

# Algorithmische Interpretation von Wegeskizzen

## Segmentierung und Gruppierung 1

Technischer Bericht Nr. 2 des Projektes:  
Konzeptualisierungsprozesse in der Sprachproduktion<sup>1</sup>

Markus Guhe  
Steffen Huber

Arbeitsbereich WSV  
Fachbereich Informatik  
Universität Hamburg  
Vogt-Kölln-Straße 30  
22527 Hamburg  
{guhe, 1huber}@informatik.uni-hamburg.de

07. Mai 2001

---

<sup>1</sup> DFG-Projekt: HA 1237/10-1 Antragsteller Christopher Habel. Dieses Projekt ist am DFG-Schwerpunktprogrammes „Sprachproduktion“ beteiligt.

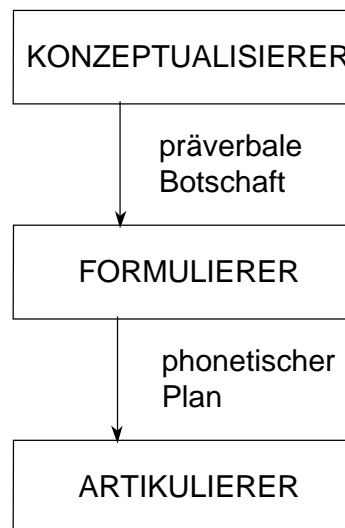
<b>1</b>	<b>EINLEITUNG</b>	<b>3</b>
<b>2</b>	<b>EINBETTUNG IN DAS GESAMTSYSTEM</b>	<b>5</b>
<b>3</b>	<b>PRINZIPIEN DER WAHRNEHMUNG</b>	<b>6</b>
3.1	DIE BAUSTEINE	6
3.2	SEGMENTIERUNG UND GRUPPIERUNG	9
3.3	ÜBERGEORDNETE PRINZIPIEN	11
3.3.1	PRINZIP DER LOKALEN ENTWICKLUNG	11
3.3.2	PRINZIP DER LINEAREN FORTSETZUNG	12
<b>4</b>	<b>MODELLIERUNG DER EREIGNIS- UND DER OBJEKTSTRUKTUR</b>	<b>12</b>
4.1	DAS ZUSAMMENSPIEL VON EREIGNISSEN UND OBJEKTEN	12
4.2	DIE ROLLE VON PAUSEN FÜR DIE MODELLIERUNG	14
4.3	DIE BEHANDLUNG VON AUSRUTSCHERN	15
4.4	ERKENNEN VON KREUZUNGEN	16
<b>5</b>	<b>DIE MODELLIERUNG</b>	<b>17</b>
5.1	DAS BEISPIEL FÜR DIE ENTWICKLUNG	17
5.2	ÜBERBLICK ÜBER DIE SYSTEMARCHITEKTUR	18
5.3	DIE MODULE IM EINZELNEN (AUßER EVENTS)	20
5.3.1	DAS MODUL CLASSIFY	20
5.3.2	DAS MODUL OBJECTS	20
5.3.3	DAS MODUL SPATIALNET	20
5.3.4	DAS MODUL STROKES	20
5.3.5	DAS MODUL GEO	20
5.3.6	DAS MODUL RELATIONEN	20
5.4	DAS MODUL EVENTS	21
5.4.1	DIE HAUPTROUTINE NEXTSTROKE	21
5.4.2	DIE ROUTINE REGISTERSEGMENT	23
5.4.3	DIE ROUTINE CROSSROADSDETECTION	24
<b>6</b>	<b>AUSBLICK</b>	<b>25</b>
	<b>ANHANG A. DIE SKIZZE 10-3</b>	<b>26</b>
	<b>ABBILDUNGSVERZEICHNIS</b>	<b>26</b>



# 1 Einleitung

Das Ziel des von der DFG geförderten Projektes „Konzeptualisierungsprozesse in der Sprachproduktion“ ist, mit theoretischen, empirischen und computerlinguistischen Mitteln Aufschlüsse über die mentalen Prozesse zu gewinnen, die in der (menschlichen) Sprachproduktion als Konzeptualisierungsprozesse bezeichnet werden. Konzeptualisierung steht dabei für den Aufbau von Sachverhaltsrepräsentationen aus perzipierten und kognizierten Entitäten, vgl. Tappe & Habel (1998) sowie Habel & Tappe (im Druck).

