem er auf der Schreibfläche eine ihm gewünschte Stelle anklickt. Danach

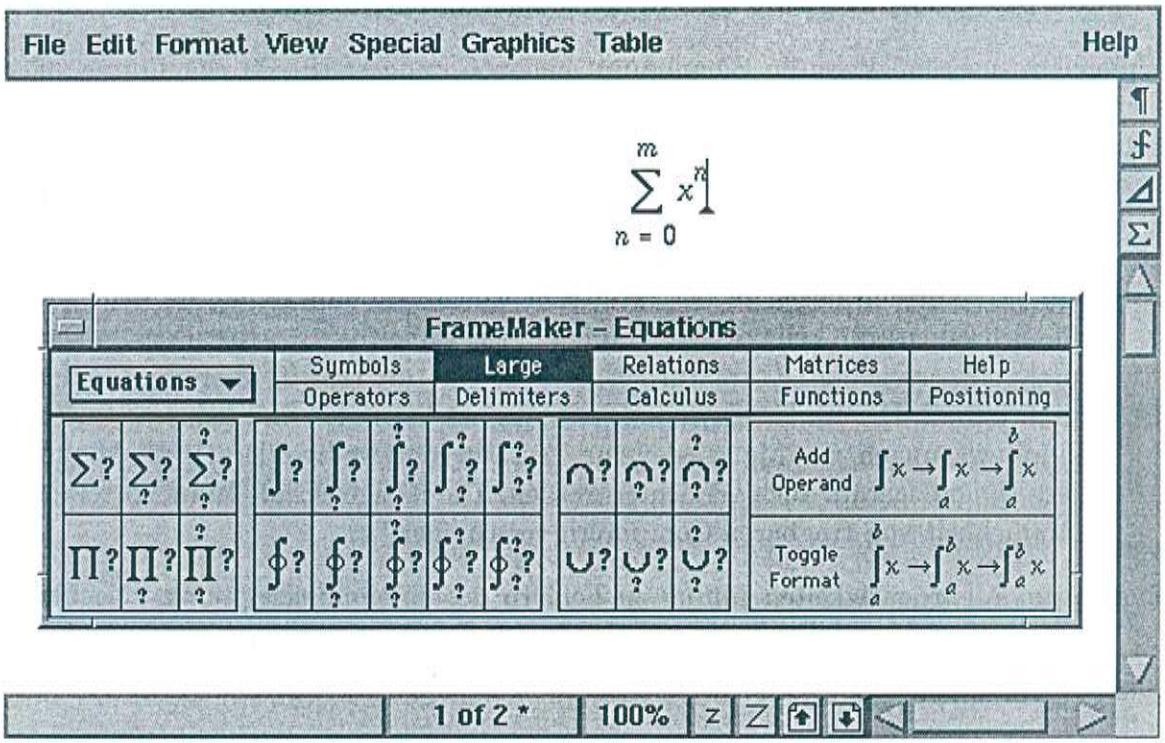


Abbildung 1.1: "Point and Click"- Verfahren im Beispiel

sucht er einen gewünschten Befehl der Toolbox aus. Dieses Verfahren wird wiederholt, bis der Ausdruck erstellt ist. Das Bild 1.1 zeigt einen Editor (in diesem Fall Frame Maker [Bra94]), der diese Methode verwendet.

Vorteile: Das Eingabeverfahren ist anschaulich und schnell erlernbar.

Nachteile: Je größer die Anzahl der Befehle und Sonderzeichen, welche vom System unterstützt werden sollen, desto unübersichtlicher wird die Toolbox.

Eingabe per Mikrofon

Eine Spracheingabe für mathematische Ausdrücke ist ebenfalls denkbar. Dabei spricht der Benutzer in ein mit dem Erkennungssystem verbundendes Mikrophon.

Vorteile: Die Eingabe ist äußerst schnell vom Benutzer ausführbar, falls der Spracherkenner echtzeitfähig ist.

Nachteile: Undeutliches Sprechen führt zu Fehlinterpretationen des Erkennungssystems. Eine Korrektur ist schwer möglich, da eine Positionierung der Korrekturstelle innerhalb eines Ausdrucks Probleme darstellt. Außerdem lassen sich Klammerungen umständlich verbal beschreiben. Z.B. ist $a^b * c^d$ mit a^{b*c^d} phonetisch leicht verwechselbar.

Eingabe per Computerstift und drucksensitivem Bildschirm

Schließlich ist es möglich, die Eingabe mit einem Computerstift durchzuführen. Dabei zeichnet der Benutzer die Zeichen auf eine drucksensitive Schreibfläche. Die Einzelzeichen und geometrischen Relationen werden mit entsprechenden Erkennungssystemen ermittelt. Es sind keine weiteren Eingaben zur Beschreibung des Ausdrucks notwendig. Ein Beipiel für eine derartige Eingabe ist in Abbildung 5.2 zusehen.

Vorteile: Die Vorteile liegen in der intuitiven Eingabemodalität, da sie dem gewohnten Schreiben mit einem Stift auf Papier ähnelt. Korrekturen lassen sich durch entsprechende Korrekturbewegungen mit dem Stift über der Fehlerstelle durchführen. Die Einlernzeit zur Bedienung ist kurz. Es sind keine zeitintensiven Handbewegungen zur Maus oder zur Tastatur notwendig. Deswegen können auf diese Eingabegeräte auch verzichtet werden. Diese Eigenschaft kann Vorteile in der Handlichkeit von tragbaren Computern verschaffen.

Nachteile: Mehrdeutigkeiten führen zu Fehlern des Erkennungssystems. Nachkorrekturen aufgrund von Fehlinterpretationen des Erkennungssystems verlangsamen die Eingabe und ermüden den Benutzer.

Eine Kombination unterschiedlicher Eingabearten ist besonders effektiv, da die speziellen Eingabemodalitäten jeweils dort eingesetzt werden könnten, bei denen ihre Stärken liegen. So läßt sich z. B. zunächst per Mikrofon die Formel eingeben und mit Stift oder Maus Korrekturen ausführen. Feinabstimmungen sind letztlich durch Editieren eines Ausschnitts des erzeugten HTML- oder Latex-Textes denkbar.

Einsatzgebiete

In vielen Anwendungsbeispielen bei denen mathematische Ausdrücke zum Einsatz kommen, werden auch ihre eindeutigen Rechnerrepräsentationen zur Weiterverabeitung benötigt. So ist es z.B. zur Darstellung in einem Textprogramm nützlich, den strukturellen Aufbau eines mathematischen Ausdruckes zu kennen, um bei wechselnden Darstellungsumgebungen eine passende Gestaltung des Ausdrucks zu erzielen.

Außerdem werden mathematische Ausdrücke in Gebieten der mathematischen Analyse mit Computeralgebrasystemen benötigt. Hier ist die Kenntnis des strukturellen Aufbaus zwingend. Maple [Nic96] oder Matlab [Inc95] sind Beispiele eines maschinellen, mathematischen Assistenten. Taschenrechnerprogramm fallen ebenfalls in diese Anwendungsklasse. Desweiteren ist ein Einsatz im Gebiet der Eingabe für Computerhochsprachen denkbar, bei dem der Algorithmus mit einer Menge von mathematischen Formeln beschrieben wird. Die Spezifikationssprache Z [Hei95] ist ein Beispiel eines derartigen Ansatzes.

1.3 Problemstellung

Das Problem der Erkennung von mathematischen Ausdrücken ist ein Teilgebiet der maschinellen Erkennung handgeschriebener Texte [TSW90], bei denen zunächst Segmentgrenzen der Worte und dann die Worte selbst erkannt werden müssen. Bei der Erkennung

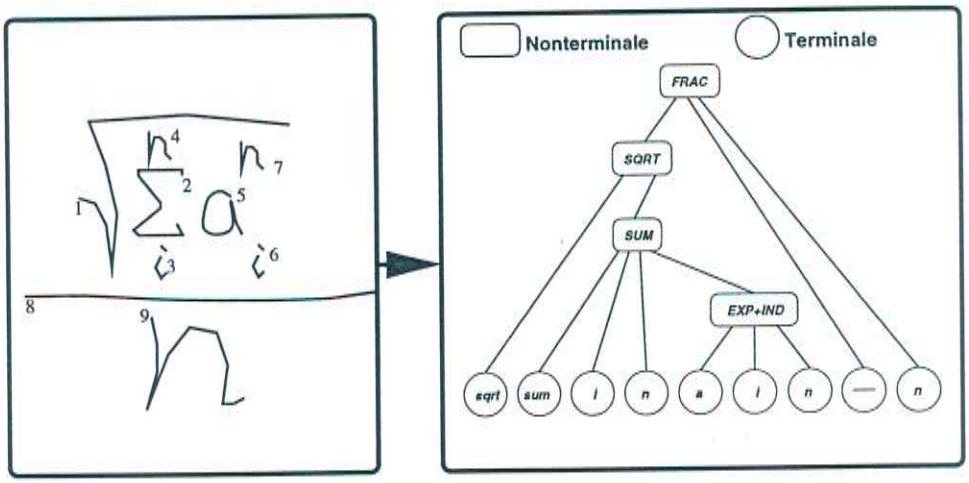


Abbildung 1.2: Konvertierung einer Stifttrajektorie eines mathematischen Ausdrucks in einen Syntaxbaum. Die Zahlen neben den Zeichen entsprechen der Eingabereihenfolge. Im rechten Teil der Graphik repräsentieren Rechtecke Verknüpfungstypen und Kreise Eingabezeichen.

von mathematischen Ausdrücken kommt hinzu, daß die naturgemäß zwei-dimensionale Struktur des Ausdrucks erfaßt und ein Sprachmodell auf mathematische Ausdrücke abgestimmt werden muß.

Mathematische Ausdrücke müssen in vielen Fällen einer Weiterverabeitung eindeutig im Rechner repräsentiert werden. Dazu eignet sich die Darstellung als Syntaxbaum, der die Struktur des Ausdruckes wiederspiegelt. In dieser Baumstruktur repräsentieren die Knoten einen Verknüpfungstyp der Nachfolger des Knotens, die wiederum mathematische Subausdrücke sind. Die Blätter entsprechen mathematischen Zeichen. Eine Repräsentation ist in Abbildung 1.2 veranschaulicht. Wohlgeformte mathematische Ausdrücke erlauben nur eine einzige syntaktische Interpretation. Deshalb ist es möglich, jedem wohlgeformten Ausdruck einen eindeutigen Syntaxbaum zuzuordnen und umgekehrt.

Ein Syntaxbaum läßt sich wiederum mit Hilfe der Prä-, In- oder Postfix-Notation in eine lineare Form umwandeln. Der mathematische Ausdruck wird in dieser linearen Form im Rechner repräsentiert. Abwandlungen dieser Notation kommen in bekannten formalen Sprachen, wie HTML [Hus97] oder Latex [Kop93], zur Anwendung.

Anlehnend an die zwei-dimensionale Struktur handgeschriebener, mathematischer Ausdrücke, bestehend aus einzelnen Zeichen, welche in geometrischen Relationen zueinander stehen, kann das Problem der Erkennung und Konvertierung in eine eindeutige lineare Notation in drei Teile untergliedert werden. Um die Einzelzeichen, aus denen der mathematische Ausdruck zusammengesetzt ist, zu erkennen, benötigt das Erkennungssystem ein entsprechendendes Modul zur Klassifikation von Teilsegmenten der Stifttrajektorie in Einzelbuchstaben. Außerdem ist ein Erkennungssystem zur Bestimmung der geometrischen Relationen dieser Einzelzeichen bzw. Subkomponenten des mathematischen Ausdrucks erforderlich. Das dritte und übergeordnete Modul bildet der Parser. Dieser Parser überprüft, ob die Stifteingabe so segmentiert werden kann, daß die daraus erkennbaren Zeichen und deren geometrischen Relationen einen Syntaxbaum ergeben,

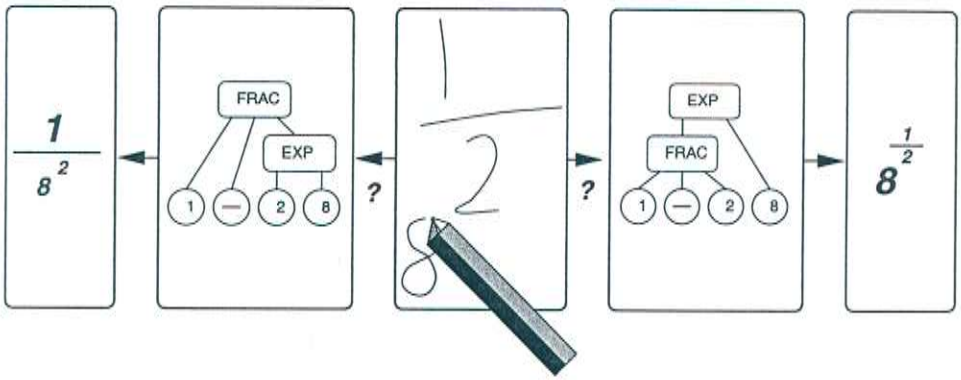


Abbildung 1.3: Beispiel einer mehrdeutigen Eingabe

welcher in seinem Aufbau einer Grammatik für mathematische Ausdrücke folgt. In Fällen, bei der die Eingabe undeutlich ist, muß das System in der Lage sein, mehrere Lösungen anzubieten. Das Bild 1.3 zeigt ein mehrdeutiges Beispiel. Je nach Auslegung der Länge des Bruchstriches kann entweder $8^{\frac{1}{2}}$ oder $\frac{1}{8^2}$ abgeleitet werden.

Jede syntaktisch korrekte Segmentierung sollte bewertet werden, damit bei Erhalt mehrerer Lösungen Vorzüge besserer Lösungsergebnisse gemacht werden können. Die Bewertung ergibt sich aus den Konfidenzmaßen der Erkenner für Einzelzeichen und der geometrischen Relationen, welche sich bzgl. der verwendeten Segmentierung ergaben. Bei Einsatz einer probabilistischen Grammatik können zusätzlich die Wahrscheinlichkeiten der angewendeten Produktion in die Bewertung einfließen. Da eine Bewertung aller Segmentierungsmöglichkeiten zu rechenaufwendig wäre, sollte der Parser ein entsprechendes Pruning-Verfahren anwenden, um den Aufbau des Syntaxbaumes abzubrechen, falls die Zwischenbewertung einen kritischen Schwellwert unterschreitet.

Die Problemfelder der maschinellen Erkennung handgeschriebener, mathematischer Ausdrücke sind in der folgenden Aufstellung überblickend aufgelistet.

Eingabe, Vorverarbeitung: Die Eingabe dient zur Aufzeichnung der Stifttrajektorie und wird mittels eines Computerstifts und drucksensitiven Bildschirms realisiert. Die Vorverarbeitung konvertiert die Stifttrajektorie in eine vom Parser gewünschte Datenstruktur.

Einzelzeichenerkenner: Der Erkennungsalgorithmus für Einzelzeichen bestimmt die Buchstaben innerhalb eines mathematischen Ausdrucks.

Erkenner für geometrische Relationen: Das Problem der Bestimmung von Positions- und Größenverhältnissen der Buchstaben und Subkomponenten zueinander, welche in mathematischen Ausdrücken eine wichtige Rolle einnehmen, muß von einem entsprechenden Erkenner für geometrische Relationen gelöst werden.

Parser: Der Parser hat die Aufgabe, die von der Vorverarbeitung stammende Stifttrajektorie in eine Folge von Teilsegmenten zu untergliedern und diese dem Einzelzeichenerkenner zu präsentieren. Die von den Ergebnissen des Einzelzeichenerkenners erzeugte Buchstabenkette muß mittels der Grammatik mathematischer Ausdrücke

syntaktisch überprüft werden. Dazu müssen auch die geometrischen Relationen der Einzelzeichen und Subkomponenten zueinander kontrolliert werden. Das Resultat bildet eine Lösungsmenge aus eindeutigen Syntaxbäumen unterschiedlicher Zeichen- und Relationskonstellationen.

Ausgabe: Die Aufgabe der Ausgabe beschränkt sich in der Umwandlung der vom Parser ermittelten Syntaxbäume in eine lineare Prä-, In- oder Postfix-Notation. Zusätzlich müssen die erkannten Ausdrücke lesbar dargestellt werden.

1.4 Beitrag dieser Arbeit

Diese Diplomarbeit stellt einen Erkennungsalgorithmus für mathematische Ausdrücke vor, der in der Lage ist, Stifttrajektorien als mathematische Ausdrücke zu interpretieren und die möglichen Hypothesen in Form einer Präfixform auszugeben. Mit einem derartigen Algorithmus ist es möglich, einen bedienungsfreundlichen Editor für mathematische Ausdrücke zu entwickeln, der die intuitive Eingabemodalität Stift mit drucksensitiven Bildschirm unterstützt. Die Erkennungsleistung erstreckt sich auf die Bestimmung der Einzelzeichen im Ausdruck und der geometrischen Relationen der Subkomponenten. Eine Grammatik muß bestimmt werden, welche die Sprache der mathematischen Ausdrucke beschreibt.

Die Diplomarbeit löst das Problem der maschinellen Erkennung von handgeschriebenen, mathematischen Ausdrücken mit einem neuen Verfahren, welches einen Parser für probabilistische, attributierte Grammatiken verwendet. Die Attributsregeln sind Erkennungsverfahren für Einzelzeichen und geometrische Relationen von Subkomponenten mathematischer Ausdrücke.

1.5 Gliederung

Zunächst werden im folgenden Kapitel existierende Erkennungssysteme für Einzelzeichen, geometrische Relationen und Systeme zur Erkennung von vollständigen, mathematischen Ausdrücken beschrieben. Im Kapitel 3 "Grundlagen" werden theoretische Grundlagen und Methoden angesprochen, die im System Verwendung finden.

Nachfolgend wird im Kapitel 4 das Datensammelprojekt vorgestellt, welches vorausgehend zur Entwicklung des Systems durchgeführt wurde, um Trainings- und Testdaten hinzuzugewinnen. Aus diesen Beispieldaten wurden dann einige heuristische Erkenntnisse entnommen, die Einfluß auf die Methoden des Erkennungssystems hatten.

Im Hauptkapitel 5 werden alle Methoden des Erkennungssystems und deren Veränderungen zu den im Kapitel "Grundlagen" eingeführten Standardverfahren erklärt. Im Kapitel 6 werden die Erkennungsergebnisse des Verfahrens aufgeführt. Eine Zusammenfassung und ein Ausblick wird im letzten Kapitel 7 gegeben, in dem auch auf mögliche weiterführende Arbeiten hingewiesen wird.

Im Anhang A dieser Arbeit wird das verwendete Alphabet des Einzelzeichenerkenners aufgelistet und einen Ausschnitt aus den Beispielvorlagen des Datensammelprojekts präsentiert. Außerdem wird im Anhang B auf das Unipen-Format eingegangen, welches

ein Format zur Beschreibung von Stifttrajektorien ist. Hier wird auch eine Erweiterung des Formats zur Beschreibung von mathematischen Ausdrücken vorgeschlagen.

Kapitel 2

Einordnung existierender Systeme

2.1 Einzelzeichenerkenner im Überblick

Mathematische Ausdrücke bestehen aus lateinischen und griechischen Buchstaben, Zahlen und mathematischen Sonderzeichen. Um diese Symbole in der Eingabe eines Ausdrucks zu detektieren, benötigt man ein entsprechendes Erkennungsystem für Einzelzeichen.

Ein Einzelzeichenerkenner liefert Erkennungshypothesen von Buchstaben eines gegebenen Alphabets aus einer Stifteingabe. Während bei der optischen Zeichenerkennung (Optical Charater Recognition, OCR) nur das Grauwertbild der Stifteingabe existiert, also nur die Information, ob der Stift bestimmte xy Positionen überstrichen hat, besitzt man bei der On-Line-Handschriftenerkennung zusätzlich die zeitliche Information der Stiftposition. Da zu jedem Zeitpunkt der Stift nur an *einer* bestimmten Stelle sein kann, ergibt sich eine chronologische Ordnung, eine Sequenz von (x, y, t) Vektoren.

Zusätzlich kann ein Einzelzeichenerkenner dahingehend eingeordnet werden, ob der zugrundeliegende Algorithmus lediglich einen bestimmten Schreibstil von einem bestimmten Schreiber akzeptiert, ob er schreiberunabhängig oder schreiberadaptiv ist, d.h ob der Algorithmus sich automatisch nach wenigen geschriebenen Buchstaben oder Wörtern auf den neuen Schreiber einzustellen vermag. Eine Vielfalt von Techniken, wie z.B. Ansätze basierend auf Neuronalen Netzen (NN), statistischen Methoden wie Hidden Markov Models (HMM) [WL90], Mustererkennung [DH73] usw. können angewendet werden, um ein adaptives Verhalten zu erlangen.

Ein OCR-Verfahren hat den Vorteil, daß die primäre Datenaufzeichnung nicht unbedingt am Computer geschehen muß, sondern auch auf Papier getätigt werden kann, das dann eingescannt wird. Ein klassisches Beispiel dieser Konstellation ist bei der Erkennung von Postleitzahlen auf Briefen für die maschinelle Weiterleitung des Postgutes gegeben [Fon92].

Es hat sich allerdings herausgestellt, daß heutige Erkennungssysteme, die mit On-Line-Daten arbeiten, bessere Erkennungsleistungen erzielen [BBNN93]. Ein Grund dafür liegt in der Möglichkeit aus On-Line-Daten ein Grauwertbild zu konvertieren, während umgekehrt eine Generierung von On-Line-Daten aus einem Grauwertbild in den meisten Fällen nicht eindeutig ist [Doe91]. Die zusätzliche temporäre Information erlaubt, einen effektiveren Erkennungsalgorithmus zu entwickeln, da sich die Eingabedimension von einem zwei-dimensionalen Grauwertbild zu einem eindimensionalen Positionsvektor reduziert und eine Segmentierung der Eingabe in Einzelzeichen robuster ist. Außerdem

bietet ein Erkennungssytem auf On-Line-Daten potentielle Echtzeitfähigkeit, da nicht auf das Ende der Eingabe gewartet werden muß, um mit der Berechnung zu beginnen. Diese meist inkrementellen Erkennunsalgorithmen werden "Run-On"-Verfahren genannt.