Wir verwenden das Sprachproduktionsmodell Levelts (1989) als Rahmen, in dem der Sprachproduktionsprozeß in drei Ebenen untergliedert wird: Konzeptualisierung, Formulierung und Artikulation, bzw. in drei Module, in denen jeweils eine dieser Ebenen realisiert ist<sup>2</sup>: den *Konzeptualisierer*, den *Formulierer* und den *Artikulierer*, vgl.



**Abbildung 1: Modularisierung der Sprachproduktionskomponenten nach Levelt (1989, 9)**

Abbildung 1. Zwischen den Ebenen der Konzeptualisierung und der Formulierung befinden sich die sog. *präverbalen Botschaften*, die das „Verbindungsglied“ zwischen Konzeptualisierer und Formulierer darstellen. Der Konzeptualisierer erzeugt solche präverbalen Botschaften, die dann vom Formulierer übernommen und weiter bearbeitet werden. Präverbale Botschaften haben ein propositionales Format und enthalten einen (den nächsten) Abschnitt der zu versprachlichenden Äußerung in einer vorsprachlichen konzeptuellen Struktur. Der Formulierer nimmt nun je eine dieser Propositionen und erzeugt durch eine syntaktische Enkodierung daraus den sog. *phonetischen Plan*, d.h. den Plan für die phonetische Realisierung. Dieser Plan wird an den Artikulierer weitergegeben, der die lautliche Realisierung steuert.

Habel und Tappe (im Druck) schlagen eine Modellierung der Konzeptualisierung aus fünf Teilprozessen vor, nämlich Segmentierung, Strukturierung, Selektion, Linear-

<sup>2</sup> Es ist eine noch nicht entschiedene Frage, ob ein Modul vollständige Ergebnisse produziert, die dann vom jeweils nächsten als Eingabe genommen werden oder ob es einen permanenten Informationsfluß gibt, bei dem evtl. die genannten Zwischenrepräsentationen ‘präverbale Botschaft’ und ‘phonetischer Plan’ gar nicht als eigenständige Entitäten erzeugt werden.

sierung und Perspektivierung:

- „*segmentation* of states of affairs: the distinction of those entities that are relevant within a current conceptualization, especially: temporal and spatial segmentation
- *structuring*: of states of affairs: the construction of structured object and state of affairs representations, especially: the construction of hierarchical structures
- *selection*: of objects and states of affairs: the selection of entities that are to be verbalized
- *linearization*: the linear arrangement of the selected states of affairs in a propositional format
- *perspectivization*: microlinearization including focus-background structuring / subdivision“ (Habel & Tappe, im Druck)

Wir verwenden „Verbalisierungen von Wegeskizzen“, um Sprachdaten zu erhalten, die wir dann mit computerlinguistischen Mitteln analysieren und modellieren. Dazu erstellen Versuchspersonen in einer ersten Phase Skizzen, die einen Weg beschreiben. Diese Skizzen zeichnen sie auf einem Graphiktablett; das bietet die Möglichkeit, die Daten Punkt für Punkt aufzuzeichnen und sie zu einem späteren Zeitpunkt ebenfalls Punkt für Punkt wieder abzuspielen. – Außerdem sind Manipulationen der Skizzen möglich, die für einen späteren Projektabschnitt geplant sind. – Dieses „Punkt-für-Punkt-Vorgehen“ hat den Vorteil strikter Sequentialität, d.h. sämtliche Strukturierungen der Daten, z.B. Gruppierungen oder Hierarchisierungen, sind nicht durch eine bereits vorhandene intrinsische Struktur der Daten begründet. Entstehen also beispielsweise zwei parallele Striche, die eine Straße darstellen, so muß man aus der sequentiellen Anhäufung von Punkten diese Struktur ermitteln; es ist nicht möglich, sich darauf zu berufen, daß die Straße bereits in der Datenstruktur enthalten war.