Ein schreiberadaptives Erkennungssystem auf On-Line-Daten wurde von Dean Rubine vorgestellt, das er im Rahmen seiner Doktorarbeit mit dem Thema "Gestikerkennung" an der Carnegie Mellon Universität implementierte [Rub91]. Dieses System basiert auf einer parametrischen Methode zur Schätzung der Wahrscheinlickeitsverteilung im Merkmalsraum. Zunächst werden aus der Trajektorie 12 Merkmale berechnet und mit Hilfe der trainierten Mittelwerte μ_c und Kovarianzmatrizen Σ_c die Wahrscheinlichkeit mit der Formel

$$p_c(x) = \frac{1}{(2\pi)^{d/2}|\Sigma_c|^{1/2}} e^{-\frac{1}{2}(x-\mu_c)^T \Sigma_c^{-1}(x-\mu)} \tag{2.1}$$

für jede Buchstabenklasse c berechnet. Die Klasse mit der höchsten Wahrscheinlichkeit wird als Ergebnis der Erkennung ausgegeben. Die Varianzen und Mittelwerte der Gaußverteilungen werden iterativ aus Trainingsbeispielen gebildet. Der Nachteil dieses Verfahrens, liegt in der unflexiblen parametrischen Funktion, die nicht in der Lage ist, mehrere "Gaußglocken" für eine Klasse zu vereinigen und somit versagt, das Problem aufzulösen, das dem XOR Problem in der Theorie der Neuronalen Netze bekannt ist. Der Vorteil liegt in der Geschwindigkeit, mit der das System die Gewichte (Parameter) trainiert bzw. ein Erkennungsergebnis berechnet. Der Berechnungsaufwand ist proportional zur Länge der Eingabesequenz. Bei einer Evaluation auf 200 Testbeispielen von einem Schreiber aus 26 Großbuchstabenklassen wurde eine Erkennungsrate von 94% gemessen.

Ein Erkennungssystem aus der Klasse der Off-line-Verfahren mit Verwendung eines neuronalen Netzes ist das System von Der-Shung Yang [Yan92]. Ein System, das zum Erkennen von Architektursymbolen entwickelt worden ist. Es ist schreiberunabhängig und benötigt keine On-Line-Daten zur Erkennung. Das System bietet vor allem in den Fällen Vorteile, bei denen die Trajektorien zu einem Symbol stark zwischen Benutzern variieren und somit schlechte Klassifikationsmerkmale liefern. Das System ist robust gegenüber Korrekturen und anderen der normalen Trajektorie abweichende Stiftführung. Das System basiert auf einem neuronalen Klassifikationsnetz, welches mit zwei versteckten Schichten ausgestattet ist (siehe Bild 2.1). Der Berechnungsaufwand für das Ergebnis der Klassifikation ist zur Größe der Schreibfläche proportional. Bei einer Menge von 3000 Testdaten von unterschiedlichen Spendern erreichte das System auf ca. 100 Architektursymbolen eine Erkennungsleistung von 93% .

Der Einzelzeichenerkenner NPen

Das NPen System fällt in die Klasse der schreiberunabhängigen Einzelzeichenerkennungssysteme und ist robust gegenüber unterschiedlichen Schreibstilen. Das System wurde an der Universität Karlsruhe entwickelt. Es basiert auf einem Multi-state Time Delay Neural Network (MS-TDNN). Eine Beschreibung des NPen-Systems kann in [MFW95a] oder [MFW95b] nachgelesen werden.

Beim NPen System wird zunächst eine Merkmalsextraktion auf der höhen- und breitennormalisierten Stifttrajektorie vorgenommen. Dabei wird zu jedem Zeitschritt ein Merkmalsvektor berechnet. Aus diesen Vektoren ensteht eine Vektorenfolge, welche die

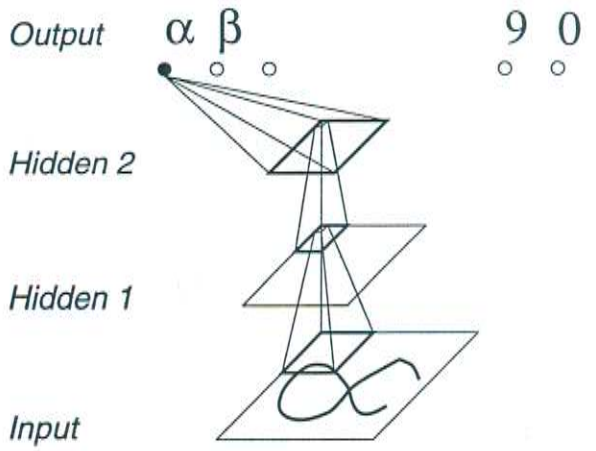


Abbildung 2.1: OCR-Algorithmus mit Einsatz eines Neuronalen Netzes mit zwei versteckten Schichten

Eingabe für das TDNN [WHG+89] bildet. Ein TDNN ähnelt einem herkömmlichen, neuronalen Klassifikationsnetz, welches aus mehreren Schichten besteht, die "übereinander" angeordnet sind und zwischen denen Verbindungen existieren, mit denen eine Anzahl von Eingabewerten der unteren Schichten gewichtet summiert an die darüberliegende Schicht weitergeleitet werden. In der Ausgabeschicht entspricht die Zahl der Zellen der Zahl der Klassen. Das TDNN des NPen Systems besitzt drei versteckte Schichten und eine Ausgabeschicht, die ihrerseits wieder aus drei Zellen für jede Buchstabenklasse besteht. Diese Ausgabezellen repräsentieren die Buchstabenpositionshypothesen, welche anzeigen, ob sich der Stift bzgl. der Stelle der Stifttrajektorie am Anfang, im Mittelteil oder im Endbereich eines Buchstabens befindet. Diese Vektorenfolge der Hypothesen wird wie folgt berechnet:

Ein TDNN schiebt ein Fenster über die Eingabedaten, bestehend aus den Merkmalsvektoren der Stifttrajektorie. Die Koeffizienten, die in diesem Fenster liegen, stellen die Eingabe an einer Spalte der ersten versteckten Schicht dar. Im nächsten Schritt wird das Fenster weitergeschoben und die nächsten "sichtbaren" Koeffizienten werden nach oben zur versteckten Schicht geleitet. Somit werden nacheinanderfolgend alle Spalten der ersten versteckten Schicht berechnet. Das Verfahren der Weiterleitung der Koeffizienten wird in den oberen Schichten wiederholt bis die Zellen der Ausgabeschicht ermittelt sind.

Nachdem die Bewertung der Buchstabenpositionshypothesen berechnet worden sind, wird mit dem Viterbialgorithmus [Rya93] der beste Pfad durch die vom Buchstabenmodell vorgegebenen Zustände gesucht. Für alle Buchstaben des Alphabets werden entsprechende Modelle definiert, welche aus Transitions- und Emmisionswahrscheinlichkeiten bestehen. Das Gütekriterium für ein Pfad ist die Multiplikation der Hypothesenbewertung entlang des Pfades mit Betrachtung der verwendeten Transitionen. Der Buchstabe mit dem höchsten Gütekriterium wird als Ergebnis der Erkennung angesehen. Das Bild in 2.2 zeigt einen schematischen Überblick des NPen Systems.

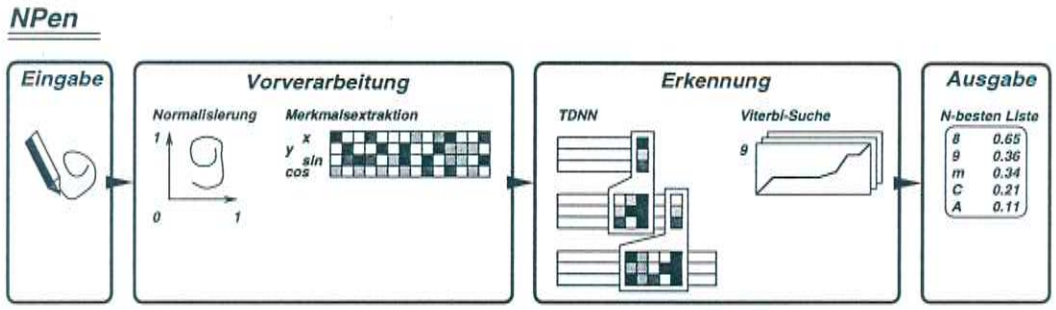


Abbildung 2.2: Das NPen Erkennungssystem mit vier Komponenten: Eingabe, Vorverarbeitung, Erkennung und Ausgabe der N-besten Liste

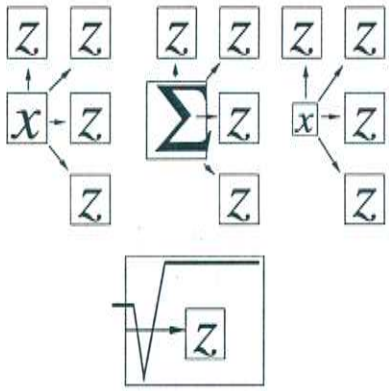


Abbildung 2.3: Geometrische Relationen in einem mathematischen Ausdruck

2.2 Erkenner geometrischer Relationen im Überblick

Die zwei-dimensionale Gestalt eines mathematischen Ausdruckes ergibt sich aus unterschiedlichen geometrischen Anordnungen seiner Subkomponenten. Diese geometrischen Relationen müssen vom Erkennungssystems ermittelt werden. Während die vorangegangenen Teilprobleme aus Lösungen von bereits bekannten Lösungsmethoden stammen, ist die Erkennung von geometrischen Relationen zweier mathematischer Subausdrücke ein spezifisches Problem der Erkennung mathemetischer Ausdrücke.

Gegeben sind zwei Mengen von Strokes. Strokes sind Teilabschnitte der Stifttrajektorie, die von einem Stiftabsetzen und Stiftabheben eingegrenzt sind. Die Aufgabe ist es, herauszufinden, in welcher geometrischen Relation diese Mengen zueinander stehen.

Es gibt bei mathematischen Ausdrücken mindestens 13 verschiedene Möglichkeiten, zwei Mengen von Komponenten anzuordnen. Zum einen kann die zweite Komponente in vier Richtungen von der ersten Komponente ausgehen (N, NO, O, SO) zum anderen kann das Größenverhältnis der Komponente sich in drei verschiedene Arten unterscheiden (identisch, kleiner, größer). Zusätzlich kann die zweite Komponente innerhalb der anderen stehen. Bild 2.3 zeigt diese geometrischen Relationen.

Das Problem der Symbolgruppierung wurde in anderen Arbeiten in unterschiedlicher Weise gelöst. Die meisten Ansätze basieren auf den Werten des jeweiligen einrahmenden

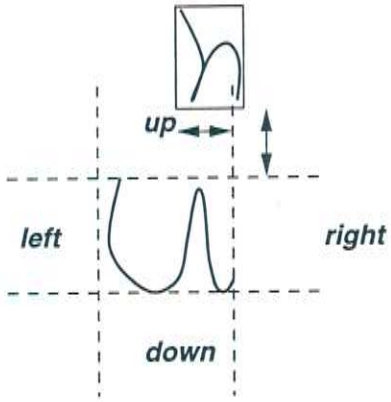


Abbildung 2.4: Regionenunterteilung durch Trennlinien nach der Symbolgruppierung von Winkler

Rechtecks, welche die Strokemenge umspannt. Das heißt, es werden jeweils die Minimum- und Maximumwerte der Trajektorien der Komponenten berechnet. Die Position der Grund- bzw. Mittellinie bzgl. des Rechteckes werden zusätzlich zur Bestimmung der geometrischen Relation herangezogen.

Hans-Jürgen Winkler [Win95b] schlägt einen Ansatz zur Lösung des Problems vor, bei dem der Zugehörigkeitsgrad der zweiten Komponente zu einer von vier Regionen berechnet wird. Diese Regionen ("up", "down", "left", "right") werden durch vier Trennlinien bestimmt, welche die erste Komponente umschließen (siehe Bild 2.4). Liegt nun die zweite Box vollständig in nur einer Region, wird keine Mehrdeutigkeit detektiert und das Ergebnis wird entsprechend ausgegeben. Sind Überlappungen vorhanden, werden Abstände der Box der zweiten Komponente zu den Trennlinien der Regionen berechnet und entsprechend die Zugehörigkeitsgrade zu den Klassen "up", "down", "left", "right" ermittelt. Aus diesen Zugehörigkeitsgraden können dann die geometrischen Relationen abgeleitet werden.

2.3 Erkenner handgeschriebener, mathematischer Ausdrücke im Überblick

Das Forschungsfeld der maschinellen Erkennung von Buchstaben, Worten und Texten von einer gedruckten oder handschriftlich erstellten Vorlage begann Anfang der 60er Jahre. Ein Überblick dieser Forschungsbereiche kann dem Artikel von C.C. Tappert [TSW90] entnommen werden.

In dem jungen Forschungsbereich der Erkennung mathematischer Ausdrücke hat sich noch kein klares Einordnungsschema von Erkennern ergeben. Folgende Merkmale eines Erkennungssystems für mathematische Ausdrücke könnten für eine Klassifizierung bedacht werden: Es ist nicht umbedingt notwendig, daß alle drei Aufgaben (Einzelzeichenerkennung, Erkennung geometrischer Relationen und Parser) vom Rechner übernommen werden. So ist es denkbar (und in einigen Systemen realisiert), daß z.B. nur der Einzelzeichenerkenner als lernfähiges System eingesetzt wird, während die Segmentierung und

Erkennung geometrischer Relationen durch die Tastatur geschehen muß. In der Diplomarbeit von Dirk Hopf [Hop96] konnte auf ein Erkennungsalgorithmus für geometrische Relationen zweier Komponenten verzichtet werden, da ein Editor dem Benutzer mehrere Eingabekästen zur Verfügung stellt, die jeweils zu einer Relation zur Hauptkomponente korrespondieren. Dem Benutzer war es nur erlaubt, in einer dieser Kästen zu schreiben. Die Einzelzeichen wurden mittels eines OCR Systems bestimmt. Durch die Verwendung von Eingabekästen konnte auch auf eine automatische Segmentgrenzendetektion einzelner Zeichen verzichtet werden.

Die Erkennungssysteme für Einzelzeichen und geometrische Relationen fallen ihrerseits in ein Klassifizierungsschema (siehe 2.1 und 2.2). So arbeitet z.B. das System von Zi-Xiong Wang [Wan88] mit einem Grauwertbild als Eingabendaten, die von einem gescannten Bild eines mathematischen Ausdrucks stammt.

Falls die Eingabe per Stift und drucksensitivem Bildschirm getätigt werden, könnten Einordnungkriterien hinzukommen, die sich auf die Eingaberestriktion für den Benutzer beziehen. So kann z.B. unterschieden werden, ob der Benutzer sich an eine bestimmte Reihenfolge beim Aufzeichnen von mathematischen Strukturen halten muß (wie z.B. bei einem Divisionsausdruck: 1) Bruchstrich, 2) Zähler, 3) Nenner) oder ob die Erkennung reihenfolgeninvariant ist. Ist eine Korrektur von Subkomponenten erlaubt? Muß zwischen zwei Zeichen ein Stiftabheben folgen? Wieviele Strokes darf ein Zeichen höchstens besitzen?

Die Hierarchie der Grammatik, welche der Parser benutzt, kann als weiteres Unterscheidungsmerkmal dienen. Ist die Grammatik in der Lage, einen syntaktisch unkorrekten Ausdruck zu erzeugen, so toleriert das System damit entsprechend falsche oder unvollständige Ausdrücke. Wenn die Grammatik kontext-sensitiv ist, so kann sie auch beliebige Matrizentypen generieren (siehe 3.1) und entsprechend akzeptiert das Erkennungssystem derartige Eingaben.

Mit den Arbeiten von H.-J. Winkler und M. Koschinski 1995 [Win95a],[Kos95] wurde ein System vorgestellt, das neben der Strukturierung zusätzlich eine Segmentierung und Erkennung von handgeschriebenen, mathematischen Zeichen realisiert. Die Primärdaten bestanden aus Stifttrajektorien, also On-Line-Daten. Die Restriktionen in der Eingabe eines Ausdrucks waren folgende:

- Die Reihenfolge von Subkomponenten dürfen von der Standardreihenfolge nicht abweichen, da der Parser nur eine lineare Grammatik parsen kann.
- Jedes Einzelzeichen darf höchstens aus vier Strokes bestehen. Diese Einschränkung wurde zugunsten einer kürzeren Berechnungszeit gewählt.
- Korrekturen und Rückspringen zu anderen Teilausdrücken während einer Aufzeichnung sind nicht erlaubt.
- Zwischen zwei Einzelzeichen befindet sich ein Aufheben des Stiftes.
- Es werden nur syntaktisch korrekte Ausdrücke akzeptiert.

Das System dieser Diplomarbeit ist dem Erkennungssystem von H.-J. Winkler ähnlich. Allerdings wurde ein Parser verwendet, der anstelle einer linearen eine kontext-freie Grammatik verwendet und so Reihenfolgenvertauschungen in der Eingabe von Subkomponenten

zulassen kann. Während das System von Winkler ein OCR-System benutzt, basiert das Einzelzeichensystem dieser Arbeit auf ein On-Line-Handschriftenerkenner. Ähnlichkeiten sind in der Art der Bestimmung von geometrischen Relationen zu finden. Beide Methoden geben feste Verknüpfungsregel auf unscharfe Merkmalswerte vor.

Kapitel 3

Grundlagen

3.1 Grammatiken und Sprachen

Mathematische Ausdrücke unterliegen in ihrem Aufbau einer gewissen Regelmäßigkeit der Verknüpfung der Komponenten und Einzelzeichen. Diese Regeln lassen sich in Form einer Grammatik beschreiben, wie sie in der Theorie der formalen Sprachen [MAK88] definiert werden.

Definition einer Grammatik: Eine Grammatik läßt sich als ein Quadrupel (T, N, S, P) notieren. Dabei steht T für eine Menge von Terminalen, N für eine Menge von Nichtterminalen, $S \in N$ ist das Startzeichen und P ist die Menge der Produktionen der Grammatik. Die Produktionen sind in der allgemeinen Form durch eine Menge von Paaren (v, w) von Zeichenketten erklärt. Die Zeichenkette $v \in (T \cup N)^+$ besteht aus einer gemischten Folge von Terminalen und Nonterminalen, besitzt aber mindestens ein Element. Die Zeichenkette w ist ein beliebiges Element aus der Menge $(X \cup V)^*$ einschließlich der leeren Zeichenkette, welche als λ notiert wird.

Mit der Grammatik lassen sich nun eine Menge von Zeichenketten generieren. Beginnend mit dem Startzeichen S sind mehrfach beliebige Produktionen anwendbar, bei denen v mit einem Teil der Zeichenkette übereinstimmt. Dieser Teil wird durch w ersetzt. Alle generierten Zeichenketten, die lediglich aus Terminalen bestehen, sind Elemente der Sprache $L(G)$ bzgl. der Grammatik. Die Sequenz der verwendeten Produktionen bestimmt auch den Syntaxbaum der Eingabe.

Definition der Sprache: $L(G) := \left\{u | u \in T^* \text{ und } S \stackrel{*}{\rightarrow} u\right\}$

wobei G eine Grammatik ist, T die Menge der Terminale und u eine Zeichenkette bestehend aus Terminalen, die vom Startsymbol abgeleitet werden kann.

Falls für eine Eingabe mehrere Syntaxbäume generiert werden können, nennt man die Grammatik ambigue. Wie bereits in der Einleitung 1.3 erörtert, können jedoch Eingaben von mathematischen Ausdrücken mit einer nicht-ambiguen Grammatik erzeugt werden, d.h. zu jeder Eingabe existiert genau eine Folge von Produktionen und demnach genau ein Syntaxbaum.

Normalformen für kontext-freie Grammatiken:

Kontext-freie Grammatiken können sich in den Produktionen unterscheiden, ohne jedoch die Menge der zu generierenden Zeichenketten zu verändern. Ein einfaches Beispiel

dafür ist mit Hilfe der Sprache $\{a^n | n > 0\}$ zu zeigen, die jeweils mit der Grammatik $(\{a\}, \{\}, S, \{S \rightarrow Sa|a\})$ oder $(\{a\}, \{\}, S, \{S \rightarrow aS|a\})$ erzeugt werden kann. Produktion von Grammatiken, die gewissen Strukturen unterliegen, nennt man Normalformen. Eine wichtige Normalform für den Cocke Kasami Younger Algorithmus, der im System zur Anwendung kommt und im Abschnitt 3.3 erklärt wird, ist die Chomsky-Normalform.

Definition der Chomsky-Normalform: Eine kontext-freie Grammatik $G = (T, N, S, P)$ ist in Chomsky-Normalform, wenn alle Produktionen einer der folgenden Formen annehmen:

1. $S \rightarrow \lambda$,
2. $A \rightarrow BC$,
3. $A \rightarrow a$,

wobei S das Startsymbol ist, $A \in N$, $a \in T$ und entweder $B, C \in N - \{S\}$, falls $S \rightarrow \lambda \in P$, andernfalls $B, C \in N$. Diese Fallunterscheidung für B bzw. C ist notwendig, um zu gewährleisten, das mindestens zwei Zeichen beim Parsen konsumiert werden, falls eine Produktion der 2. Art angewendet wird.

Man kann zeigen, daß jede kontext-freie Grammatik in eine Chomsky-Normalform umgewandelt werden kann [MAK88]. Diese Umformung ist mit Aufwand m ausführbar, wobei m die Länge der längsten rechten Seite einer Produktion ist (auch im weiteren als Länge einer Produktion bezeichnet).

Chomsky-Hierarchien

Man kann eine Grammatik nun damit klassifizieren, welche Klasse von Sprachen mit ihr generiert werden kann. In diesem Abschnitt soll ein kurzer Einblick in die wichtigsten Klassen von Sprachen gegeben werden. Die verwendete Einordnung folgt dem Hierachie-Theorem von N. Chomsky [Cho56]. Das Therorem besagt, daß es eine strikte Klassenhierarchie gibt mit der folgenden Eigenschaft:

$$L_3 \preceq L_2 \preceq L_1 \preceq L_0, \tag{3.1}$$

wobei

- L_0 die Klasse der freien Sprachen,
- L_1 die Klasse der kontext-sensitiven Sprachen ,
- L_2 die der kontext-freien und
- L_3 schließlich der linearen Sprachen ist.

Definition der Hierachien 0-3 nach Chomsky:

Hierarchie 0: Bei freien Grammatiken, sind alle Produktionen erlaubt, bei denen die linke Seite aus der Menge T^+ und die rechte Seite aus der Menge T^* entnommen wurde. Man stellt fest, das in freien Grammatiken keine Nonterminale benötigt werden.

Hierarchie 1: Eine Grammatik ist kontext-sensitiv, falls jede Produktion die Form $xAy \xrightarrow{*} xuy$ besitzt, wobei $x, y \in T^*, A \in N, u \in (T \cup N)^+$.

Hierarchie 2: Eine Grammatik ist kontext-frei, falls alle Produktionen die Form $A \rightarrow u, A \in V$ und $u \in (T \cup N)^*$ besitzen.

Hierarchie 3: Eine Grammatik ist rechts linear, falls jede Produktion die Form $A \rightarrow a, A \in N$ und $a \in T \cup \{\lambda\}$ oder die Form $A \rightarrow aB, A, B \in N$ und $a \in T$.

Einordung der Grammatik von mathematischen Ausdrücken

Im diesem Abschnitt soll gezeigt werden, daß die Sprache der mathematischen Ausdrücke nicht kontext-frei ist. Das Ergebnis hat Einfluß auf die Auswahl eines Parsers von mathematischen Ausdrücken.

Zum Beweis der Zugehörigkeit bzw. Nicht-Zugehörigkeit einer Sprache zu der Klasse kontext-freier Sprachen kann das Pumping Lemma (auch als *uvwxy* Theorem bekannt) verwendet werden. Ein Beweis der Korrektheit des Pumping Lemmas wird beispielsweise in dem Buch von Moll, Arbib, Kfoury [MAK88] geführt.

Das Pumping Lemma: Zunächst sei $Z_1(\hat{L}, q)$ die Teilmenge der Sprache $\hat{L}$ mit lediglich nur q Zeichen und sei $Z_2(\hat{L}, p, q)$ die Teilmenge der Sprache $\hat{L}$ mit der Eigenschaft, daß jedes Element z aus dieser Teilmenge als $uvwxy$ geschrieben werden kann, so daß

1. $|uwx| \leq q$
2. mindestens eine Zeichenkette von v und x leer ist und
3. für alle $i \geq 0$: $uv^iwx^iy \in \hat{L}$;

Sei nun L eine kontext-freie Sprache. Dann existieren Konstanten p und q, die lediglich von L abhängen, so daß $Z_1(L, q) \subseteq Z_2(L, p, q)$.

Um die Nicht-Zugehörigkeit der Sprache mathematischer Ausdrücke zu der Klasse der kontext-freien Sprachen zu zeigen, reicht es, einen Widerspruch des Pumping Lemmas herzuleitet. Es ist durchaus legitim die Elemente einer $n \times 3$-Matrix in folgender Reihenfolge niederzuschreiben: Zunächst wird die erste Spalte der Matrix mit "0" gefüllt, dann eine Spalte mit "1" und schließlich besteht die dritte Spalte nur aus "2"-Zeichen. Die Länge jeder Spalte soll identisch mit n sein, da es sich ja um eine $n \times 3$-Matrix handeln soll. Die Sprache zur Generierung dieses Matrixtypes entspricht also der Menge $L = \{w | w \in X \cup V, w = 0^n1^n2^n\}$. Es gilt nun zu zeigen, daß diese Sprache nicht zur Klasse der kontext-freien Sprachen gehört.

Sei nun diese Sprache L kontext-frei, dann müßte es auch ein p und q geben, welche die Bedingungen des Pumping Lemmas genügen. Betrachtet man $0^r1^r2^r$ mit $r > p$ und $r > q$, dann kann das Pumping Lemma angewendet werden und man kann $z = uvwxy$ schreiben. Die Teilkette uvw könnte vollständig in dem "0"-Block, "1"-Block oder "2"-Block fallen, da ja $q < r$ und erfüllt damit den ersten Punkt des Pumping Lemmas. Oder sie könnte über Blockgrenzen von "0" zu "1" oder "1" zu "2" fallen. Allerdings ist die Teilkette zu kurz, als daß sie über beide Blockgrenzen reichen könnte. Damit ist also der zweite Punkt erfüllt, der besagt, daß mindestens eine von den Variablen v oder x leer ist.

Betrachtet man jedoch nun die "aufgepumpte" Zeichenkette $uvvwxxy$, dann stellt man fest, daß mindestens die Anzahl eines Symbols nicht mit den anderen steigt und somit nicht mehr zu $0^n1^n2^n$ paßt. Das ist allerdings ein Widerspruch zur Annahme und es folgt, daß L (und somit die Sprache der mathematischen Ausdrücke) keine kontext-freie Sprache ist.

3.2 Einordnung von Parsern

Ein Parser überprüft, ob eine Eingabe bzgl. einer gegebenenen Grammatik syntaktisch korrekt ist. Eine Einordnung eines Parsers folgt der Klassifizierung der Grammatik. Man stellt fest, daß Parser, die in der Lage sind kontext-sensitive Sprachen zu überprüfen, einen höheren Berechnungsaufwand bedürfen und deswegen im praktischen Einsatz aus Gründen der langen Rechenzeit ungeeignet sind. Im allgemeinen haben Parser für Sprachen höherer Chomsky-Hierarchien einen niedrigen Berechnungsaufwand. So kommen z.B. Parser für lineare Sprachen auf einen Berechnungsaufwand von $O(n)$ während Parser für kontext-freie Sprachen $O(n^2 \log n)$ Schritte benötigen. Im hier vorgestellten Erkennungssystem wird ein Parser für kontext-freie Ausdrücke verwendet, um den Berechnungsaufwand zu minimieren, obwohl damit nicht alle möglichen legitimen, mathematischen Ausdrücke geparst werden können (siehe Abschnitt 3.1).

Ein weiteres Einordnungsmerkmal eines Parsers ist, ob er einen Bottom-Up Ansatz oder einen Top-Down Ansatz benutzt. Während bei ersterem der Parserbaum von den Blättern nach oben zu der Wurzel aufgebaut wird, arbeiten Parser mit einer Top-Down Methode in der Art, daß sie Anfragen nach gewissen Symbolen rekursiv bis zu den Blättern stellen und so den Syntaxbaum nach unten erweitern.

Eine andere wichtige Eigenschaft ist es, ob der Parser mit probabilistischen Produktionen arbeitet. Diese Art von Parsern erlaubt, auch syntaktisch fehlerhafte Ausdrücke zu generieren, indem entsprechende Produktionen existieren. Diese "fehlerhaften" Produktionen besitzen im Gegensatz zu "korrekten" Produktionen jedoch einen geringeren Wahrscheinlichkeitswert, damit im Fall einer mehrdeutigen Entscheidungssituation die Produktion bevorteilt wird, die zu einem syntaktisch korrekten Ausdruck leitet. Die Wahrscheinlichkeiten der Produktionen können entweder per Hand eingestellt werden, oder mit Hilfe eines lernfähigen Verfahrens geschätzt werden. Ein Trainingsprogramm für die Produktionswahrscheinlichkeiten kann in dem Artikel von S. Seneff [Sen92] gefunden werden.

3.3 Der Cocke Kasami Younger Algorithmus

Der in dieser Arbeit verwendete Parser basiert auf dem Cocke Kasami Younger (CKY) Algorithmus [AU72]. Dieser Parseralgorithmus wurde gewählt, weil eine Eingabe in $O(n^3)$ Schritten überprüft werden kann, ob sie von einer gegebenen kontext-freien Grammatik erzeugt werden kann, wobei n die Länge der Eingabekette ist. Der CKY Algorithmus verfolgt den Bottom-Up Ansatz und ist für eine Chomsky-Normalform (siehe Abschnitt 3.1) konzepiert. Um das Prinzip des Algorithmus darzustellen, kann man sich ein versetztes Zellengitter in Form einer Pyramide vorstellen. Die unterste Stufe dieser Pyramide hat

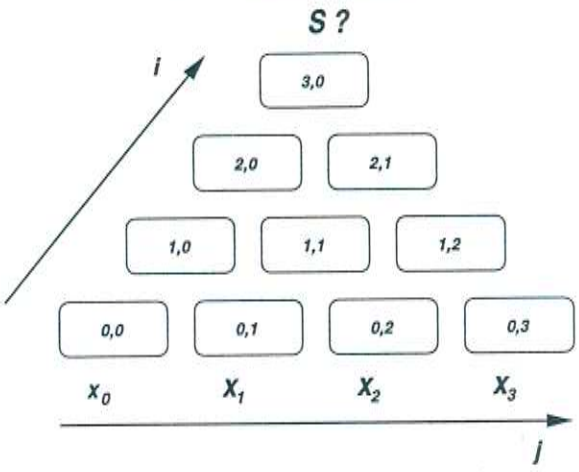


Abbildung 3.1: CKY-Pyramide

n Zellen, entsprechend der Anzahl von Terminalen in der Eingabe. Aufbauend auf der untersten 0. Stufe befindet sich versetzt eine weitere Stufe mit $n-1$ Zellen. Das setzt sich fort, bis schließlich die $(n-1)$. Stufe lediglich aus einer einzigen Zelle besteht. Das Bild 3.1 zeigt graphisch den Aufbau und die zweiwertige Indizierung der Zellen. Diese Indizierung spiegelt die Bedeckung des Eingabebereiches wieder. Das Feld (i,j) überdeckt z.B. eine Teilkette $y(i,j) = x_i \cdots x_{i+j}$ der ganzen Eingabe $z = x_0 \cdots x_{n-1}$, angefangen mit dem j. Zeichen bis hin zum $(j+i)$. Zeichen. Wobei das erste Zeichen in der Eingabezeichenkette z null als Index hat.

Zunächst wird im ersten Schritt die Eingabe bestehend aus einer Reihe von Terminalen in die erste, unterste Stufe kopiert. Der zweite Schritt des CKY-Algorithmus ersetzt alle Terminale in der ersten Stufe durch ihre nonterminalen Partner. Die weiteren Stufen werden vom Boden bis hin zur Spitze der Pyramide mit Nonterminalen gefüllt, passend zu den Produktionen der Grammatik. Dabei wird jeweils dann ein Nonterminal $A \in N$ in die Zelle (i,j) abgelegt, falls $A \stackrel{*}{\rightarrow} y(i,j)$. Befindet sich nach der Füllung in der Spitze also in Zelle $(n-1,0)$ das Startsymbol S, so ist die Eingabe z in der Sprache der Grammatik.

Der Aufwand zum Füllen einer Zelle (i,j) ist proportional zu j. Dieser niedrige Berechnungsaufwand wird zum einen durch die Verwendung der Chomsky-Normalform erzielt. Zum anderen wird die algorithmentechnische Methode des dynamischen Programmierens ausgenutzt, welche berechnete Symbole in den Zellen zwischenspeichert. Dabei wird für die Zelle (i,j) lediglich in den Zellenpaaren $((0,j),(i-1,j+1))$ bis $(i-1,j),(0,i+j))$ geprüft, ob sich dort die passenden Nonterminale B bzw. C befinden. Mit anderen Worten werden aus der Zeichenkette $y(i,j)$ alle Aufteilungen in zwei Komponenten ermittelt und getestet, ob diese bereits durch B bzw. C abgeleitet werden konnten. Der Gesamtaufwand des Füllens aller $(n+1)n/2 \in O(n^2)$ Zellen einer Pyramide mit Basisbreite n ist somit proportional zu n^3. Das Bild 3.2 zeigt eine Kombinationsmöglichkeit der Produktion $A \rightarrow BC$ von der Zelle (i,j).

Definition des CKY-Algorithmus: Gegeben ist eine kontext-freie Grammatik $G = (X,V,S,P)$ in Chomsky-Normalform, wie sie oben beschrieben worden ist. Man kann entscheiden, ob $z = x_0 \cdots x_{n-1}$ in der Sprache $L(G)$ liegt, indem man eine Pyramide nach folgenden Vorschriften aufbaut:

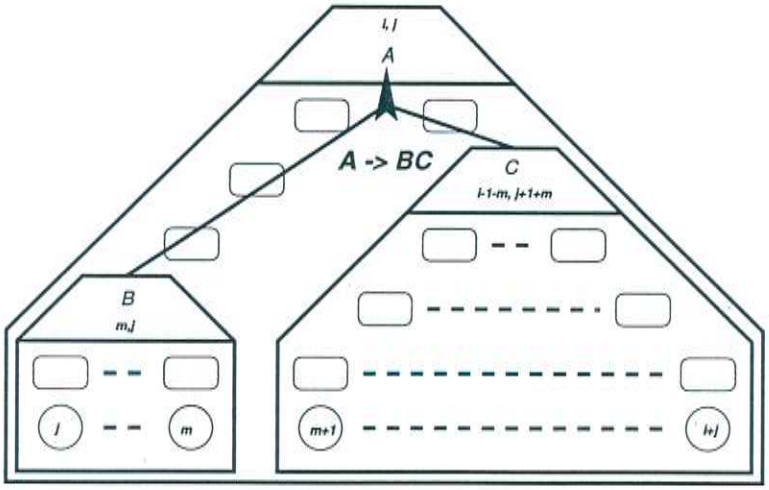


Abbildung 3.2: Das Bild zeigt eine Kombinationsmöglichkeit des Teilausdrucks $y(i,j)$ angewendet auf die Produktion $A \rightarrow BC$

1. Erzeuge die Basiszellen: Für $j = 0$ bis $n-1$, füge ein $A \in N$ in Zelle $(0,j)$, falls die Produktion $A \rightarrow x_j$ mit $x_j \in T$ Element der Grammatik ist.

2. Erzeuge die weiteren Zeilen: Für jede Zeile $i > 0$ kann man davon ausgehen, daß die unteren Zeilen bereits aufgebaut sind. Dann für $j = 0$ bis $n - i$ erzeuge Zelle (i,j) wie folgt: Für $m = 0$ bis $i - 1$ füge in (i,j) ein Element mit Symbol A ein, falls

   (a) in Zelle (m,j) ein Element mit Symbol B,

   (b) in Zelle $(i-1-m, j+1+m)$ ein Element mit Symbol C vorliegt und

   (c) falls es eine Produktion $A \rightarrow BC$ in der Menge P aller Produktionen der Grammatik G gibt.

Die Eingabe z ist dann und genau dann in $L(G)$, wenn sich in der obersten Zelle $(n-1,0)$ ein Element mit dem Startsymbol S befindet.

3.4 Fuzzy-Mengen

Der Erkennungsalgorithmus zur Bestimmung der geometrischen Relationen zweier Komponenten basiert in diesem vorgestellten System auf der Fuzzy-Logik [Zad65], [KF88].

In der klassischen Algebra wird eine Menge M auf einer Grundmenge G so erklärt, daß man festlegt, welche Elemente x der Grundmenge zur Menge M gehören sollen ($x \in M$) bzw. nicht gehören sollen ($x \notin M$). Die Zugehörigkeit bzw. Nicht-Zugehörigkeit eines Elementes $x \in G$ zur Menge M kann man Hilfe einer *charakteristischen Funktion* μ_M notieren. Dabei nimmt $\mu_M(x)$ den Wert eins an, falls $x \in M$, und den Wert null, falls $x \notin M$.

$$\mu_M(x) = \begin{cases} 1 & \text{für } x \in M \\ 0 & \text{für } x \notin M \end{cases} \tag{3.2}$$

Die Funktion μ_M hat also die Grundmenge G als Definitionsbereich und die 2-elementige Menge $\{0;1\}$ als Wertebereich. Bei einer Fuzzifikation wird diese Zweiwertigkeit zugunsten eines kontinuierlichen Wertebereichs aufgegeben. Dabei erweitert man den Wertebereich der charakteristischen Funktion auf alle reellen Zahlen zwischen 0 und 1 und interpretiert die Funktionswerte $\mu_M(x)$ als Maß für die Mitgliedschaft bzw. die Zugehörigkeit zur betreffenden Menge. Je größer der Wert $\mu_M(x)$, desto stärker die Zugehörigkeit. Als Grenzfälle sind die klassische Zugehörigkeit $\mu_M(x) = 1 \Leftrightarrow x \in M$, sowie die klassische Nicht-Zugehörigkeit $\mu_M(x) = 0 \Leftrightarrow x \notin M$ darin enthalten.

Definition von Fuzzy-Mengen: Sei G eine Grundmenge, μ_A eine Funktion der Grundmenge G in das Einheitsintervall $[0;1]$ der reellen Achse:

$$\mu_A : G \to [0;1] \tag{3.3}$$

Dann heißt die Menge A aller Paare $(x, \mu_A(x))$

$$A = \{(x, \mu_A(x)) | x \in G\} \tag{3.4}$$

eine Fuzzy-Menge über G. Der Mathematiker L. A. Zadeh [Zad65] schlägt folgende Notation für unendliche Fuzzy-Mengen mit dem klassischen Integralzeichen vor:

$$A = \int_G \frac{\mu_A(x)}{x} = \int \mu_A(x)/x \tag{3.5}$$

Viele Fuzzy-Mengen sind durch ausgesprochen einfache Mitgliedsfunktionen erklärt. Dieser Sachverhalt ist charakteristisch zumindest für den momentanen Stand der Fuzzy-Logik. Die häufigsten Formen der Funktionen sollen hier aufgelistet werden.

Type 1: μ_A ist monoton wachsend.

Type 2: μ_A ist monoton fallend.

Type 3: μ_A wächst monoton, nimmt für genau ein x seinen größten Wert $\mu_A(x) = 1$ an und fällt dann wieder monoton ab.

Type 4: μ_A wächst monoton, nimmt für einen Teilintervall der Grundmenge den maximalen Wert 1 an und fällt dann wieder monoton ab.

Die Abbildung 3.3 zeigt den Verlauf der Funktionen graphisch.

Fuzzy-Durchschnitt, -Vereinigung, -Komplement

Zur funktionellen Beschreibung von Fuzzy-Durchschnitt, Vereinigung und Komplement behilft man sich mit den Minimum-, Maximum-, bzw. Negation-Funktionen. Anlehnend an die schreibweise mit logischen Operatoren für Prädikate zur Erklärung der mengenalgebraischen Verknüpfungen kann man Operationen für reelle Zahlen zur Definition der

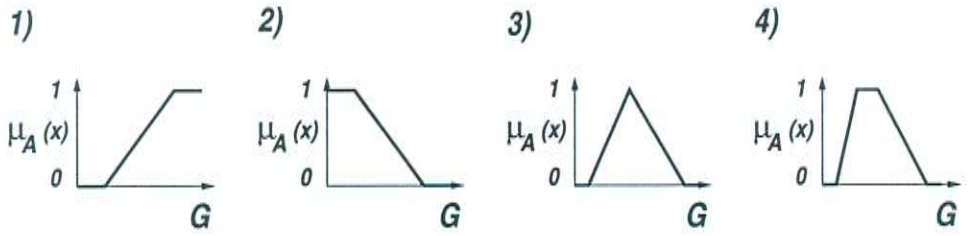


Abbildung 3.3: Typen von Fuzzy-Mengen: 1) monoton steigende 2) monoton fallende 3) trianguläre 4) trapezförmige Fuzzy-Menge

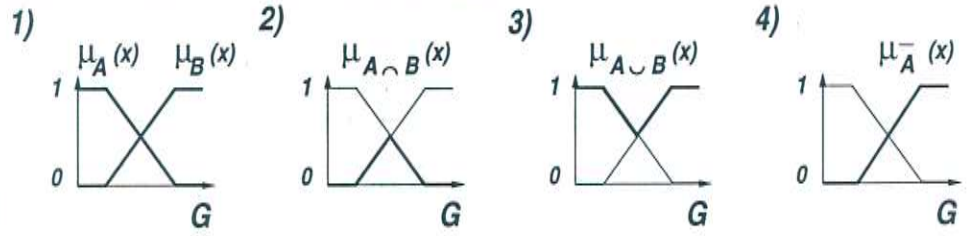


Abbildung 3.4: Fuzzy-Operationen: 1) Fuzzy-Mengen A, B 2) Durchschnitt 3) Vereinigung 4) Komplement

Fuzzy-Operationen heranziehen. In der Abbildung 3.4 sind die Operationen graphisch angezeigt.

Definition von Fuzzy-Mengen-Durchschnitt, -Vereinigung, Komplement: Auf der Grundmenge G seien die Fuzzy-Mengen A, B durch ihre Mitgliedsgradfunktionen μ_A, μ_B gegeben. Damit erklärt man

- als Fuzzy-Mengen-Durchschnitt die Fuzzy-Menge

$$A \cap B : \mu_{A\cap B}(x) = \min(\mu_A(x), \mu_B(x)) \tag{3.6}$$

- als Fuzzy-Mengen-Vereinigung die Fuzzy-Menge

$$A \cup B : \mu_{A\cup B}(x) = \max(\mu_A(x), \mu_B(x)) \tag{3.7}$$

- als Fuzzy-Mengen-Komplement die Fuzzy-Menge

$$\overline{A} : \mu_{\overline{A}}(x) = 1 - \mu_A(x) \tag{3.8}$$

t- und s-Normen

Nun sind *min*, *max* und das Einskomplement nicht die einzigen möglichen Operatoren für Fuzzy-Mengen [Yag80]. Vielmehr kann man mit einer Menge von Bedingungen an Operatoren mehrere unterschiedliche Ausprägungen entwickeln. Diese Plattform bringt die meisten einschlägigen Operationen unter, die nun Normen genannt werden. Entsprechend

dem dualen Charakter dieser Struktur gibt es zu jeder Norm eine "t-Norm" und eine "s-Norm". Es hat sich gezeigt, daß es zweckmäßig ist, diese Normen-Verknüpfungen auf dem Einheitsintervall $[0;1]$ zu erklären.

$$s, t : [0;1] \times [0;1] \rightarrow [0;1] \tag{3.9}$$

Definition von t- und s-Normen: Die zweistelligen Operationen t, s auf $[0;1]$ heißen t-Norm bzw. s-Norm, wenn sie für alle $x, y, z \in [0;1]$ folgende Eigenschaften besitzen.

- 1 (für t) und 0 (für s) sind "Neutralelemente"

$$t(1, x) = x \tag{3.10}$$
$$s(0, x) = x \tag{3.11}$$

- t, s sind kommutativ:

$$t(x, y) = t(y, x) \tag{3.12}$$
$$s(x, y) = s(y, x) \tag{3.13}$$

- t, s sind assoziativ:

$$t(t(x, y), z) = t(x, t(y, z)) \tag{3.14}$$
$$s(s(x, y), z) = s(x, s(y, z)) \tag{3.15}$$

- t, s sind monoton wachsend:

$$(x \leq y) \wedge (y \leq w) \Rightarrow (t(x, y) \leq t(z, w)) \tag{3.16}$$
$$(x \leq y) \wedge (y \leq w) \Rightarrow (s(x, y) \leq s(z, w)) \tag{3.17}$$

Nachdem nun das Gerüst der Normen definiert worden sind, sollen hier einige Normen aufgelistet werden, welche im vorgestellten System zur Verwendung kommen:

- **Definition der min-, max-Norm:** Die bereits bekannten Min-, Max-Operation sollen aus Gründen der Vollständigkeit nochmals angegeben werden

$$min(x, y) = \begin{cases} x & \text{für } x \geq y \\ y & \text{sonst} \end{cases} \tag{3.18}$$

$$max(x, y) = \begin{cases} x & \text{für } x \leq y \\ y & \text{sonst} \end{cases} \tag{3.19}$$

- **Definition der algebraischen Norm:**

$$alg_t(x, y) = xy \tag{3.20}$$
$$alg_s(x, y) = x + y - xy \tag{3.21}$$

- **Definition der beschränkenden Norm:**

$$bes_t(x,y) = max(0, x+y-1) \tag{3.22}$$
$$bes_s(x,y) = min(1, x+y) \tag{3.23}$$

- **Definition der drastischen Norm:**

$$dra_t(x,y) = \begin{cases} x & \text{für } y = 1 \\ y & \text{für } x = 1 \\ 0 & \text{sonst} \end{cases} \tag{3.24}$$

$$dra_s(x,y) = \begin{cases} x & \text{für } y = 0 \\ y & \text{für } x = 0 \\ 1 & \text{sonst} \end{cases} \tag{3.25}$$

Zur Bestimmung geometrischer Relationen zweier Subkomponenten benötigt das Erkennungssytem t- und s-Normoperationen. Wegen der gemeinsamen Eigenschaften können die oben definierten Normen wahlweise ausgetauscht und resultierende Erkennungsleistungen verglichen werden. Die Erkennungsleistungen des Gesamtsystems sind in Tabelle 6.2 in Kapitel 6 in Abhängigkeit der verwendeten Norm aufgelistet.