Die so festgehaltene Entstehung einer Skizze wird in einer zweiten Phase anderen Versuchspersonen auf einem Bildschirm dargeboten. Da wir in unserem Projekt insbesondere die Konzeptualisierungsprozesse von Ereignisbeschreibungen betrachten, wird die Skizze so abgespielt wie sie aufgenommen wurde und nicht als fertige Skizze präsentiert; denn nicht diese, sondern die Prozesse der Skizzenentstehung stehen im Kern des Interesses. Die Aufgabe der Versuchsperson besteht nun in dieser Phase darin, zu beschreiben, „was auf dem Bildschirm passiert“, sie *verbalisiert* die Skizze. Dabei werden zwei unterschiedliche Modi verwendet: *online* und *offline*, d.h. in dem einen Fall verbalisiert die Versuchsperson die Skizze *während* der Entstehung (*online*), in dem anderen *nachdem* die Skizze abgeschlossen und nicht mehr auf dem Bildschirm zu sehen ist (*offline*). Der zweite verwendete Parameter ist, den Versuchspersonen vorab Kontextinformationen zu geben, sie also darüber zu informieren, daß ihnen eine Wegeskizze präsentiert wird, oder sie nicht über den Charakter des Darzustellenden aufzuklären.

Die Erkenntnisse, die wir aus diesen empirisch erhobenen Daten gewonnen haben, führen zu theoretischen Überlegungen, die wir mit Hilfe computerlinguistischer Mittel nachvollziehen und überprüfen. Hier liegt ein weiterer Vorteil der strikt sequentiellen Natur der Ausgangsdaten: Da diese Überprüfungen durch Modellierungen – zumindest von Teilen und Teilaspekten – erfolgen, ist es essentiell, daß wir wohldefinierte Eingabedaten haben, um eine gesicherte, wenn auch spezielle Basis zu haben.

Die Entstehung einer Skizze kann insgesamt als ein Ereignis konzeptualisiert werden; wir sprechen deshalb auch von *Skizzenentstehungsereignissen*. Diese Art von Ereignissen zu betrachten, bietet neben dem gerade geschilderten Vorteil der strikten Sequentialität

lität einen weiteren, nämlich den, daß diese Ereignisse *resultative Ereignisse* sind. Die Resultativität ist deswegen ein Vorteil, weil die Zuordnung von Ereignis und dem dazugehörigen Ergebnis oder Ergebniszustand, einfacher ist als in anderen Domänen. Auf der anderen Seite jedoch fällt diese Unterscheidung nicht immer leicht; denn alle Objekte, die in einer Skizze existieren, sind durch ein Entstehungsereignisse erzeugt worden, und alle Ereignisse führen zur Entstehung von Objekten – nimmt man einmal Ereignisse aus, die dazu dienen, bereits existierende Objekte „durchzustreichen“ –. Das Zeichnen eines Striches beispielsweise wird also als Entstehungsereignis des Striches wahrgenommen<sup>3</sup>, während das Ergebnis das graphische Objekt ‘Strich’ ist.

## 2 Einbettung in das Gesamtsystem

Bei der Fähigkeit des Menschen, Entitäten in der ihn umgebenden Welt wahrzunehmen, unterscheidet man im allgemeinen einen *perzeptuellen* und einen *konzeptuellen*, oder vielmehr *konzeptualisierenden* Bereich. Es ist allerdings fraglich, ob eine solche Abgrenzung überhaupt in der Weise erfolgen kann, daß man zwei voneinander getrennt arbeitende Module erhält; denn nimmt man beispielsweise ein Eichhörnchen wahr, perzipiert es also, führt man zugleich einen Konzeptualisierungsschritt durch, nämlich die Zuordnung des visuellen Eindrucks zu dem Konzept EICHHÖRNCHEN.

Dieser üblicherweise in der Kognitionswissenschaft angenommene perzeptuelle (Vor-)Verarbeitungsschritt vor einem konzeptualisierenden findet sich auch bei unserem System – und wir treffen auch auf die gleichen Abgrenzungsprobleme –; denn die konzeptualisierende Komponente wird in einer Erweiterung des Formalismus *Semantic Representation Language* (SRL) (Habel, 1986) modelliert, der als eine Implementierung in Prolog existiert (Eschenbach, 1988). – Für diese Implementierung gibt es bereits Vorarbeiten, um damit Skizzen in einem referentiellen Netz darstellen zu können (Huber & Foth, 1998). – Prolog jedoch ist aufgrund der Sprachstruktur und seines hohen Ressourcenbedarfs denkbar ungeeignet, große Mengen gleichförmiger Daten zu verarbeiten, wie