Kapitel 4

Datensammelprojekt und empirische Studien

4.1 Datensammelprojekt

Vorangehend zur Implementierung des Erkennungssystems wurde ein Datensammelprojekt durchgeführt, um handschriftliche Test- und Trainingsdaten zu erhalten. Diese Beispieldaten werden zum Trainieren der lernfähigen Algorithmen, sowie zur Evaluation des implementierten Verfahrens benötigt. Außerdem wurde versucht, heuristische Merkmale von den Eingaben zu entdecken, um das Erkennungssystem daraufhin zu optimieren.

Datenbestand und -bedarf

Zur Durchführung eines Trainings des Erkennungssystems auf mathematische Zeichen bedurfte es einer Trainingsmenge aus Einzelzeichen "a" - "z"und "A" - "Z", sowie Zahlzeichen "0" - "9". Diese Menge der Einzelzeichen stammte aus dem NPen Projekt [MFW95a], welche in Karlsruhe und an der Carnegie Mellon Universität in Pittsburgh gesammelt worden war. Ca. 4000 Beispiele wurden aus dieser Quelle ausgewählt, die sich durch eine gute Schreibqualität auszeichneten und von mehreren Schreibern mit unterschiedlichen Schreibstilen stammten.

Durch einen geringeren Forschungsumfang im Bereich der Erkennungsalgorithmen für mathematische Ausdrücke sind öffentliche Quellen von Stifteingaben handgeschriebener, mathematischer Ausdrücke rar und es bedurfte der Durchführung eines eigenen Datensammelprojekts. In diesem Projekt wurden mathematische Sonderzeichen und griechische Buchstaben gesammelt. Desweiteren benötigte das Erkennungssystem für geometrische Relationen Geometrieinformationen von Subkomponenten. Deswegen war es notwendig, Spender zu bitten, komplexe Ausdrücke niederzuschreiben, welche aus den Bereichen der Algebra, Analysis und Statistik, sowie der Informatik und Aussagenlogik stammten (siehe Anhang A.2), um die Spendedaten dann auf Teilausdrücke zu segmentieren und ihre Relationen zueinander festzuhalten.

Aufbau eines Datensammelplatzes

Ein robustes und benutzerfreundliches Datensammelprogramm mußte implementiert werden, damit die Spender möglichst auf eine natürliche Weise Daten eingeben konnten. Zusätzliche Editierfunktionen zur Korrektur von Eingaben wurden berücksichtigt. Alle notwendigen Informationen wurden auf dem Bildschirm angezeigt, an dem der Benutzer

auch die Eingabe durchführte. Die Eingabe wurde von einem drucksensitiven Bildschirm entnommen, welcher die Stiftspitze lokalisierte. Es war allerdings notwendig einen Bildschirm zu verwenden, der horizontal stand, da bei einer normalen aufrechten Bildschirmposition Benutzer schnell ermüdeten und die Schreibqualität nachließ (siehe auch [Tap86]).

Da Sonderzeichen innerhalb der Ausdrücke aus speziellen Sachgebieten stammten, war es notwendig, Personen mit geeignetem Vorwissen für die Spende zu gewinnen. Dazu wurden Studenten mittels eines Aushanges lediglich bei ausgewählten Instituten der Carnegie Mellon Universität von diesem Projekt informiert (siehe Anhang C.1). Erklärte sich ein Spender bereit, mathematische Ausdrücke zu schreiben, wurden zunächst einige persönliche Daten des Spenders notiert, wie Herkunft, Geschlecht und Alter. Nach einer kurzen Einführung in das Aufzeichnungsgerät, wurde er aufgefordert, die Zeichen und Ausdrücke auf einer Eingabefläche zu kopieren. Der Schreiber wurde gebeten, so zu schreiben, wie er auch auf Papier schreiben würde. Es wurden keine Restriktionen bzgl. des Schreibstils gemacht, was zur Konsequenz hatte, daß sich die aufgebaute Datenbank durch eine hohe Varianz von verschiedenen Schreibstilen auszeichnete. Ausdrücke mit Sonderzeichen, die dem Benutzer nicht bekannt waren, sollten übersprungen werden.

Nachdem der Spender ca. 100 Sonderzeichen und weitere 250 Ausdrücke geschrieben hatte, wurde er mit einem kleinen Entgelt belohnt. Die Daten wurden nochmals von einem Mitglied des Datensammelteams auf Qualität kontrolliert. Fehleingaben aus mangelndem Verständis oder durch Versagen des Eingabesystems wurden ausgemustert. Zusätzlich wurde ein Teil der Ausdrücke auf Einzelzeichen segmentiert und die geometrischen Relationen notiert.

Ergebnis der Datensammlung

Es wurden 22 Studenten der Carnegie Mellon Universität gewonnen, handschriftliche Daten von mathematischen Ausdrücken und Sonderzeichen zu spenden. Die Daten weisen keinen festen Schreibstil auf und variieren zwischen kursiver und gedruckter Schrift. Ein Datenumfang von ca. 6300 Ausdrücken und 2700 Sonderzeichen wurde erzielt. Einen Ausschnitt aus der gespendeten Datenmenge ist im Bild 4.1 gezeigt. Ca. 3000 Zeichen wurden aus ca. 300 Ausdrücken segmentiert und die geometrischen Relationen ihrer Subkomponenten festgehalten. Die Daten wurden im Unipen-Standard (siehe `http://hwr.nici.kun.nl/unipen`) abgespeichert, ein Format, das zur Spezifizierung von handschriftlichen On-Line-Daten entwickelt wurde. Eine Einführung des Formats, sowie ein Vorschlag einer Erweiterung für mathematische Ausdrücke ist in Anhang B angegeben. Die Datensammelaktion dauerte fünf Wochen.

4.2 Empirische Studien

Mittels der ca. 6300 mathematischen Ausdrücke, die aus dem Datensammelprojekt stammten, wurde versucht, Regelmäßigkeiten bzw. Heuristiken in der Stifteingabe zu finden, um diese Erkenntnisse zur Entwicklung der Erkennungsmethoden des Gesamtsystems auszunutzen.

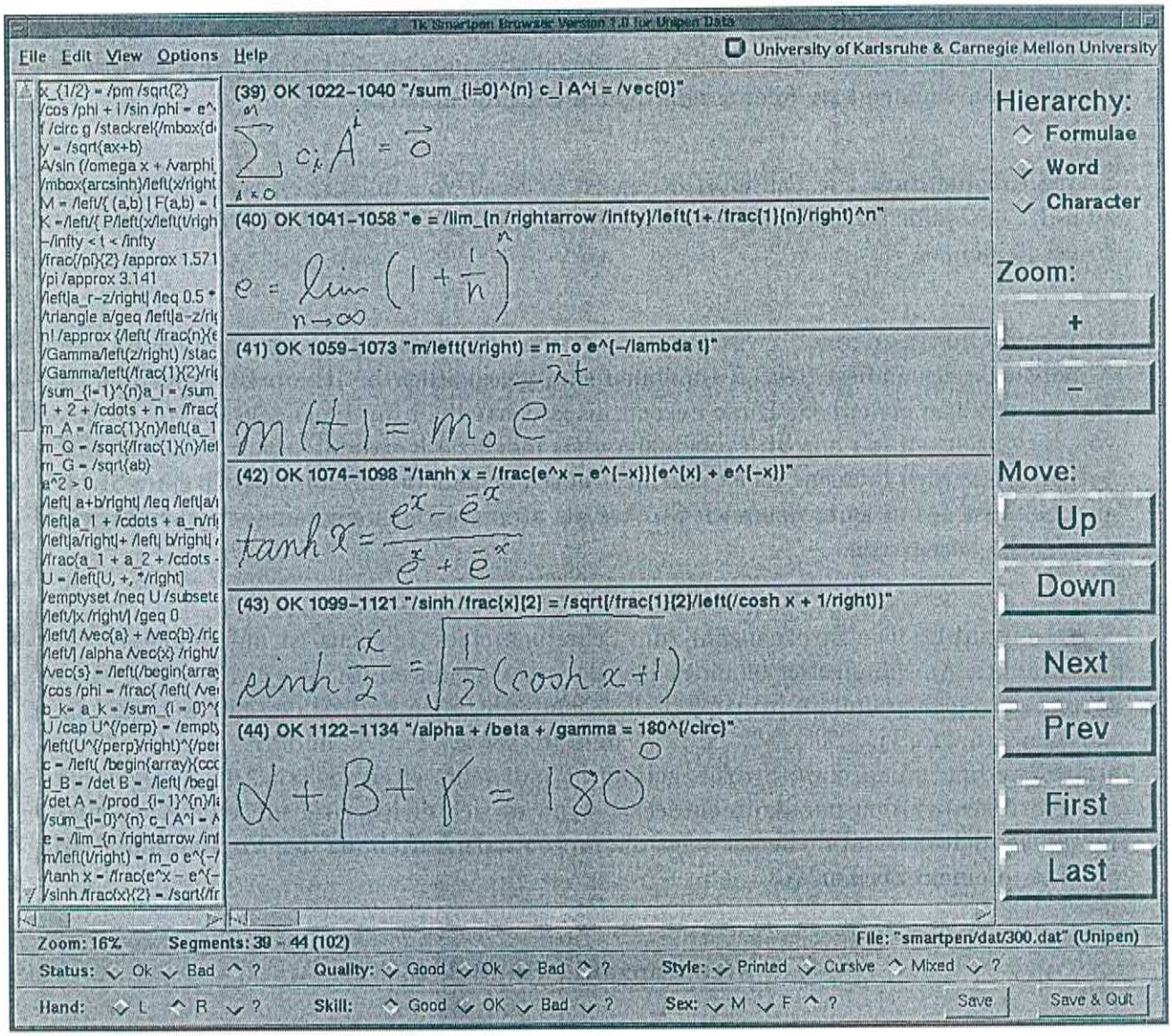


Abbildung 4.1: Ein Ausschnitt aus der gespendeten Datenmenge

Eingabezeit: Die Eingabezeit eines Ausdrucks auf einem drucksensitiven Bildschirm unterscheidet sich unwesentlich von der auf Papier. Die Schreibgeschwindigkeit verkleinert sich, falls komplexere Ausdrücke kopiert werden sollten. Es wurde eine Eingabezeit von ca. $0,8$ Sekunden pro Symbol gemessen. Die Zeit, die durchschnittlich für das Zeichnen eines Strokes gemessen wurde, betrug $0,5$ Sekunden. Eingabepausen zwischen Strokes betrugen ca. $0,4$ Sekunden. Daraus ergibt sich eine durchschnittliche Schreibdauer eines mathematischen Ausdrucks mit 8 Zeichen von $9,6$ Sekunden. Diese Zeitdauer ist maßgebend für eine Realisierung eines Erkennungssystems, das in Echtzeit Ergebnisse liefern soll. Ein inkrementeller Aufbau des Algorithmus ("Run-On") ist mit dieser Anforderung zwingend.

Qualität der Eingabe: Im Gegensatz zur Qualität bei der Eingabe von Texten ist festzustellen, daß die Qualität der Eingabe von Einzelbuchstaben bei mathematischen

Ausdrücke relativ hoch ist. Das kann vornehmlich daran liegen, daß in mathematischen Ausdrücken weniger Redundanzen in der Eingabe vorhanden sind als bei Texten, was den Benutzer anscheinend zu einer sorgfältigeren Eingabe anhält.

Schreibrichtung: Die Schreibrichtung ist generell von links nach rechts. Durch die zwei-dimensionale Natur mathematischer Ausdrücke sind aber Rückläufe des Stiftes nicht zu vermeiden.

Einzelzeichengrenzen: Zwischen zwei Zeichen befindet sich in 98,4% aller Fälle ein Abheben des Eingabestiftes. Ausnahmen sind gelegentlich bei Hauptkomponenten und deren Indizes zu bemerken. Diese Erkenntnis ist vorteilhaft für die Detektion von Grenzen von Einzelzeichen, da nur Stiftabhebungen als mögliche Kandidaten von Zeichengrenzen betrachtet werden müssen. Es ist also nicht nötig, Segmentgrenzen innerhalb eines Strokes anzunehmen. Man kann demnach Strokes als atomare Bausteine eines mathematischen Ausdrucks bezeichnen.

Strokeanzahl: Die Strokeanzahl zum Schreiben eines Zeichens ist auf wenige Strokes beschränkt. Am häufigsten benötigt der Benutzer genau ein Stroke pro Zeichen. Diese Menge der Zeichen umfaßt prinzipiell alle Kleinbuchstaben und Zahlen mit wenigen Ausnahmen, wie "i", "j", "k", "t", "x", "y" und "z" sowie "4", "5" und "7". Bei Großbuchstaben werden häufig zwei Strokes gebraucht und sehr selten mehr als vier. Sechs und mehr Strokes wurden nur dann entdeckt, wenn es sich gleichzeitig um eine Zeichenkorrektur handelte. Die Tabelle 4.1 zeigt die prozentuale Aufteilung der Vorkommnisse von Strokes in einem Zeichen.

Strokeanzahl	Häufigkeit
1	46,4%
2	23,1%
3	15,9%
4	4,6%
5	5,1%
6	2,2%
7	1,8%
8 und mehr	0,9%

Tabelle 4.1: Anzahl der Stokes und ihre Häufigkeit

Die Tabelle macht deutlich, daß 4.9% aller Zeichen mit weniger als sechs Strokes geschrieben worden. Je kleiner die Anzahl der erlaubten Strokes zum Schreiben eines Zeichen ist, desto kleiner ist auch der Berechnungsaufwand. Zwar liegen alle Verfahren mit einer konstanten oberen Anzahl von erlaubten Strokes pro Zeichen in der gleichen Menge des O-Kalküls, doch ist eine Begrenzung auf fünf Strokes ein Kompromiß zwischen praktikablen Berechnungsaufwand und akzeptabler Erkennungsleistung.

Eingabereihenfolge, zeitliche Konvexität: Eine wichtige Tatsache ist, daß Teilausdrücke meist vollständig beendet werden, bevor ein neuer Teilausdruck angefangen wird. Bei der Betrachtung des Beispiels eines Ausdrucks

$$\sum_{n=0}^{m} x^n \tag{4.1}$$

eines Spenders wurde beobachtet, daß zunächst das Summenzeichen Σ geschrieben worden ist, folgend mit der unteren Grenze "$n = 0$", weiterhin "m" als obere Grenze der Summe und abschließend "x^n". Alle vier Komponenten wurden in einem abgeschlossenen Zeitblock begonnen und beendet. Ein Rückkehren zu einem nicht vollendenten Teilausdruck wurde nur dann beobachtet, wenn es sich um eine Korrektur handelte. Diese Arbeit möchte aber nicht auf handschriftliche Korrekturverfahren eingegangen. Es sei in diesem Zusammenhang auf Korrekturprogramme für Stifteingaben verwiesen, wie sie z.B. in der Diplomarbeit von W. Hürst [Hue97] erörtert werden.

Die Eingabereihenfolge der Komponenten eines Teilausdrucks variierten jedoch. In Bezug auf das obere Beispiel sind Eingaben von Benutzern bekannt, bei denen zunächst das Summenzeichen geschrieben wurde, dann die unteren und oberen Grenzen, bevor letzlich der Summand aufgezeichnet worden ist, während andere Spender wiederum die Grenzen am Ende hinzufügten. Ein Erkennungssystem für mathematische Ausdrücke sollte in der Lage sein, derartige Varianzen der Eingabereihenfolge zu akzeptieren.

Geometrische Relationen: Mathematische Ausdrücke erhalten ihren semantischen Gehalt auch durch die geometrische Relation von Teilausdrücken zueinander.

Es gibt zwei verschiedene Relationsarten: zum einen die Richtungsrelation, zum anderen die Größenrelation. Beide Informationen spielen eine Rolle bei der richtigen semantischen Bestimmung des mathematischen Ausdrucks. Welche geometrischen Relationen existieren, wurde bereits im Abschnitt 2.2 dargestellt.

Kapitel 5

Der Algorithmus

5.1 Überblick der Module des Gesamtsystems

Das Problem der maschinellen Erkennung von handgeschriebenen, mathematischen Ausdrücken kann in mehrere Teilaufgaben unterteilt werden. Ein Überblick des Gesamtsystems wird im Bild 5.1 dargestellt.

Eingabe: Die Eingabe der Stifttrajektorie erfolgt durch ein Aufzeichnungsprogramm auf einem Rechner mit einem drucksensitiven Bildschirm.

Vorverarbeitung: Segmentierung der Stifttrajektorie in Strokes.

Parser: Der Parser generiert den Syntaxbaum des mathematischen Ausdrucks mittels einer entsprechenden Grammatik, sowie der Erkennungsalgorithmen für Zeichen und geometrische Relationen.

Ausgabe: Der vom Parser generierte Syntaxbaum wird mit dem Ausgabeprogramm in eine geeignete Textstruktursprache umgewandelt, die Eingabe eines entsprechenden Darstellungsprogramms ist. Dieses Programm visualisiert den mathematischen Ausdruck.

5.2 Eingabe

Die Eingabe wird mittels eines Computerstiftes und einem drucksensitiven Bildschirm realisiert. Ein Aufzeichnungsprogramm speichert die Trajektorie des vom Benutzer geführten Stiftes und zeigt sie gleichzeitig auf dem Monitor an. Das Programm wurde in Tcl/Tk [Ous94] geschrieben. Es bietet dem Benutzer zusätzliche Funktionen, wie Löschen, Verschieben, Laden, Speichern, usw. an. Die Benutzeroberfläche ist im Bild 5.2 gezeigt.

5.3 Vorverarbeitung

Die allgemeine Aufgabe der Vorverarbeitung beschränkt sich auf die Segmentierung der Trajektorie in Strokes. Diese Aufgabe ist einfach, da bereits das Aufzeichnungsgerät ein Absetzen des Stiftes von der Schreibfläche detektiert. Jedoch wurden bei einigen

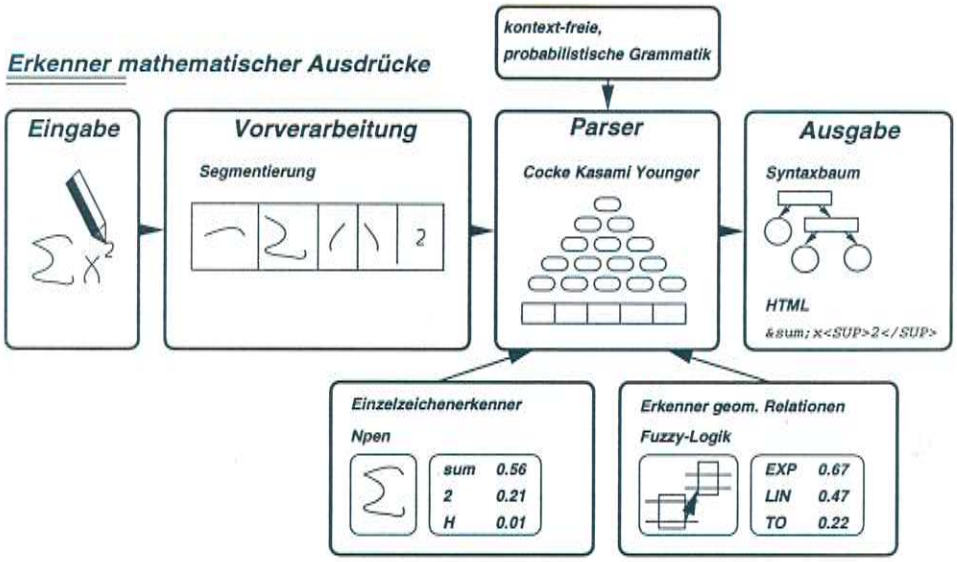


Abbildung 5.1: Die Gliederung des Gesamtssystems

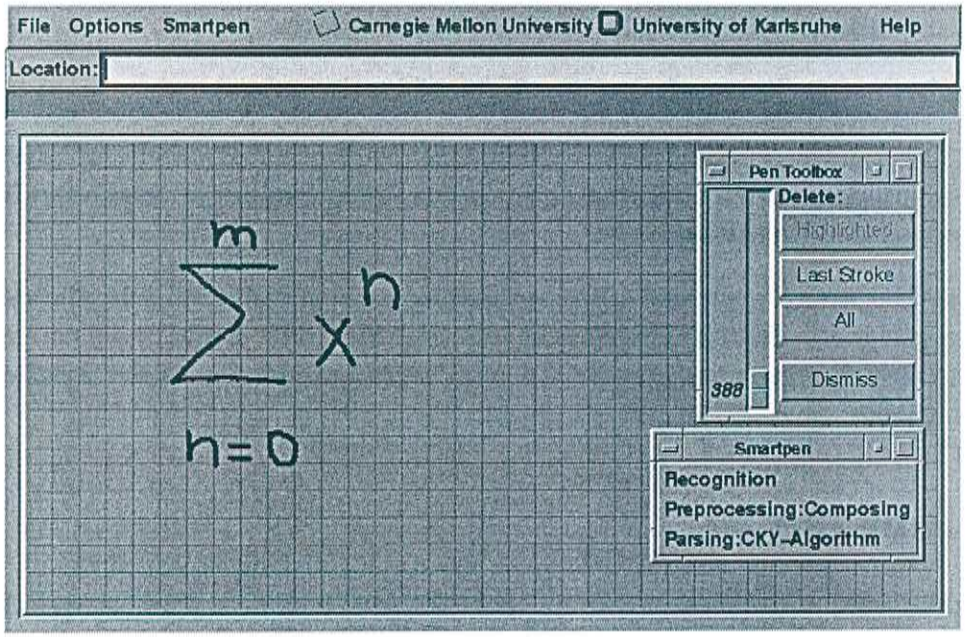


Abbildung 5.2: Die Eingabeschnittstelle des Erkennungssystems

Geräten eine fehlerhafte Detektion bemerkt, die auf kurze Sprünge des Stiftes auf der rauhen Oberfläche der Schreibfläche zurückzuführen sind. Diese vom Benutzer ungewollten Sprünge des Stiftes müssen von dem Segmentierungsprogramm erkannt und eliminiert werden. Dazu wurde ein Zeitschwellwert von $30ms$ gewählt, der die Dauer einer Stiftabsetzpause nicht unterschreiten darf, um als Strokesegmentgrenze interpretiert zu werden.

5.4 Parser

Das Hauptmodul des Erkennungssystems ist der Parser, der bei gegebener Grammatik für mathematische Ausdrücke einen Syntaxbaum aufbaut. Dabei behilft sich der Parser zusätzlich mit einem Erkennungsalgorithmus für mathematische Zeichen bzw. einem Erkennungsalgorithmus für geometrische Relationen von Teilausdrücken.

5.4.1 CKY für beliebige kontext-freie Grammatiken

In der konkreten Implementierung des Parsers für die Erkennung mathematischer Ausdrücke wurde der CKY Algorithmus gewählt, welcher sogar beliebige kontext-freie Grammatiken zuläßt. Diese Verallgemeinerung auf beliebige Produktionslängen hat einen Vorteil in der Implementierung, da alle notwendigen Komponenten gleichzeitig dem Relationserkenner zur Verfügung gestellt werden können. Die Produktion der Grammatik für einen Summenausdruck z.B. ist mit bis zu vier Subkomponenten ausgestattet ($\sum$, $E_{SUMFROM}$, E_{SUMTO}, E_{BODY}), die zueinander in der richtigen geometrischen Relation stehen müssen. Diese vier Subkomponenten können nun geschlossen als Eingabe dem Erkennungsalgorithmus für geometrische Relationen zugeführt werden.

Eine Verallgemeinerung des CKY auf alle möglichen kontext-freien Grammatiken erhöht den Aufwand zu $O(n^{m+1})$, wobei m die längste Produktionslänge in der Grammatik darstellt. Der einzige Unterschied zum CKY liegt in der Füllung der Zellen. Statt wie üblich nur i Zellenpaare zu observieren, um Zelle (i, j) zu füllen, müssen nun $O(i^{m-1})$ Kombinationen betrachtet werden. Diese Anzahl der Kombinationen enspricht der Zahl der möglichen Segmentierungen in m Teilketten einer Zeichenkette mit i Zeichen.

Der Parseralgorithmus setzt voraus, daß eine Segmentgrenze zwischen Zeichen mit einem Stiftabheben zusammenfällt. Das ist keine große Einschränkung, da in nur $1,6\%$ aller Fälle andersartige Beobachtungen gemacht wurden (siehe auch Abschnitt "Empirische Studien" in 4.2). Somit können die Strokes der Stifteingabe als Terminale der Grammatik aufgefaßt werden. Der Parser liefert dem Einzelzeichenerkenner bis zu fünf zeitlich benachbarte Strokes. Dieser ermittelt die n besten Zeichenhypothesen. Aus diesen Informationen wird ein Element generiert, welches passend der Position der Teilkette in der Gesamteingabe und der verwendeten Strokeanzahl in eine Zelle der Pyramide eingeschrieben wird. So werden z.B. die Elemente mit drei Strokes, deren Indizes in der Eingabezeichenkette 3, 4 und 5 sind, in die Zelle $(2, 3)$ eingefügt.

Entsprechend der Beschränkung auf maximal fünf Strokes pro Zeichen generiert der Einzelzeichenerkenner nur Elemente für die untersten fünf Stufen der CKY-Pyramide. Der Einzelzeichenerkenner wird im weiteren Verlauf der Füllung der Pyramide vom Parser nicht mehr benötigt. Stattdessen versucht nun der Parser die Produktionen der Grammatik so anzuwenden, daß in der obersten Zelle Elemente mit dem Startzeichen ermittelt werden. Dabei wird versucht zu jeder Produktion geeignete Elemente aus unteren Schichten zu finden, die zum einen passende Symbole der Grammatik repräsentieren und zum anderen zueinander geometrische Relationen einnehmen, die der Anforderung der Produktion entspricht. Dieses Verfahren wird beginnend von Boden hin zur Spitze wiederholt. Das Bild 5.3 zeigt den Einsatzbereich der Erkennungsalgorithmen während des Parsens.

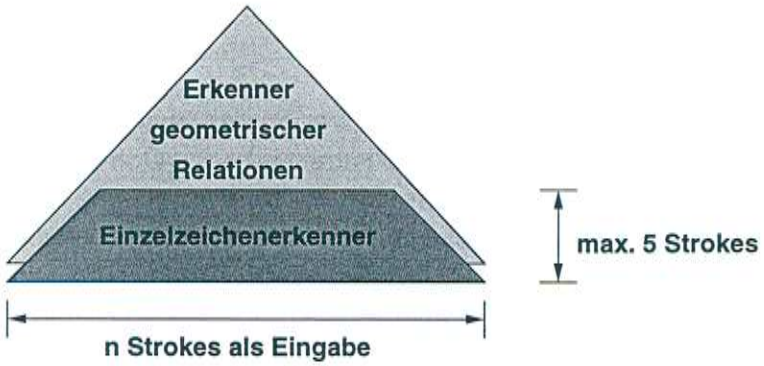
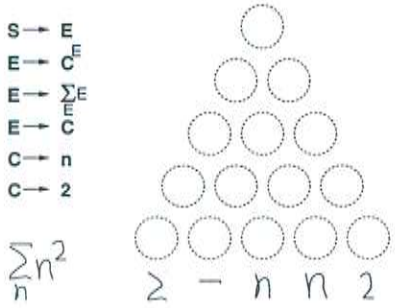
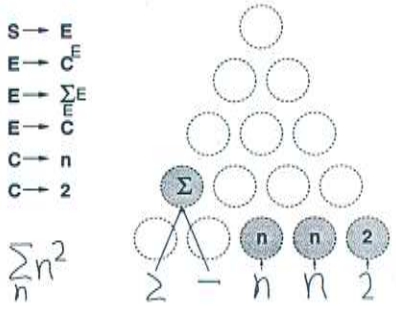


Abbildung 5.3: Der Einsatzbereich der Erkennungsalgorithmen

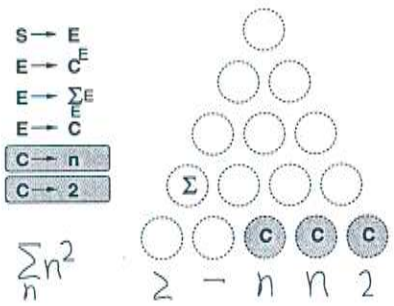
Ein Beispiel eines Parservorgangs soll mit Hilfe der folgenden Abbildungen verdeutlicht werden. Die Abbildungen zeigen Schritt für Schritt wie die Zellen der CKY-Pyramide bzgl. der gegebenen Grammatik für mathematische Ausdrücke und einer Eingabe von unten nach oben gefüllt werden. Dabei sind alle Produktionen links im Bild angegeben, die während des Parsens benötigt werden.



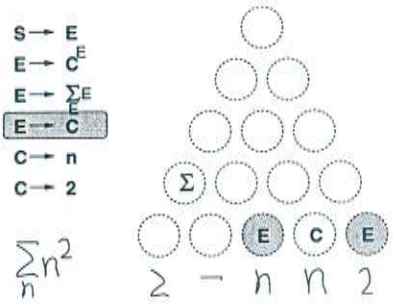
1. Parservorgang



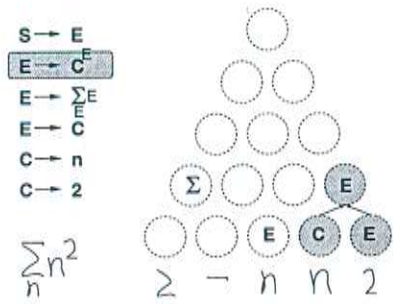
2. Parservorgang



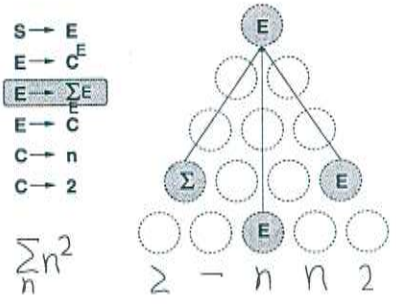
3. Parservorgang



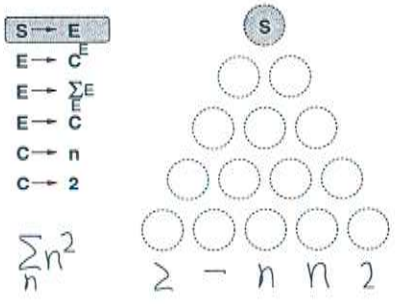
4. Parservorgang



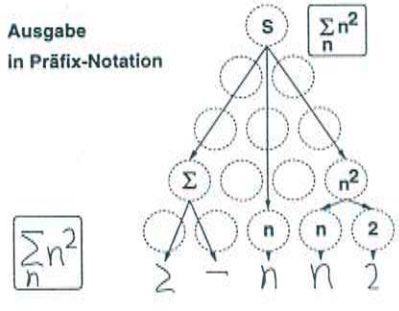
5. Parservorgang



6. Parservorgang



7. Parservorgang



8. Parservorgang (Ausgabe)

Das erste Bild "1. Parservorgang" zeigt eine leere CKY-Pyramide. Der Ausdruck

$$\sum_{n} n^2$$

wurde vom Benutzer mit fünf Strokes geschrieben, wobei das Summenzeichen $\sum$ aus zwei Strokes besteht. Demgemäß hat die Pyramide eine Breite von fünf Zellen.

Im zweiten Bild "2. Parservorgang" sind bereits die Zeichen als Nonterminale in den Zellen der Pyramide enthalten. Dazu wurde der Einzelzeichenerkenner verwendet, der aus allen Kombinationen benachbarter Strokes Zeichenhypothesen ermittelte. Das Summenzeichen $\sum$ steht im Gegensatz zu den anderen Einzelzeichen in der zweiten Stufe der Pyramide, da es aus zwei Strokes zusammengesetzt wurde. Im Bild sind aus Gründen der Übersichtlichkeit nur die Zeichenhypothesen (n, 2 und $\sum$) dargestellt, die zur Lösung führen. In der Praxis sind alle Zellen bis zur fünften Stufe mit Zeichenhypothesen besetzt.

Das dritte Bild "3. Parservorgang" zeigt zum ersten Mal eine Anwendung zweier Produktionen $C \rightarrow n$ und $C \rightarrow 2$. Die Verwendung der Produktionen ist in der Darstellung durch eine graue Hinterlegung angezeigt. Die Produktionen ersetzen die Buchstabensymbole durch ein abstraktes Grammatiksymbol C, welches für die Klasse der Einzelzeichen steht.

Das Symbol C der Grammatik wird durch das Symbol E im Bild "4. Parservorgang" ersetzt (E steht für einen mathematischen Ausdruck). Die hervorgehobene Produktion $E \rightarrow C$ kommt dabei zur Anwendung. Es sei erwähnt, daß alle bisher ermittelten Symbole bestehen bleiben und für Produktionen zur Verfügung stehen. In diesem Beispiel sollen jedoch nur die relevanten Symbole und Produktionen sichtbar gemacht werden.

Der Erkenner für geometrische Relationen kommt in Bild "5. Parservorgang" zum Einsatz. Zur Produktion $E \rightarrow C^E$ können passende Elemente in der CKY-Pyramide gefunden werden, die nicht nur die notwendigen Symbole C bzw. E besitzen, sondern auch in der korrekten geometrischen Exponent-Relation zueinander stehen. Das neu generierte Symbol kommt in die zweite Stufe, da der Teilausdruck n^2, welcher mit dem neuen Symbol repräsentiert wird, zwei Strokes enthält.

Im Bild "6. Parservorgang" wird die Produktion für einen Summenausdruck angewendet. Die drei Subkomponenten $\sum$, n und n^2 werden dazu herangezogen. Durch die Anwendung dieser Produktion entsteht ein Element mit dem Grammatiksymbol E in der obersten Zelle, welches dann auch mit $S \rightarrow E$ zum Startsymbol umgewandelt werden kann (Bild "7. Parservorgang"). Durch die Existenz des Startsymbols S in der Spitze der CKY-Pyramide ist konstruktiv bewiesen, daß die Eingabe einen mathematische Ausdruck darstellt. Der Syntaxbaum ist durch die Anwendungsfolge der Produktionen definiert (Bild "8. Parservorgang").

5.4.2 Probabilistische Parser

Da zum einen fehlerhafte Ausdrücke als Eingabe nicht grundsätzlich abgelehnt werden sollten, und um die Hypothesen der Einzelzeichen und Relationen mit deren Wahrscheinlichkeitswerten zu verarbeiten, bietet sich die Benutzung eines Parsers mit probabilistischen Produktionen an.

Die Menge aller Komponenten in der CKY-Pyramide soll hier mit K bezeichnet werden. Es werden zu jeder Komponente $k \in K$ zwei Werte k_S und k_p eingeführt. Zum einen beschreibt $k_S \in (T \cup N)$ ein Symbol aus der Grammatik G, welches das Element repäsentiert, zum anderen ist $k_p \in [0,1]$ die Wahrscheinlichkeit der Symbolzugehörigkeit der Komponente. Der Fall $k_S = A$ und $k_p = 0$ entspricht z. B., daß das Element k das Symbol A *nicht* repäsentiert.

Die Wahrscheinlichkeiten der Produktion mit gleichem linken Nonterminal summieren sich auf eins.

$$\forall_{A \in N} \sum_{q=A \to BC} p(q) = 1 \tag{5.1}$$

$$\forall_{A \in N} \sum_{q=A \to v} p(q) = 1 \tag{5.2}$$

Die Symbolzugehörigkeit und ihre Zugehörigkeitswahrscheinlichkeit einer Komponente werden wie folgt berechnet:

$$k_S := A \tag{5.3}$$

$$k_p := \begin{cases} p(q)l_p & \text{falls } q = A \to v, l_S = v \\ p(q)l_p m_p & \text{falls } q = A \to BC, l_S = B, m_S = C \end{cases} \tag{5.4}$$

Die Wahrscheinlichkeitswerte der Produktionen werden mittels eines Trainingsprogramms berechnet. Dabei werden die verwendeten Produktionen zur Generierung der Trainingbeispiele gezählt und ihre Wahrscheinlichkeitswerte entsprechend gesetzt. Eine derartige Trainingsmethode kann in dem Artikel von S. Seneff [Sen92] nachgelesen werden.

Konsequenz dieser Methode ist, daß sich in einer Zelle der CKY Pyramide nun mehrere Komponenten mit gleichen Nonterminalsymbolen befinden können, jedoch aus unterschiedlichen Produktionen entstanden sind. Entsprechend gibt es im allgemeinen nicht nur eine Lösung einer Interpretation der Eingabe als mathematischer Ausdruck, sondern eine mehrelementige Lösungsmenge

$$\{k | k_S = S, k \text{ ist Element der obersten CKY-Pyramidenzelle}\}.$$

Die Elemente der Lösungsmenge unterscheiden sich durch ihre Produktionsfolgen, aus denen sie entstanden sind.

Die folgende Tabelle zeigt einen Ausschnitt der Grammatik, die aus den vorangegangen Überlegungen resultiert. Die Werte überhalb der Ableitungspfeile sind die Wahrscheinlichkeitswerte der Produktionen.

$$\begin{aligned} G &= (X, V, S, P) \\ X &= \{A - Z, \\ &\quad a - z, \\ &\quad 0 - 9, \\ &\quad alpha - zeta, \\ &\quad \textstyle\sum, prod, int, =, <=, >=, <, >, (,), \infty, \cdots\} \end{aligned}$$

$$
\begin{aligned}
V &= \{S, E, E_S, E_{SUMFROM}, C, \cdots\} \\
P &= \{S \xrightarrow{1} E, \\
&\quad E \xrightarrow{0,2} EE_S \\
&\quad E \xrightarrow{0,3} E + E_S \\
&\quad E \xrightarrow{0,2} E - E_S \\
&\quad E \xrightarrow{0,3} E_S \\
&\quad E_S \xrightarrow{0,15} C \\
&\quad E_S \xrightarrow{0,1} C_E \\
&\quad E_S \xrightarrow{0,1} C^E \\
&\quad E_S \xrightarrow{0,05} C_E^E \\
&\quad E_S \xrightarrow{0,1} \sum_{E_{SUMFROM}}^{E} E \\
&\quad E_S \xrightarrow{0,1} (E) \\
&\quad E_{SUMFROM} \xrightarrow{0,2} C \\
&\quad E_{SUMFROM} \xrightarrow{0,8} C = E \\
&\quad C \xrightarrow{0,02} a \\
&\quad C \xrightarrow{0,02} b \\
&\quad \ldots \\
&\quad \}
\end{aligned}
$$

Diese Grammatik erlaubt, einfache Ausdrücke zu parsen, wie arithmetische Ausdrücke "$a+b-c$" mit Klammerung oder Summationsausdrücke "$\sum i = 2nn$". Die geometrischen und weiteren Informationen einer Komponente, die für das korrekte Parsen notwendig wären, sind nicht vorhanden. So soll z.B. die Produktion $E_S \rightarrow CE$ nur dann zur Anwendung kommen, falls C einer Variable entspricht und die Komponente E zu C als Index steht, da z.B. "6_2" in mathematischen Ausdrücken selten vorkommt.

Die Idee, den Symbolen weitere Informationen beizufügen und Produktionen mit zusätzlichen Regeln zu versehen, leitet zur Benutzung einer attributierten Grammatik.

5.4.3 Attributierte Grammatiken

Attributierte Grammatiken wurden von N.Knuth [Knu68] eingeführt. Sie dienen vornehmlich zum Parsen von Computersprachen und erregten schon bald nach ihrer Einführung Aufmerksamkeit. Eine umfangreiche Theorie entwickelte sich aus dieser Idee. Attributierte Grammatiken wurden bald in vielen Projekten angewendet. Bei U. Karsten [Kas80, Kas76], sowie in dem Buch von W. M. Waite und G. Goos [MG84] sind theoretische Hintergründe zu finden.

Definition einer attributierten Grammatik: Es gibt in der einfachen Form einer attributierten Grammatik drei Bestandteile (G, A, R):

- eine kontext-freie Grammatik G=(V,X,S,P), wie sie bereits oben vorgestellt wurde,
- $A = \bigcup_{x \in T \cup N} A(x)$ eine Menge von Attributen für jedes Symbol der kontext-freien Grammatik
- $R = \bigcup_{p \in P} R(p)$ eine Menge von Attributsregeln, die auf Attribute der Symbole von der rechten Seite einer Produktion (im folgenden vererbende Attribute genannt) angewendet werden und somit das Attribut der linken Seite (synthetisches Attribut) bestimmen. Attribute von Terminalen bezeichnet man als spezifische Attribute.

Die Idee ist nun, mittels der Attribute dem Terminal bzw. Nonterminal weitere Informationen anzuheften, um Entscheidungshilfen während des Parsens zu geben. So ist z.B. für die Ermittlung der geometrischen Relation zweier Komponenten ihre umrahmenden Rechtecke sowie die Lage ihrer Basis- und Mittellinien relevant. Diese Werte werden in Form von Attributen jedem Element beigefügt und bei jeder Produktion entsprechend aktualisiert. Eine Auflistung aller Attributswerte, der Grund ihrer Existenz und ihre Berechnungsform soll im folgenden angegeben werden.

Zur Zeit besitzt jede Komponente k in der CKY-Pyramide folgende Attributswerte, die jeweils von den Attributsregeln berechnet werden, oder vordefiniert von dem Einzelzeichenerkenner stammen.

Symbol: Jede Komponente repräsentiert ein Symbol $k_S \in (T \cup N)$ der kontext-freien Grammatik G, wie sie bereits im Abnschnitt "probabilistische Grammatik" 5.4.2 vorgestellt wurde.

Score: Die Berechnungsvorschrift für k_p ändert sich zum Vergleich zur probabilistischen Grammatiken in der Form, daß nun auch die Hypothese des Relationserkenners $f_\Sigma(k^1, ..., k^n)$ in Abhängigkeit der beteiligten Komponenten k^1 bis k^n einfließt.

$$k_p = p(q) f_\Sigma(k^1, ..., k^n) \prod_i k_p^i, \tag{5.5}$$

wobei $q = A \rightarrow A_1 \cdots A_n \in P$, $k_S^1 = A_1, \ldots, k_S^n = A_n$.

Die Faktoren im Einzelnen:

Produktionswahrscheinlichkeit $p(q)$**:** Sie stammen aus dem Training der probabilistischen Grammatik.

Scores der vererbenden Attribute k_p^i**:** Die Zugehörigkeitswahrscheinlichkeiten stammen rekursiv von vorangegangenen Berechnungen dieser Attributsregel.

Konfidenzmaß des Relationserkenners $f_\Sigma(k^1, ..., k^n)$**:** Der Erkenner für geometrische Relationen liefert eine entsprechende Einschätzung der gefordeten geometrischen Positionsverhältnisse Σ der Komponenten k^i. Dabei steht Σ für eine bestimmte geometrische Struktur wie z. B. IND für die Index-, EXP für Exponent-, *sum* für die Summenstruktur von Subkomponenten. Das Konfidenzmaß einer Summenstruktur *sum* wird z.B. aus ihren elementaren, geometrischen Relationen abgeleitet. Dabei werden die entsprechenden Hypothesen des Erkenners miteinander multipliziert.

$$f_{sum}(\sum_{A_1}^{A_2} A_3) = f_{FROM}(\sum, A_1) f_{TO}(\sum, A_2) f_{LIN}(\sum, A_3)$$

Im Abschnitt "Erkenner von geometrischen Relationen" 5.6 wirde erklärt, wie die Konfidenzmaße berechnet werden. Die für die Erkennung notwendigen Daten der umrahmenden Rechtecke und der Höhen der Basis- und Mittellinien werden in Attributswerten festgehalten:

Bounding Box Die Information des umrahmenden Rechtecks besteht aus vier Daten $(k_{x(min)}, k_{y(min)}, k_{x(max)}, k_{y(max)})$, die sich aus den einhüllenden Rahmen der Komponenten ergeben.

Die Berechung wird wie folgt ausgeführt:

$$k_{x(min)} = \min_{i \in 1...n} k^i_{x(min)} \tag{5.6}$$

$$k_{y(min)} = \min_{i \in 1...n} k^i_{y(min)} \tag{5.7}$$

$$k_{x(max)} = \max_{i \in 1...n} k^i_{x(max)} \tag{5.8}$$

$$k_{y(max)} = \max_{i \in 1...n} k^i_{y(max)} \tag{5.9}$$

Höhen $k_{h(t)}, t\{\in basis, mittel\}$ beschreiben die Höhen der Basis- und Mittellinie der Komponente. Diese Werte sind für die Ermittlung der richtigen geometrischen Relation ebenfalls notwendig.

Zur Bestimmung der Basis- und Mittellinie für ein Buchstabe, werden zwei Faktoren r_1 und r_2 benötigt. Diese Faktoren bestimmen sich aus der Kategorie eines Symbols. Es gibt insgesamt vier Kategorien:

1. Das Einzelzeichen steht auf der Basislinie und erreicht die Höhe der Mittelline $(r_1 = 0, r_2 = 1)$. Der Fall liegt bei Zeichen wie "a", "c", "e" usw. vor.
2. Das Symbol hat einen Unterhaken $(r_1 = 0,5; r_2 = 1)$ wie z.B. bei "g", "y" usw.
3. Das Symbol hat einen Oberhaken, ist ein Großbuchstabe oder eine Zahl $(r_1 = 0,0; r_2 = 0,5)$ wie bei "h", "Σ", "A", "B", "0", "1" usw.
4. Das Symbol hat einen Ober- und Unterhaken $(r_1 = 0,3; r_2 = 0,7)$.

Mit diesen Faktoren lassen sich schließlich die Attribute $k_{h(basis)}$ und $k_{h(mittel)}$ des Elements berechnen.

$$k_{h(basis)} = k_{y(min)} + r_1(k_{y(max)} - k_{y(min)}) \tag{5.10}$$

$$k_{h(mittel)} = k_{y(min)} + r_2(k_{y(max)} - k_{y(min)}) \tag{5.11}$$

Die Daten dieser Attribute erlauben, zwischen Buchstaben zu unterscheiden, welche größen- bzw. positionsnormiert eine gleiche Erscheinung haben (z.B. "u" und "U", "p" und "P").

Die synthetischen Attribute werden aus den geometrischen Informationen der beteiligten Komponenten berechnet. Dabei werden jeweils die Höhenwerte der Basis- und Mittellinien gewichtet summiert und dem entsprechenden synthetischen Attribut zugewiesen. Die Gewichte sind von der angewendeten Produktion abhängig. Betrachtet man das Beispiel der Summenstruktur,

$$E \rightarrow \sum_{E_1} E_2,$$

so spielt die Komponente E_1 keine Rolle für die Berechnung der Mittel- und Basislinien der neuen Komponente E. Die Gewichte $w(k^i, q, i, t)$ werden entsprechend eingestellt. Die Einstellung geschieht zur Zeit manuell. Eine automatische Einstellung der Gewichte mit Hilfe von Trainingsdaten wurde aus Gründen der beschränkten Bearbeitungszeit nicht unternommen, verspricht aber eine leichte Verbesserung der Erkennugsleistung des Gesamtsystems. In der Tabelle 5.1 sind Belegegungen der Gewichte von Relationsbeispielen zusammengefaßt.

$$k_{h(t)} = \prod_i w(k^i, q, i, t) k^i_{h(t)}, t \in \{basis, mittel\}\,, \tag{5.12}$$

wobei $\sum_i w(k^i, q, i, t) = 1$ ist.

$w(k^i, q, i, basis)/w(k^i, q, i, mittel)$	$i = 1$	$i = 2$	$i = 3$	$i = 4$
$q = A_1 A_2$	0,5/0,5	0,5/0,5		
$q = A_1$ IND A_2	1,0/1,0	0,0/0,0		
$q = A_1^{A_2}$	1,0/0,8	0,0/0,2		
$q = \frac{A_1}{A_2}$	0,2/1,0	0,8/0,0		
$q = \sqrt[A_2]{A_3}$, $A_1 = \sqrt{\cdots}$	1,0/1,0	0,0/0,0	0,0/0,0	
$q = \sum_{A_2}^{A_3} A_4$, $A_1 = \sum$	0,6/0,6	0,0/0,0	0,0/0,0	0,4/0,4

Tabelle 5.1: Gewichtseinstellung zur Bestimmung der synthetischen Attributswerte der Höhenlinien

Variable k_V ist ein Wert, der auf eins gesetzt wird, falls die Komponente ein Variablensymbol besitzt, ansonsten wird er zurückgesetzt. Dieses Attribut ist hilfreich, um Produktionen zu dämpfen, bei denen eine Kombination mit einer Variable bzw. einer Zahl keinen Sinn macht. Ein Beispiel ist die Indizierung einer Zahl wie "6_2", die in mathematischen Ausdrücken selten vorkommt.

Belegung $k_{b(R)}$ Die Variablen $k_{b(R)}, R \in \{IND, EXP, FROM, TO, IN\}$ repräsentieren die Belegung von entsprechenden Relationen zur Hauptkomponente k. Diese Variablen sind deshalb nützlich, da eine Doppelbelegung dieser Satellitenpositionen syntaktisch unkorrekt ist. So soll z.B. eine Produktion $E \rightarrow E_1^{E_2}$ nicht zur Anwendung kommen, wenn das Element k^1 bereits einen Exponenten ($k^1_{b(EXP)} = 1$) besitzt.

Mit der Einführung der attributierten Grammatik vervollständigt sich die Version von 5.4.2 zu dem folgenden, entgültigen Ergebnis:

$$\begin{aligned} A &= (G, A, R) \\ G &= (X, V, S, P) \\ X &= \{A - Z, \\ &\quad a - z, \\ &\quad 0 - 9, \end{aligned}$$

$$
\begin{aligned}
& alpha - zeta, \\
& \sum, \prod, \int, =, <=, >=, <, >, (,), \infty, \cdots\} \\
V &= \{S, A, E, E_S, E_{SUMFROM}, C, \cdots\} \\
P &= \{S \overset{0,7}{\rightarrow} E \\
& S \overset{0,3}{\rightarrow} A \\
& A \overset{0,6}{\rightarrow} E = E \\
& A \overset{0,1}{\rightarrow} E >= E \\
& A \overset{0,1}{\rightarrow} E <= E \\
& A \overset{0,1}{\rightarrow} E > E \\
& A \overset{0,1}{\rightarrow} E < E \\
& E \overset{0,2}{\rightarrow} EE_S \\
& E \overset{0,3}{\rightarrow} E + E_S \\
& E \overset{0,2}{\rightarrow} E - E_S \\
& E \overset{0,3}{\rightarrow} E_S \\
& E \overset{0,15}{\rightarrow} C \\
& E \overset{0,1}{\rightarrow} C_E \\
& E_S \overset{0,1}{\rightarrow} C^E \\
& E \overset{0,05}{\rightarrow} C_E^E \\
& E \overset{0,1}{\rightarrow} \sum_{E_{SUMFROM}}^{E} E \\
& E \overset{0,1}{\rightarrow} E_{SUMFROM} E \\
& E \overset{0,1}{\rightarrow} (E) \\
& E_{SUMFROM} \overset{0,5}{\rightarrow} C \\
& E_{SUMFROM} \overset{0,5}{\rightarrow} C = E \\
& C \overset{0,02}{\rightarrow} a \\
& C \overset{0,02}{\rightarrow} b \\
& \cdots \\
& \} \\
A &= \{S, p, x(min), y(min), x(max), y(max), h(t), V, b(R)\} \\
R &= \left\{f_{ind}(A_B), f_{exp}(A_B^C), f_{sum}(A_B^C D), f_{concat}(ABC), \cdots\right\}
\end{aligned}
$$

5.4.4 Pruning-Verfahren

Die Anzahl der Elemente in einer Zelle steigt exponentiell mit der Höhe der Stufe in der Pyramide. Das liegt daran, daß viele Elementkombinationen der Anforderung der Produktionen genügen und so immer mehr Elemente generiert werden. Dieser Wachs-

tum erhöht allerdings den Berechnungsaufwand für das Füllen einer Zelle höherer Stufen. Um diese kombinatorische Ausuferung zu vermeiden, können unterschiedliche sogenannte "Pruning"-Verfahren angewendet werden. Alle Pruning-Verfahren haben gemeinsam, nur soviel Komponenten in einer Zeile zu löschen, daß eine von der Höhe der Stufe unabhängige Anzahl nicht überschritten wird. Gleichzeitig versuchen Pruning-Verfahren den Informationsverlust zu minimieren.

- Zum einen ist es möglich n Komponenten mit den höchsten Wahrscheinlichkeiten in einer Zelle weiterzuführen, während die restlichen Komponenten gelöscht werden. Dieser Ansatz wurde in dem hier vorgestellten System verwendet.
- Die Methode der n-besten Komponenten kann innerhalb einer Gruppe mit gleichen Nonterminalen ausgetragen werden. Das hat den Vorteil, daß auch selten vorkommende Nonterminale mit niedrigen Wahrscheinlichkeiten eine "Überlebenschance" haben. Dieses Pruning-Verfahren findet auch Anwendung in dem Viterbi-Algorithmus [Rya93] zum Parsen einer linearen Grammatik.

5.5 Einzelzeichenerkenner

Wie aus dem vorherigen Kapiteln der Diplomarbeit resultiert, benötigt der Parser zwei unterschiedliche Erkennungssysteme zur Durchführung einer Formelerkennung. Zum einen braucht das System einen Erkenner für Zeichen, zum anderen einen Erkenner für geometrische Relationen zweier Komponenten.

Die Segmentierung eines Ausdrucks auf Einzelzeichen geschieht mit Hilfe des Einzelzeichenerkenners. Dabei wird diejenige Segmentierung bevorzugt, welche die besten Hypothesen des Erkenners aller segmentierten Zeichen aufweisen kann. Während das Bewerten aller denkbaren Segmentierungen zu rechenaufwendig wäre, da es einen Aufwand von $O(n^n)$ verursacht (n ist die Anzahl der Eingabestrokes), ist eine Betrachtung aller Verknüpfungen von fünf aufeinanderfolgen Strokes pro Zeichen praktikabel.

Die Zusammenfassung von bis zu fünf Strokes und deren Zuführung zum Einzelzeichenerkenner wird vom Parseralgorithmus übernommen. Der Einzelzeichenerkenner berechnet aus diesen zusammengefaßten Strokes die spezifischen Attribute, welche im Kapitel 5.4.3 vorgestellt worden sind. Der Einzelzeichenerkenner basiert auf dem NPen System [MFW95a], welches im Kapitel 2.1 beschrieben worden ist. Er generiert zu einer Stifteingabe eine Lösungsmenge, die aus den Symbolhypothesen besteht. Der Umfang der Lösungsmenge des Einzelzeichenerkenners kann entweder durch eine vorgebene Zahl beschränkt werden oder die Hypothesen müssen ein vordefinierten Schwellwert überschrei-ten. Entsprechend des Umfangs der Lösungsmenge des Einzelzeichenerkenners werden neue Elemente in die CKY-Pyramide eingefügt und ihre Attribute k_S und k_p belegt. Die spezifischen Attribute eines Elements k bestehen nicht nur aus der Information des Symbols k_S und der dazugehörigen Hypothese k_p, sondern auch aus den Geometrieinformationen, der Einstellung des Zeichentypes k_V und der Satellitenbelegungen $k_{b(R)}$. Diese Attribute werden gemäß der folgenden Auflistung eingestellt.

$$k_S = z, z \in \{0-9, a-z, A-Z, \alpha-\omega, \text{etc.}\} \tag{5.13}$$

$$k_p = g(T, z) \tag{5.14}$$
$$k_{x(min)} = \min(T_x) \tag{5.15}$$
$$k_{y(min)} = \min(T_y) \tag{5.16}$$
$$k_{x(max)} = \max(T_x) \tag{5.17}$$
$$k_{y(max)} = \max(T_y) \tag{5.18}$$
$$k_V = \begin{cases} 1 & \text{falls } z \text{ Variable ist} \\ 0 & \text{sonst} \end{cases} \tag{5.19}$$
$$k_{b(R)} = 0, \tag{5.20}$$

wobei T die Stiftrajektorie der verknüpften Strokes darstellt, T_x bzw. T_y der Projektion der Trajektorie auf die x-Achse bzw. der y-Achse entspricht und die Funktion $g(T, z)$ eine Hypothese des Zeichens z bzgl. der Stroktrajektorie T berechnet.

Das mit den Attributswerten belegte Element wird nun entsprechend seiner Anfangsposition i der in Strokes untergliederten Gesamteingabe und der Anzahl der verwendeten Strokes j in die Zelle mit dem Index $(i, j-1)$ der CKY-Pyramide eingefügt.

5.6 Erkenner geometrischer Relationen

Der zweite Erkennungsalgorithmus des Systems ist der Erkenner geometrischer Relationen. Dieser dient dazu, festzustellen, in welchem geometrischen Verhältnis zwei Komponenten stehen.

Aus Gründen der Einfachheit schränkt sich das System auf 7 von den 13 möglichen Relationen ein, da die Ausdrücke des Datensammelprojekts nur aus diesen 7 geometrischen Relationen aufgebaut sind. Diese Relationen sind mit $\{LIN, UNDER, IND, EXP, FROM, TO, IN\}$ bezeichnet. Die Tabelle 5.2 listet die Relationstypen auf und beschreibt sie stichwortartig. Die Abbildung 5.4 zeigt Subkomponenten eines Ausdrucks mit ihren geometrischen Relationen an Hand eines Beispiels.

Die geometrischen Relationen können durch die Positionen der umrahmenden Rechtecke und den Grund- und Mittellinien der beiden Komponenten abstrahiert werden. Deswegen benutzt das System momentan vier aus diesen Daten abgeleitete Merkmalswerte (y, d, b, c), die jeweils das Merkmal Höhenverhältnis (y_F), horizontaler Abstand (d_F), Basisabstand (b_F) und Mittellinienabstand (c_F) repräsentiert. Diese Zahlen werden wie folgt aus der Komponente k und l bestimmt: (Die Abbildung 5.5 stellt die Konstruktion der Werte graphisch dar.)

$$y := \frac{l_{h(mittel)} - l_{h(basis)}}{k_{h(mittel)} - k_{h(basis)}} \tag{5.21}$$
$$d := \frac{l_{x(min)} - k_{x(max)}}{k_{x(max)} - k_{x(min)}} \tag{5.22}$$
$$b := \frac{l_{h(basis)} - k_{h(basis)}}{k_{h(mittel)} - k_{h(basis)}} \tag{5.23}$$
$$c := \frac{l_{h(mittel)} - k_{h(mittel)}}{k_{h(mittel)} - k_{h(basis)}} \tag{5.24}$$

Klassenname	Richtung	Größenverh.	Beschreibung
LIN	O	identisch	zweiter Ausdruck steht in der selben Linie wie der erste Ausdruck
UNDER	S	identisch	zweiter Ausduck steht unter dem ersten (für Bruchstriche, Vektoren, etc.)
IND	SO	kleiner	zweiter Ausdruck ist Index
EXP	NO	kleiner	zweiter Ausdruck ist Exponent
FROM	S	kleiner	zweiter Ausdruck ist Index, der unterhalb des ersten Zeichens steht (für Summationsgrenzen, etc.)
TO	N	kleiner	zweiter Ausdruck ist Index, der überhalb des ersten Zeichen steht
IN	-	kleiner	zweiter Ausdruck steht innerhalb des umklammernden ersten Ausdrucks (für Wurzelausdrücke)

Tabelle 5.2: Alle vom System unterstützten Relationen

Höhenverhältnis y: Das Höhenverhältnis speichert das Verhältnis der Schriftgrößen beider Komponenten. Ein Wert von y stark kleiner als 1 deutet darauf hin, daß Komponente l eine Satellitenkomponente von k ist.

Horizontaler Abstand d: Der horizontale Abstand entspricht der Distanz gemessen von dem rechten Rand der Ausgangskomponente k zum linken Rand der Komponente l. Diese Distanz ist durch die Breite der Komponente k normiert. Ein negativer Wert von d deutet auf eine "UNDER"-, "FROM"- oder "TO"-Relation hin.

Basisabstand b: Der Basisabstand ist der vertikale Abstand beider Basislinien. Dieser Wert ist normiert durch die Höhe von der Komponente k. Ein Wert von b stark negativ deutet z.B. auf einen Index hin, während $b > 1$ dafür spricht, daß Komponente l z.B. der Exponent von k ist.

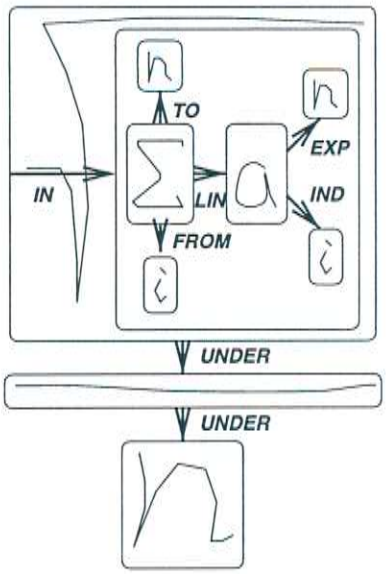


Abbildung 5.4: Die vom System unterstützen Relationen. Die Bedeutungen der Abkürzungen sind in der Tabelle 5.2 aufgelistet.

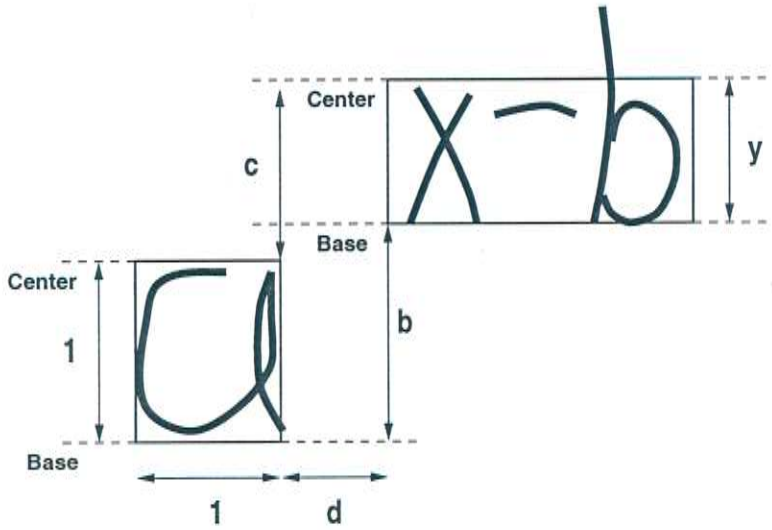


Abbildung 5.5: Die Konstruktion der für die Bestimmung der geometrischen Relation relevanten Merkmalswerte

Mittellinienabstand c Entsprechend der Basisdifferenz stellt der Wert c die Mittelliniendifferenz dar. Auch dieser Wert wird durch die Höhe von der Komponente k normiert.

Einsatz von Fuzzy-Logik zur Bestimmung der geometrischen Relationen:

Zur Bestimmung der geometrischen Relationen aus den oben beschriebene Merkmalswerten (y, d, b, c) werden Fuzzy-Methoden verwendet, welche in Form von Verknüpfungs-

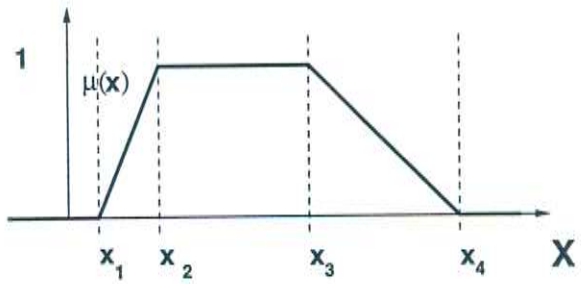


Abbildung 5.6: Die parametrisierten Mitgliedsgradfunktion

operationen vom Programmierer fest vorgegeben werden. Für jede geometrische Relation wird nun ein Konfidenzmaß ermittelt, indem die unscharfen Aussagenwerte für jede geometrische Relation mit den Fuzzy-Mengen berechnet werden.

Die Erkennung von geometrischen Relationen verwendet Fuzzy-Mengen des Types 4 der im Kapitel 3.4 vorgestellten Typen von Fuzzy-Mengen. Dabei werden Funktionen mit stückweise linearem Verlauf für $\mu(x)$ gewählt, die für $x = -\infty$ und $x = \infty$ auf null zurückfallen. Diese Art von Mitgliedsgradfunktionen lassen sich mittels vier Parametern $x_1 \leq x_2 \leq x_3 \leq x_4$ beschreiben. Der Graph des Funktionstypes wird in 5.6 gezeigt.

$$\mu(x) = \begin{cases} \frac{x-x_1}{x_2-x_1} & \text{für } x_1 \leq x < x_2 \\ 1 & \text{für } x_2 \leq x < x_3 \\ \frac{x_4-x}{x_4-x_3} & \text{für } x_3 \leq x < x_4 \\ 0 & \text{sonst} \end{cases} \tag{5.25}$$

Diese parametrische Funktion kann ebenfalls die Funktionstypen 1 und 2 simulieren, wenn man davon ausgeht, daß Werte aus dem Grundbereich nur in einem gewissen Intervall $[a; b]$ plausibel bzw. möglich sind. In diesem Fall setzt man die Parameter x_1, x_2, x_3, x_4 entsprechend außerhalb dieses Intervalls. Die Einstellung der Parameter jeder Fuzzy-Menge wird mittels einem einfachen Trainingsverfahren erzielt. Dabei werden alle Parameterkombinationen getestet, bei denen sich die Parameter in einem vom Programmierer vorgegebenen Intervall befinden. Für jede Einstellung wird ein Testdurchlauf gestartet und die Anzahl der Übereinstimmungen von erkannten und tatsächlichen Relationen gezählt. Nach Test aller Kombinationen wählt man die Parametereinstellung aus, welche die meisten Übereinstimmungen erbringen. Die ermittelten Werte der Parameter kann in der Tabelle 5.3 abgelesen werden. Die Graphen der Mitgliedsgradfunktionen der geometrischen Relation EXP sind in 5.7 dargestellt.

Es werden nun für jede der obigen Merkmale $F \in \{y_F, d_F, c_F, b_F\}$ und für jede der zu erkennenden Klasse $R \in \{LIN, UNDER, IND, EXP, FROM, TO, IN\}$ die Parameter der Mitgliedsgradfunktionen $\mu_{R,F}$ gesucht. Diese Mitgliedsgradfunktionen definieren die entsprechenden Fuzzy-Mengen $M_{R,F}$.

$$M_{R,F} = \int \mu_{R,F}/x \tag{5.26}$$

(5.27)

x_1, x_2, x_3, x_4	Y	D	B	C
LIN	0,0; 0,5; 3,5; 4,5	0,0; 0,1; 2,0; 2,3	-1,9; -1,8; 0,5; 1,0	-1,0; -0,5; 0,9; 2,0
$UNDER$	0,5; 0,6; 4,5; 6,5	-9,0; -8,0; -0,5; 0,5	-7,0; -6,0; 0,0; 0,5	-6,0; -5,0;-1,0;-0,5
IND	0,0; 0,1; 1,2; 1,4	-0,3; -0,1; 1,2; 1,5	-3,0; -3,0; 0,0; 1,0	-2,0; -2,0;-0,5; 0,0
EXP	0,0; 0,1; 1,3; 1,6	-0,3; -0,0; 1,1; 1,4	0,0; 0,5; 3,2; 3,8	0,0; 0,0; 0,2; 1,0
$FROM$	0,0; 0,1; 1,5; 2,1	-2,0; -1,1; -0,7; -0,5	-5,0; -4,0; 0,0; 0,5	-4,0 -3,0 0,0; 0,15
TO	0,0; 0,1; 1,5; 2,2	-2,0; -1,2; -0,5; -0,1	0,4; 1,0; 3,0; 5,0	-0,3; 0,0; 3,0; 3,5
IN	0,0; 0,1; 0,9 1,1	-1,0; -1,0; -0,1; 0,0	0,0; 0,0; 0,7; 0,9	-0,9; -0,8;-0,1; 0,0

Tabelle 5.3: Die trainierten Parameter der Mitgliedgradfunktionen

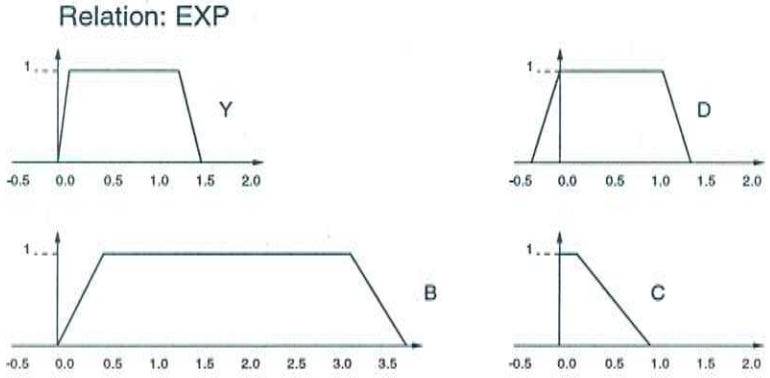


Abbildung 5.7: Die vier Mitgliedgradfunktionen für die geometrische Relation EXP

Um die Konfidenzmaße der geometrischen Relationen $\{LIN$, $UNDER$, IND, EXP, $FROM$, TO, $IN\}$ zu bestimmen, werden alle Fuzzy-Mengen mit übereinstimmenden Relationsindizes miteinander mittels einer t-Norm verknüpft. Die Wahl der t-Norm fiel auf die Minimumnorm, die im Vergleich zu den anderen Fuzzy-Normen alg, bes, dra höhere Erkennungsraten erzielt hat (siehe Abschnitt 6). Daraus ergibt sich die Berechnung wie folgt:

$$z_R = \min \{\mu_{R,v_F}(v) | v \in \{y, d, c, b\}\} \tag{5.28}$$

Nachdem für alle Relationen diese Bewertungen berechnet worden sind, sucht man den Maximalwert $Z^\star$ heraus und betrachtet die korrespondierende Relation $R^\star$ als das Resultat der Erkennung der geometrischen Relation der beiden Eingangskomponenten.

$$z^\star = \max_{r \in R} z_r \tag{5.29}$$

$$R^\star = \text{argmax}_{r \in R} z_r \tag{5.30}$$

5.7 Ausgabe

Nachdem in der obersten Zelle der CKY-Pyramide eine Menge von Komponenten mit dem Startsymbol generiert wurden ist ($k_S = S$), kann nun die Ausgabe entsprechend

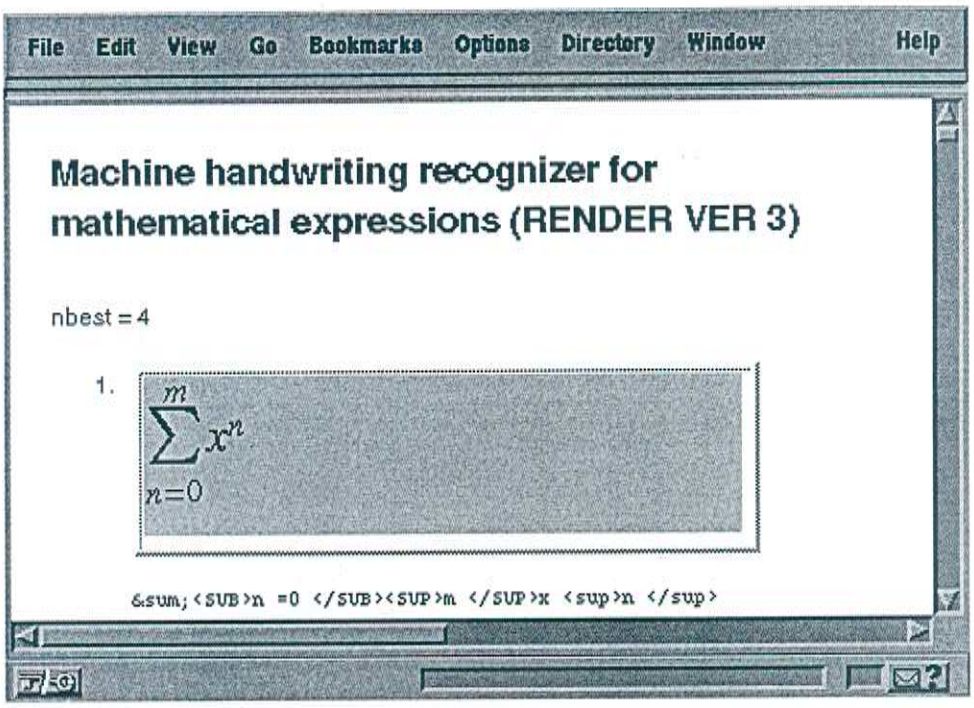


Abbildung 5.8: Ausgabe eines mathematischen Ausdrucks realisiert mit einem Java-Applett

ermittelt werden.

Für alle Elemente k, bei denen die Zugehörigkeitswahrscheinlichkeit k_p größer als null ist, wird eine Präfix-Notation erstellt. Dazu wird der Syntaxbaum beginnend von dem Startelement in der Spitze der Pyramide hinunter zum Boden verfolgt und eine Präfix-Notation berechnet. Dabei betrachtet man, welche Produktion bei der Generierung des Elements zur Auswahl kam, notiert diese und verfolgt dann rekursiv die Komponenten der Produktion bis schließlich die Terminale am Boden erreicht werden.

Aus dieser Präfix-Notation wird dann ein Skript der formalen Sprache Latex oder HTML produziert. Dieser Vorgang ist leicht ausführbar, da die Produktionen der kontextfreien Grammatik, die dem Parser zugrunde lag, dieser formalen Sprachen ähnlich sind. Mittels entsprechender Anzeigeprogramme für Latex (ghostview von Erik M. van der Poel) oder für HTML (WebEq vom Instititut für Mathematik, Hiroshima, Japan) ist es möglich die Ausdrücke leicht lesbar als Bild anzuzeigen. Es hat sich gezeigt, daß der Zeitbedarf zur Generierung eines Bildes nicht unerheblich ist (bis zu 4 Sekunden auf einer DEC Alpha 3000). Das Bild 5.8 zeigt die Interpretation der Stifteingabe von Bild 5.2.

Kapitel 6

Erkennungsergebnisse des Gesamtsystems

Trainings- und Testdaten des Einzelzeichenerkenners

Trainings- und Testbeispieldaten für den Einzelzeichenerkenner NPen stammen zum einen aus dem Datensammelprojekt (siehe dazu das Kapitel 4), zum anderen aus dem NPen Projekt selbst [MFW95a]. Diese NPen-Datenmenge ergibt sich aus mehreren Schreibern und besteht aus Beispielen vom Alphabet der lateinischen Klein- und Großbuchstaben, sowie Zahlen. Aus dieser Datenquelle wurden ca. 4000 Zeichen entnommen. Das Datensammelprojekt dieser Arbeit lieferte ca. 2000 mathematische Sonderzeichen, welche aus insgesamt 187 verschiedenen Symbolklassen stammten. Davon wurden 52 Symbolklassen verwendet.

Erkennungsleistung des Einzelzeichenerkenners

Zur Evaluation des Einzelzeichenerkennungssystems wurde nun eine zu der Trainingsmenge disjunkte Testmenge verwendet. Das System erreichte auf diesen Daten eine Erkennungsleistung von 88,3%. Die meisten Fehler entstanden bei der Verwechslung von Zeichen, die bereits in ihrer Form Ähnlichkeiten haben (wie z.B. "1", "(", ")"). Das System von Z. Wang [Wan88] löst das Problem der Ähnlichkeit damit, daß es diese Symbole in die gleiche Klasse abbildet und hofft, daß der Kontext des Symbols die Erkennung des richtigen Zeichens ermöglicht. Auch das hier vorgestellte System verwendet diese Methode. Ein Abfall der Erkennungsleistung gegenüber dem in [MFW95a] angegebenen Erkennungsergebnis von 97% kann auf den begrenzten Umfang der Trainingsmenge zurückgeführt werden.

Anzahl der Symbolklassen	Trainingsbeispiele	Testbeispiele	Erkennungsrate
52	4300	2000	88,3%

Tabelle 6.1: Ergebnisse des Einzelzeichenerkenners

Trainings- und Testmenge des Erkenners für geometrische Relationen

Das Datensammelprojekt lieferte 1200 mathematische Ausdrücke. Diese Ausdrücke wurde auf ca. 8500 Subkomponeneten segmentiert, um deren geometrischen Relationen zu notieren. Diese Relationen fallen in eine von 7 Relationsklassen $\{LIN, UNDER, IND, EXP, FROM, TO, IN\}$, welche in der Tabelle 5.2 beschrieben sind. Alle Beispieldaten waren von einer hohen Qualität. Abnorme Positionsverhältnisse der Komponenten zueinander waren nicht vorhanden.

Erkennungsleistung des Erkenners für geometrische Relationen

Das System erzielte mit 4000 Testdaten von 22 Schreiben eine Erkennungsleistung von $89,2\%$ bei 7 verschiedenen Relationsklassen. Dieses Ergebnis wurde mit der Einstellung der Fuzzy-Mengen, wie sie in Tabelle 5.3 zu sehen ist, erbracht. Außerdem wurde beobeachtet, daß die Minimumnorm als Fuzzy-Norm, mit dem die Merkmalswerte y, b, c, d zusammengefaßt wurden die besten Erkennungsergebnisse lieferte. Dabei standen vier t-Normen zum Test zur Verfügung, zum einen die algebraische Norm alg, dann die beschränkende und drastische Norm bes und dra und schließlich die Minimumnorm min. Die Tabelle 6.2 zeigt die Erkennungsergebnisse in Abhängigkeit der ausgewählten Norm.

t-Norm	Erkennungsleistung
min	$89,2\%$
alg	$86,5\%$
bes	$45,4\%$
dra	$26,2\%$

Tabelle 6.2: Erkennungsleistung der geometrischen Relationen in Abhängigkeit der verwendeten t-Norm

Die häufigsten Fehler entstanden bei der Verwechslung der Relationen $FROM$ und $UNDER$, IND und $FROM$ sowie TO und EXP. Diese Verwechslung wird aber durch den Parser teilweise eliminiert, da die Grammatik nicht alle geometrischen Kombinationen zuläßt. So wird z.B. ein Konflikt zwischen der möglichen Fehlklassifikation $UNDER$ einer Komponente im Summenkonstrukt aufgelöst, da die Grammatik mathematischer Ausdrücke nur eine Relation $FROM$ zwischen dem Summenzeichen $\sum$ und der darunterliegenden Komponente zuläßt. Eine tiefergehende Erörterung der Synergieeffekte der Grammatik und den Erkennungssystemen wird im Kapitel 6.3 ausgeführt.

Anzahl der Relationsklassen	Trainingsbeispiele	Testbeispiele	Erkennungsrate
7	4500	4000	$89,2\%$

Tabelle 6.3: Erkennungsergebnisse geometrischer Relationen

6.1 Fehlermaß

Zur Berechnung der Leistungsfähigkeit des Gesamtsystems wurde die auch in der kontinuierlichen Spracherkennung bekannten Methode der Worterkennungsrate verwendet. Die Worterkennungsrate berechnet sich aus der Anzahl von Löschungen D, Einfügungen I und Ersetzungen S von Zeichen und aus der vorgegebene Referenz mit R Zeichen und Relationen in folgender Weise:

$$\mathrm{WA} = 100\% \left(1 - \frac{D+I+S}{R}\right). \tag{6.1}$$

Dabei wird eine Relation als Wort angesehen. Erkannte das System z.B. den Ausdruck "a IND b" aus der Referenz "a EXP b" so ist das Fehlermaß $100\% \left(1 - \frac{0+0+1}{3}\right) = 75\%$. Der Unterschied zur einer einfachen Gleichheitsüberprüfung liegt also darin, daß bei der Worterkennungsrate differenzierte Fehlermaße zwischen 0% und 100% entstehen, während eine einfache Gleichheitsüberprüfung 0% als Fehlermaß liefert, auch wenn nur ein Zeichen in einem komplexen Ausdruck mit vielen Zeichen fehlerhaft klassifizert wurde.

6.2 Erkennungsleistung

Die Beispieldaten zur Ermittlung der Erkennungsleistung des Gesamtsystems stammten aus dem Datensammelprojekt. Es wurden bei 1200 Beispielen, die richtige Transkription erfaßt, um das Erkennungsergebnis eines Ausdrucks mit seiner Referenz zu vergleichen. Diese Daten stammten von 22 Schreibern und besaßen unterschiedliche Schreibstile.

Die Erkennungsleistung ist von der Anzahl der Einzelzeichen in einem Ausdruck abhängig. Im Durchschnitt aller 1200 Ausdrücke sind ca. fünf Zeichen in einem Ausdruck. Bei diesem Zeichenumfang erreicht das System eine Erkennungsleistung von 66, 2%. Es stellt sich heraus, daß die Fehlerrate mit der Anzahl der Zeichen im Ausdruck zunimmt. Das liegt vor allem daran, daß die Anzahl der Kombinationen ebenfalls wächst, in der man die Eingabe als Ausdruck interpretieren kann. Während die Tabelle 6.4 die gemessenen Erkennungsleistungen in Abhängigkeit der Zeichenanzahl im Referenzausdruck zeigt, untergliedert die Tabelle 6.5 Erkennungsergebnisse bzgl. der 22 Schriftspender, die am Datensammelprojekt teilgenommen haben.

Der im Rahmen seiner Doktorarbeit entstandende maschinelle Erkenner für handgeschriebene, mathematische Ausdrücke von H.-J. Winkler [Win95b] erreicht bei Eingaben mit ca. 40 Zeichen eine Erkennungsleistung von 88% und erzielt damit eine bessere Erkennungsrate als das hier vorgestellte System. Diese Überlegenheit in der Erkennungsleistung kann auf einen größeren Umfang von zur Verfügung stehenden Trainingsdaten für mathematischen Sonderzeichen, welche zudem aus einer kleineren Gruppe von 7 Schriftspendern stammen, zurückgeführt werden. Desweitern kommt bei dem System von Winkler ein Parseralgorithmus für lineare Grammatiken zum Einsatz, was zu Reihenfolgenrestriktionen in der Eingabe von Subkomponenten mathematischer Ausdrücke für den Benutzer führt. In dem System dieser Diplomarbeit wird ein Parser für kontext-freie Sprachen verwendet, der bei gleichem Ausdruck häufiger auf Entscheidungssituationen trifft und somit mehr Fehler machen kann. Der Vorteil jedoch liegt darin, daß eine höhere Flexibilität bei der Eingabe geboten werden kann.

Berechnungszeiten

Das System basiert auf einer Methode, welche zur Berechnung $O(n^{m+1})$ Schritte benötigt, wobei n die Anzahl der Eingabestrokes und m die maximale Produktionslänge darstellt. Die in diesem System verwendete Grammatik besitzt eine maximale Produktionslänge von vier Symbolen. Die Berechnungszeiten können in vier Teile untergliedert werden: Eingabe, Berechnung der Hypothesen der Einzelzeichen, Parsen mittels des CKY-Algorithmus und schließlich der Ausgabe.

Die Tabelle 6.6 listet die benötigte Zeitdauer auf, sortiert nach der Anzahl der verwendeten Strokes in der Eingabe und den Programmabschnitten.

6.3 Zusammenspiel der Grammatik mit den Erkennungssystemen

Eine interessante Fragestellung ist, inwieweit der Parser mit Hilfe der probabilistischen Grammatik in der Lage ist, Fehlklassifikationen des Einzelzeichenerkenners und des Erkenners geometrischer Relationen zu korrigieren. Um diese Frage zu beantworten, wurden zu allen Testbeispielen, während der Generierung des Syntaxbaumes, die Ergebnisse der beiden Erkennungssysteme notiert und mit der Entscheidung des Parsers, welcher von der probabilistischen Grammatik gesteuert wird, verglichen.

Es stellte sich heraus, daß durch den Einsatz der probabilstischen Grammatik eine Fehlklassifikation des Erkenners für geometrische Relationen zu 85% zum korrekten Ergebnis verbessert wurde. Allerdings war der Parser nur in 12% aller Fälle in der Lage, eine Fehlklassifikation des Einzelzeichenerkenners zu korrigieren

Zusätzlich wurde beobachtet, daß der Parser durch die probabilistische Grammatik korrekte Klassifikationen des Relations- und Zeichenerkenners in der Art benachteiligt hat, daß sie beim Pruning verloren gingen. Die Tabelle 6.7 zeigt die Synergieeffekte zwischen Grammatik und Erkennungsalgorithmen. Insgesamt steigerte der Einsatz einer probabilisitschen Grammatik die Erkennungsleistung um $9,8\%$ im Gegensatz zur Verwendung einer Grammatik, bei der die Produktionswahrscheinlichkeiten ungewichtet (gleichverteilt) eingestellt sind.

Einzelzeichen im Ausdruck	Anz. der Testmenge	WA
10 und mehr	130	43,4%
7 – 9	332	57,2%
3 – 6	401	69,6%
1 – 2	383	77,9%
durchschnitt. 5	1246	66,2%

Tabelle 6.4: Erkennungsleistung des Systems in Abhängigkeit der Zeichenanzahl im Referenzausdruck

Schreiber ID	Erkennungsleistung
70	76%
71	75%
200	53%
298	51%
299	69%
300	71%
301	75%
302	54%
304	86%
305	68%
306	72%
307	54%
308	37%
309	59%
310	67%
311	60%
312	81%
313	74%
314	72%
315	67%
329	52%
400	69%

Tabelle 6.5: Erkennungsleistung in Abhängigkeit unterschiedlicher Schriftspender

Strokes	Eingabe (ca. sec)	Einzel-zeichenerkenner (ca. sec)	CKY-Algorithmus (ca. sec)	Ausgabe (ca. sec)	total ohne Eingabezeit (ca. sec)
1	1,2	0,5	0,1	2,0	2,6
2	2,4	1,5	0,3	2,0	3,8
3	3,6	3,0	0,6	2,0	5,6
4	4,8	5,0	1,0	3,0	9,0
5	6,0	7,5	1,5	3,0	12,0
6	7,2	10,0	2,4	3,0	15,4
7	8,4	12,5	4,5	4,0	21,0
8	9,6	15,0	6,4	4,0	25,4
9	10,8	17,5	8,1	4,0	39,6
10	12,0	20,0	12,0	4,0	36,0
11	13,2	22,5	15,2	5,0	42,7
12	14,4	25,0	20,2	5,0	50,2

Tabelle 6.6: Berechnungszeit des Systems, unterteilt in seine Teilmodule (gemessen auf DEC Alpha 3000)

probabilistische Grammatik	Einzelzeichen-erkenner	Relations-erkenner	gesamt
Korrektur	15,9%	6,9%	22,8%
Fehlentscheidung durch Grammatik	11,7%	1,5%	13,2%
Verwendungsvorteil	4,2%	5,4%	9,6%

Tabelle 6.7: Korrektur und Fehlentscheidung durch Einsatz probabilistischer Grammatik

Kapitel 7

Zusammenfassung und Ausblick

7.1 Zusammenfassung

Es wurde ein Erkenner für mathematische Ausdrücke vorgestellt, der mit einer probabilistischen, attributierten Grammatik arbeitet. Der Einzelzeichenerkenner des Systems wurde mittels eines MS-TDNN Moduls realisiert, während der Erkenner für geometrische Relationen Fuzzy-Mengen zur Klassifikation verwendete. Das System ist in der Lage, handgeschriebene, mathematische Ausdrücke in einen entsprechenden Syntaxbaum zu konvertieren. Dabei erreicht das System bei durchnittlich 5 Zeichen in einem Ausdruck eine Worterkennungsrate von 66, 2% auf schreiberunabhängigen Testdaten.

Ein auf Stift und drucksensitiven Bildschirm basierender Editor ist mit Hilfe des Erkennungssystems dieser Diplomarbeit in der Lage, dem Benutzer eine intuitive und schnell erlernbare Eingabemöglichkeit für mathematische Ausdrücke anzubieten. Diese Einfachheit der Benutzung kommt der Eingabegeschwindigkeit zu Gute und erweitert dadurch einen Einsatz von Formeln in Entwicklungs- und Anwendungsprogrammen.

7.2 Ausblick

Ein System dieser Art wird immer Stellen bieten, an denen geschickte Veränderungen zu Verbesserungen führen. Während der Implemtierung des Verfahrens, sind folgende Verbesserungsmöglichkeiten zur Erkennungsrate, zur Berechnungsgeschwindigkeit und zum Eingabekomfort aufgefallen, die aber in Anbetracht der begrenzten Bearbeitungszeit für das Thema nicht realisiert werden konnten. Diese Verbesserungsvorschläge können Grundlage für weiterführende Arbeiten sein.

- Die Erkennungsleistung könnte durch Verwendung von mehr Trainingsdaten erhöht werden.
- Eine Befreiung der Restriktion, daß Zeichengrenzen auch Strokegrenzen sein müssen, führt zu einem Programm, bei dem letzlich auf den Viterbialgorithmus des NPen Systems verzichtet werden kann. Dabei sind die atomaren Bausteine des mathematischen Ausdrucks nicht mehr die Strokes, sondern die Merkmalsvektoren der Ausgabe des Time Delay Neural Network (TDNN). Diese Umstellung würde zu einem monolytischen Gesamtsystem führen.

- Eine Umwandlung der verwendeten normalen kontext-freien Grammatik für mathematische Ausdrücke zu einer Chomsky-Normalform würde den Aufwand des Parsens im O-Kalkül von jetzt $O(n^{m+1})$ auf $O(n^3)$ beschränken, wobei n die Anzahl der Strokes und m die maximale Produktionslänge ist.
- Eine inkrementelle Realisierung der Erkennung der Einzelzeichen (Run-on Erkennung), sowie ein inkrementeller Aufbau des Parsers würde die Berechnungszeit erheblich verkürzen.
- Der Einsatz von anderen Parseralgorithmen (Earley [Ear68], Final State Automata, Phoenix [War91], etc.) könnte eine Verbesserung im Berechnungsaufwand ergeben.
- Eine Benutzung von einfacheren aber schnelleren Einzelzeichenerkennern könnte Rechenzeitvorteile erzielen.
- Eine Einbindung von Handschrift-Korrekturfunktionen würde die Bedienungsfreundlichkeit erhöhen.
- Der Einsatz von anderen Eingabearten verspricht eine Erhöhung des Eingabekomforts und könnte die Eingabe mathematischer Ausdrücke zusätzlich erleichtern. Die Eingabe könnte z.B. per Mikrofon geleistet werden. Dazu ist keine große Umstellung des momentanen Systems notwendig. Anstelle eines Einzelzeichenerkenners kann ein Einzelwort-Spracherkenner [Hil96] eingesetzt werden.
- Eine Einbindung zusätzlicher geometrischer Relationen, wie die Erkennung von Zwischenraumgrößen, Tabellenformen für Matrizen usw. und der Ausdehnung des Wortschatzes auf chemische Reaktionsgleichungen oder spezielle Notationen in der Elektrotechnik, Physik und Informatik, würde die Ausdrucksmöglichkeiten des Benutzers erweitern.

Anhang A

Sonderzeichen und Ausdrücke

Im Datensammelprojekt wurden Spender gebeten, mathematische Zeichen und Ausdrücke mit einem Aufzeichnungsprogramm zu schreiben. Es wurden keine Restriktionen im Schreibstil verlangt, so daß möglichst eine natürliche Eingabe erzielt werden konnte.

A.1 Sonderzeichen

Die folgenden Abschnitte zeigen die Sonderzeichen und Ausdrücke, die den Spendern zur Kopie vorgelegt wurden.

Lateinische Kleinbuchstaben: a b c d e f g h i j k l m n o p q r s t u v w x y z

Lateinische Großbuchstaben: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Zahlen: 0 1 2 3 4 5 6 7 8 9

Griechische Kleinbuchstaben: α β γ δ ϵ ε ζ η θ ϑ κ λ μ ν ξ π ρ ϱ σ τ υ ϕ φ χ ψ ω

Griechische Großbuchstaben: Γ Δ Θ Λ Ξ Π Σ Υ Φ Ψ Ω

Wörter: if then else

Funktionsnamen: sin cos tan cot arcsin arccos arctan sinh cosh tanh coth arcsinh arccosh arctanh arg dim lim Pr det max min sup inf exp log lg ln $\sqrt{\ }$

Mathematische Sonderzeichen: ∞ $\emptyset$ def

Einstellige Operationen: $\forall$ $\exists$ $\neg$ $\int$ $\oint$ ∂ ∇ $\angle$

Zweistellige Operationen: $- + \pm \mp \setminus \times / \circ * \equiv \models = \neq := < > \leq \geq$
$\ll \gg \approx \sim \subset \supset \subseteq \supseteq \in \ni \vee \wedge \lhd \rhd$

Pfeile: $\Leftrightarrow \Leftarrow \Rightarrow \leftrightarrow \leftarrow \rightarrow \mapsto$

Begrenzungszeichen: $\{ \} [ ] \frown \smile \| ( )$

A.2 Ausdrücke

In diesem Abschnitt wird ein Teil der insgesamt 400 verschiedenen mathematischen Ausdrücke aufgelistet, welche aus dem Bereich der Algebra, Analysis, Numerik, Statistik und der Informatik, bzw. Aussagenlogik stammen.

$$y = \sqrt{ax + b}$$
$$A \sin(\omega x + \varphi_0)$$
$$\operatorname{arcsinh}(x) = \ln\left(x + \sqrt{x^2 - 1}\right)$$
$$M = \{(a, b) | F(a, b) = 0\}$$
$$K = \{P(x(t), y(t)) | t \in I\}$$
$$-\infty < t < \infty$$
$$\frac{\pi}{2} \approx 1.571$$
$$\pi \approx 3.141$$
$$|a_r - z| \leq 0.5 * 10^{-r}$$
$$\triangle a \geq |a - z|$$
$$n! \approx \left(\frac{n}{e}\right)^n \sqrt{2\pi n}$$
$$\Gamma(z) \stackrel{\text{def}}{=} \int_0^\infty e^{-t} t^{z-1}$$
$$\Gamma\left(\frac{1}{2}\right) = \sqrt{\pi}$$
$$\sum_{i=1}^{n} a_i = \sum_{k=1-r}^{n-r} a_{k+r}$$
$$1 + 2 + \cdots + n = \frac{n(n+1)}{2}$$
$$m_A = \frac{1}{n}(a_1 + \cdots + a_n)$$
$$m_Q = \sqrt{\frac{1}{n}(a_1^2 + \cdots a_n^2)}$$
$$m_G = \sqrt{ab}$$
$$a^2 > 0$$
$$|a + b| \leq |a| + |b|$$
$$|a_1 + \cdots + a_n| \leq |a_1| + \cdots + |a_n|$$
$$|a| + |b| \geq |a - b| \geq |a| - |b|$$
$$\frac{a_1 + a_2 + \cdots + a_n}{n} \geq \sqrt[n]{a_1 a_2 \cdots a_n}$$
$$U = [U, +, *]$$
$$\emptyset \neq U \subseteq V$$
$$\|x\| \geq 0$$
$$\left\|\vec{a} + \vec{b}\right\| \leq \|\vec{a}\| + \left\|\vec{b}\right\|$$
$$\|\alpha \vec{x}\| = |\alpha| \, \|\vec{x}\|$$
$$\vec{s} = \begin{pmatrix} 12 \\ a(t) \\ b \end{pmatrix}$$

$$\cos\phi = \frac{(\vec{x},\vec{y})}{\|\vec{x}\|\|\vec{y}\|}$$
$$b_k = a_k = \sum_{i=0}^{k-1} \frac{(a_k,b_i)}{(b_i,b_i)} b_i$$
$$U \cap U^{\perp} = \emptyset$$
$$\left(U^{\perp}\right)^{\perp} = U$$
$$c = \begin{pmatrix} c_{11} & \cdots & c_{1m} \\ \vdots & & \vdots \\ c_{n1} & \cdots & c_{nm} \end{pmatrix}$$
$$d_B = \det B = \begin{vmatrix} b_{11} & \cdots & b_{1n} \\ \vdots & & \vdots \\ b_{n1} & \cdots & b_{nn} \end{vmatrix}$$
$$\det A = \prod_{i=1}^{n} \lambda_i$$
$$\sum_{i=0}^{n} c_i A^i = \vec{0}$$
$$e = \lim_{n\to\infty} \left(1 + \frac{1}{n}\right)^n$$
$$m\left(t\right) = m_o e^{-\lambda t}$$
$$\tanh x = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$
$$\sinh \frac{x}{2} = \sqrt{\frac{1}{2}\left(\cosh x + 1\right)}$$
$$\alpha + \beta + \gamma = 180°$$
$$x = \frac{\sum m_i x_i}{\sum m_i}$$
$$|MK| = \left|\overline{MF}\right|$$
$$[a,b) = \{x | x \in R, a \leq x < b\}$$
$$U_{\varepsilon}\left(P_0\right) = \{P | d\left(P,P_0\right) < \varepsilon\}$$
$$\sup\{a_n\} = \lim_{n\to\infty} \sup a_n = \overline{\lim_{n\to\infty} a_n}$$
$$\nabla = \frac{\partial}{\partial x}$$
$$\int_a^b \sqrt[n]{x} dx = \frac{1}{n\sqrt[n]{x^{n-1}}}\Big|_a^b$$
$$3\% = \frac{3}{100}$$
$$\delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{else} \end{cases}$$
$$\phi \gg \nu$$
$$N_{\mu,\sigma}$$
$$\dim\left(\Phi\right) = 0 \Rightarrow \dim\left(\Psi\right) = 0$$
$$R = 8\Omega$$
$$\Pr\left(\theta|\vartheta\right) = \frac{\Pr(\vartheta|\theta)\Pr(\theta)}{\Pr(\vartheta)}$$
$$\exists_1 p \in \Lambda$$
$$\underbrace{\overbrace{1+2}^{=3}+3}_{=6}$$
$$\overline{g \wedge h} \Leftrightarrow \neg g \vee \neg h$$
$$M \models \emptyset$$
$$8^{\frac{1}{2}}$$
$$a, b, c$$
$$\sqrt{\sqrt{x}+2}$$
$$\lceil x|y \rceil$$
$$\langle x|y \rangle$$

$$B = \begin{vmatrix} a & b \\ c & d \end{vmatrix}$$

$$\frac{x-x_1}{a} = \frac{y-y_1}{b}$$

$$A \times \vec{b} = \vec{c}$$

$$M = \left\{ \tfrac{1}{n} | n \in N \setminus \{0\} \right\}$$

$$\lim_{n \to \infty} \sqrt[n]{a} = 1$$

$$f : R \mapsto R$$

$$g^{-1}(y) = D$$

$$0 < |x| < \delta$$

$$\lim_{x \to x_0} \frac{f'(x)}{g'(x)} = C$$

$$g(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$$

$$V\,(f, Z)$$

$$i := \sqrt{-1}$$

$$1 = e^{i\pi/2}$$

$$a^n = |a|^n \, e^{in\phi}$$

$$\xi = \tfrac{\alpha}{1+\alpha^2+\beta^2}$$

$$\eta = \tfrac{\beta}{1+\alpha^2+\beta^2}$$

$$\int \tfrac{\partial u}{\partial x} dy$$

$$\tfrac{1}{2\pi i} \oint \tfrac{f(\zeta)}{\zeta - z} d\zeta$$

$$|a| = \sqrt{\alpha^2 + \beta^2}$$

$$\cos\phi + i\sin\phi = e^{i\phi}$$

$$f \circ g \stackrel{\text{def}}{=} (x, y)$$

$$(f \circ g)^{-1} = g^{-1} \circ f^{-1}$$

$$a \times b$$

$$x_{1/2} = \pm\sqrt{2}$$

$$G \cup H$$

$$l \lhd m \lhd n$$

$$1 \prec 2 \prec 3$$

$$\chi^1_{ab} = \tau^1_{ba}$$

$$K \subset \{1, 2, 3\}$$

$$1 \in K$$

$$\mu \succeq \nu$$

$$P_{m_2} \subseteq P_{m_1}$$

$$\angle x, y = \psi$$

$$\prod_n \tfrac{1}{n^2} = \max\left(a_k, a_{k-1}\right)$$

$$\prod_{k=1}^{\infty} k = \infty$$

$$\sum_n \sum_m f(n, m)$$

$$r \approx \pi/2$$

$$y''(t)$$

$$K = -\gamma \tfrac{m_1 m_2}{r_{12}^3} r$$

$$E = mc^2$$

$$\sqrt{\tfrac{t_h a + n^k}{y(o_u)}}$$

Anhang B

Unipen-Format

Zum Speichern von mathematischen Ausdrücken der Stifttrajektorien wurde das Unipen-Datenformat (siehe `http://hwr.nici.kun.nl/unipen`) gewählt. Der erste Anstoß zur Entwicklung des Unipen-Formats wurde auf der 11. Internationalen Konferenz IAPR für Mustererkennung im September 1992 von Professor Rejoun Plamondon gegeben. Im Mai 1993 wurde das Projekt gegründet, welches zur Aufgabe hat, handgeschriebene Stifttrajektorien für Forschungszwecke im Bereich der Handschriftenerkennung anzubieten. Dabei soll die Datenbank alle erdenklichen Bereiche handschriftlicher Eingaben abdecken. Insbesondere Daten für Buchstaben, Worte, Texte und Formeln sind verfügbar, letztere jedoch nur in geringem Umfang. Die Schriftdaten sind teilweise segmentiert und für eine Evaluation von Erkennungssystemen geeignet. Der Zugriff auf die Unipen-Datenbank ist all denjenigen gestattet, die zuvor eine Spende von segmentierten Daten für die Datenbank geleistet haben.

Neben der Stifttrajektorie, und ihrer Segmentierung zu Subkomponenten ist das Format in der Lage, Schreiberinformationen, wie z. B. das Alter oder Geschlecht, Herkunft oder allgemeiner Schreibstil, anzubieten. Dadurch ist eine gezielte Selektion für Test- und Trainingsdaten möglich.

Mit dem Befehl

```
.SEGMENT HIERARCHY DELINEATION QUALITY LABEL
```

ist es möglich, eine Trajektorie in der Stiftsequenz zu segmentieren. Dabei ist `HIERARCHY` die Bezeichnung des Types von der Domäne, aus der das Label stammt. So beschreibt z.B. `WORD` oder `TEXT`, ob der segmentierte Abschnitt ein Wort oder ein Text ist. Für den Bereich der mathematischen Ausdrücke wurde das Wort `FORMULAE` reserviert.

Das zweite Argument `DELINEATION` beschreibt die Position des Segments innerhalb der Trajektorie. `QUALITY` ist Platzhalter für die Beschreibung der Qualität des Segments und `LABEL` ist die Referenz des Segments.

Vorschlag zur Befehlssatzerweiterung

Das Format erlaubt, eigene Kommandos zu spezifizieren.

So kann man zur Beschreibung geometrischer Relationen zweier Segmente folgenden neuen Befehl in die Unipen-Sprache einfügen.

```
.RELATION RELATION QUALITY HIERARCHY DELINEATION HIERARCHY
DELINEATION ,
```

Der neue Unipen-Befehl `.RELATION` hat sechs Parameter. Zunächst beschreibt `RELATION` den Typ der geometrischen Relation der Segmente zueinander. Dafür stehen reservierte Schlüsselworte `LIN`, `UNDER`, `IND`, `EXP`, `FROM`, `TO` und `IN` zur Verfügung. Dabei stehen diese Worte für Relationen, wie sie in der Tabelle 5.2 beschrieben sind. Der `QUALITY` Wert erklärt die Qualität des Datenbeispiels. Die Parameter `HIERARCHY` und `DELINEATION` sind doppelt vorhanden. Für jedes der beiden Segmente gibt es Parameter `HIERARCHY` zur Beschreibung der Schriftthierachien (z.B. `WORD`, `TEXT`, etc.) und zur Lokalisierung `DELINEATION` des Segmentes innerhalb der Trajektorie.

Anhang C

Flugblatt zum Aufruf einer Datenspende

Um Studenten auf das Datensammelprojekt aufmerksam zu machen, wurden an ausgewählten Instituten Flugblätter verteilt. Aufgrund dieser Flugblätter meldeten sich 22 Studenten, die bereit waren, Schriftdaten zu spenden.

Do you want to earn $5 easily?

Can you copy this with a pen?

$$\Phi = 2\sqrt{\frac{E}{\mu}}$$

Then come to WeH 4616 (Wean Hall Interact Lab) at CMU where a friendly crew will welcome you. Here you can donate samples of mathematical formulas handwritten by you.

Don't worry! You don't have to understand them, you just have to copy them.

After we record your handwritten math expressions, which takes approximately one hour, **you get $5**. So don't wait! And **bring your friends** to WeH 4616 (Wean Hall Interact Lab on the 4th floor) and just say "here are my hands".

Our recording hours are flexible.
Contact Alex Schulz by e-mail at schulz@cs.cmu.edu or
call 268-3805 to set up a time that is convenient for you.

This project is sponsored by the Language Techonlogy Institute of Carnegie Mellon University, 1997.

Abbildung C.1: Flugblatt zum Aufruf einer Datenspende

Literaturverzeichnis

[AU72] A.V. Aho and J. Ullman. *The Theory of Parsing, Translation and Compiling.* Prentice-Hall Inc., vol. 1 edition, 1972.

[BBNN93] E.J. Bellagarda, J.R. Bellagarda, D. Nahamoo, and K.S. Nathan. A probabilistic framework for on-line handwriting recognition. *3rd Workshop on Frontiers in Handwriting Recognition*, pages 225–234, May 1993.

[Bra94] Linda Branagan. *The Frame handbook : building framemaker documents that work.* O'Reilly & Associates, 1994.

[Cho56] N. Chomsky. *Three Models for the Description of Language.* IRE, 1956.

[DH73] Richard O. Duda and Peter E. Hart. *Pattern Classification.* Wiley-Interscience Publication, 1973.

[Doe91] David S. Doermann. *Recovery of temporal information from static images of handwriting.* University of Maryland at College Park. Center for Automation Research. Computer Vision Laboratory, College Park, Md., 1991.

[Ear68] Jay Earley. *An efficient context-free parsing algorithms.* Carnegie-Mellon University, 1968.

[Fon92] Thomas Fontaine. Character recognition using a modular spatiotemporal connectionist model. *University of Pennsylvania. School of Engineering and Applied Science. Dept. of Computer and Information Science*, 1992.

[Hei95] Maritta Heisel. *YEAST : a formal specification case study.* Berlin : Technische Universitat Berlin, Fachbereich 13, Informatik, [1995], 1995.

[Hil96] Hermann Hild. Recognition of spelled names over the telephone. In *ICSLP 96*, 1996.

[Hop96] Dirk Hopf. *Editor zur handschriftlichen Eingabe zweidimensionaler Formeln in den Computer.* Universität Saarbrücken, 1996.

[Hue97] Wolfgang Huerst. *Masterthesis: Repair on On-line Handwritting Recognition.* Carnegie Mellon University, Universität Karlsruhe, (TH), 1997.

[Hus97] Kamran Husain. *JavaScript developer's resource : client-side programming using HTML, Netscape plug-ins and Java applets / Kamran Husain, Jason Levitt.* Upper Saddle River, NJ : Prentice Hall, c1997., 1997.

[Inc95] MathWorks Inc. *The student edition of MATLAB : version 4 : user's guide.* Englewood Cliffs, NJ : Prentice Hall, 1995.

[Kas76] U. Kastens. *Ein Übersetzer-erzeugendes System auf der Basis attributierter Grammatiken.* Universität Karlsruhe, Bericht 10/76, 1976.

[Kas80] U. Kastens. *Ordered Attributed Grammars.* Universität Karlsruhe, 1980.

[KF88] G.J. Klir and T.A. Folger. *Fuzzy Sets, Uncertanty, and Information.* Prentice Hall, 1988.

[Knu68] N. Knuth. Semantics of context-free languages. *Math. Systems Theory*, Vol. 2:127–145, 1968.

[Kop93] Helmut Kopka. *A guide to LATEX : document preparation for beginners and advanced users / Helmut Kopka and Patrick W. Daly.* Wokingham, England ; Reading, Mass. : Addison-Wesley, 1993., 1993.

[Kos95] M. Koschinski. Segmentation and recognition of symbols within handwritten mathematical expressions. In *Conference Proceedings*, volume 4 of *IEEE, ICASSP-95*, pages 2439–2442, Piscataway, NJ USA, 1995. Institute for Human-Machine Communication, University of Technology, Munich, Germany.

[MAK88] Robert Moll, Micheal A. Arbib, and A.J. Kfoury. *An Introduction to Formal Language Theory.* Springer, 1988.

[MFW95a] Stefan Manke, Micheal Finke, and Alex Waibel. Npen ++: A writer independent, large vocabulary on-line cursive handwriting recognition system. In IEEE Computer Society, editor, *Proceedings of the International Conference on Document Analysis and Recognition*, 1995.

[MFW95b] Stefan Manke, Micheal Finke, and Alex Waibel. The use of dynamic writing information in a connectionistic on-line cursive handwriting recognition system. In MIT Press, editor, *Advance in Neural Information Processing 7*, Cambridge(MA), 1995.

[MG84] William M.Waite and Gerhard Goos. *Compiler Construction.* Springer-Verlag, 1984.

[Nic96] Roy A Nicolaides. *MAPLE : a comprehensive introduction.* Cambridge, Eng., 1996.

[Ous94] John K. Ousterhout. *Tcl and the Tk toolkit.* Addison-Wesley, 1994.

[Rub91] Dean Harris Rubine. *The Automatic Recognition of Gestures.* PhD thesis, Carnegie Mellon University, 1991.

[Rya93] Matthew S. Ryan. *The Viterbi algorithm.* PhD thesis, Warwick, England : University of Warwick, Dept. of Computer Science, Warwick, England, 1993.

[Sen92] S. Seneff. Tina: a natural language system for spoken language applications. In *Computational Linguistic*, pages 61–83, 1992.

[Tap86] Charles Tappert. Handwriting recognition on transparent tablet over flat display. In *Research Report RC*, page 14, Yorktown Heights, N.Y., 1986. International Business Machines Inc.

[TSW90] C. C. Tappert, C. Y. Suen, and T. Wakahara. The State of the Art in On-Line Handwriting Recognition. *IEEE Transactions on Pattern Analysis and Machine*, 12(8):787–808, 1990.

[Wan88] Zi-Xiong Wang, editor. *Structural Analysis of Handwritten Mathematical Expressions.* IEEE-Proceedings, 1988, 1988.

[War91] W. Ward. Understanding spontaneous speech: the phoenix system. *ICASSP'91 (Toronto, Canada)*, 1991.

[WHG+89] A. Waibel, T. Hanazawa, G.Hinton, K. Shikano, and K.Lang. Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustic, Speech and Signal Processing*, 37(3), March 1989.

[Win95a] H.-J. Winkler. A soft-decision approach for structural analysis of handwritting mathematical expressions. In *Conference Proceedings*, volume 4 of *IEEE, ICASSP-95*, pages 2459–2462, Piscataway, NJ USA, 1995. Institute for Human-Machine Communication, University of Technology, Munich, Germany.

[Win95b] Hans-Jürgen Winkler, editor. *Segmentation and recognition of Symbols within Handwritten Mathematical Expressions.* IEEE 1995, 1995.

[WL90] Alex Waibel and Kai-Fu Lee. *Readings in Speech Recogniton.* Morgan Kaufmann, 1990.

[Yag80] R.R. Yager. On a general class of fuzzy connectives. In *Fuzzy Sets and Systems*, 1980.

[Yan92] Der-Shung Yang. *Management of standard graphic symbol in a computer-aided design and drafting environment using neural network approaches.* PhD thesis, University of Illinois at Urbana-Champaign, 1992.

[Zad65] L.A. Zadeh. Fuzzy sets. In *Information and Control*, pages 338–353, 1965.