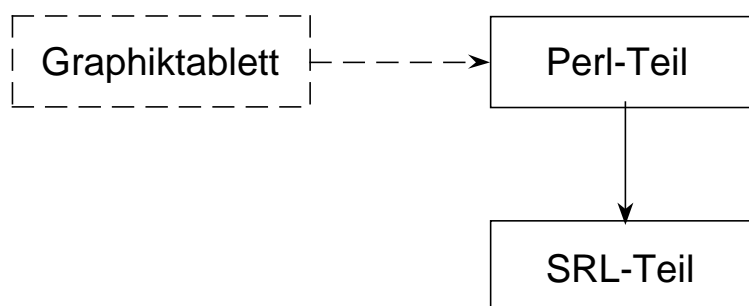


Abbildung 2: Der Datenfluß im Gesamtsystem

es bei den Daten des Graphiktablets der Fall ist. Die perzeptuelle Komponente ist des-

<sup>3</sup> Genauer gesagt besteht dieses Ereignis aus dem Aufsetzen des Zeichenstiftes, dem Erzeugen einer Reihe von (Koordinaten)Punkten und dem Absetzen des Stiftes.

wegen in Perl<sup>4</sup> geschrieben, das sich aufgrund seiner großen Variabilität und der einfachen Verbindung verschiedenster Systemebenen und Programmiersprachen dazu besonders anbietet. In diesem Bericht beschäftigen wir uns mit den Grundlagen dieses vorverarbeitenden Perl-Teils. Dieser Teil ist wie in Abbildung 2 dargestellt in das Gesamtsystem eingegliedert.

Die Daten des Graphiktablets bestehen aus Tripeln, die jeweils aus einem Koordinatenpaar und einem Flag, das anzeigt, ob der Zeichenstift auf- oder abgesetzt ist, bestehen. Die Abtastung erfolgt mit einer konstanten Abtastrate, so daß zeitliche Informationen erhalten bleiben: Jedes neue Tripel wird 50 ms nach den vorherigen erfaßt.

Bei der Analyse von Eingabedaten bieten sich prinzipiell zwei Vorgehensweisen an: *top-down* und *bottom-up*. Das erste Vorgehen – auch als *hypothesis driven* bezeichnet – besteht darin, zunächst eine Datenstruktur aufzubauen, welche die zu erwartenden Daten aufnimmt; werden dann Daten eingelesen, werden die „freien Plätze“ in der Datenstruktur nach und nach aufgefüllt. Das zweite Vorgehen – auch als *data driven* bezeichnet – entscheidet aufgrund der eingelesenen Daten, welche Strukturen darüber gebildet werden. Beide Methoden haben sowohl Nach- als auch Vorteile, die bereits oft und ausführlich diskutiert wurden, z.B. in Marcus (1980). Wie auch Marcus halten wir uns an ein kombiniertes Vorgehen: *data driven* deswegen, weil die von uns erzeugten Datenstrukturen, von den Eingabewerten abhängen; *hypothesis driven* deswegen, weil wir einen menschlichen Erkennen simulieren wollen, der mit Erwartungen<sup>5</sup> arbeitet, was z.B. durch *priming* Experimente immer wieder nachgewiesen wird. Die in Abschnitt 3.2 dargestellten Mechanismen fallen in die Kategorie *data driven*, die beiden in Abschnitt 3.3 in die Kategorie *hypothesis driven*.

## 3 Prinzipien der Wahrnehmung

Obwohl es sich bei den Daten um eine strikt sequentielle Vermehrung von Punkten auf einer zunächst leeren Fläche handelt, nimmt ein menschlicher Betrachter die Entstehung einer Skizze nicht nur als eine kontinuierliche Entwicklung wahr, sondern insbesondere als eine inkrementelle („stückchenweise“) Vermehrung von Objekten. Er untergliedert und segmentiert also den Datenstrom. Zugleich besitzt er die Fähigkeit, zur Gruppierung, d.h. werden z.B. nur drei Seiten eines Rechtecks gezeichnet, dann etwas anderes, nicht zu dem Rechteck Gehörendes, und schließlich die vierte Seite, dann ist er trotz der zeitlichen Unterbrechung der Entstehung in der Lage, die entsprechenden Elemente zu einem Rechteck zu gruppieren, vgl. Habel (1996) und Habel & Tappe (im Druck).

### 3.1 DIE BAUSTEINE

Bevor wir aber näher auf Gruppierung und Segmentierung zu sprechen kommen, wollen wir zunächst die Frage klären, was in unserem Falle die kleinsten Bausteine sind, die wir als *graphische Primitive* bezeichnen. Nach Erichsen (1997, 61) gibt es drei verschiedenen Perspektiven aus denen man sie definieren kann:

<sup>4</sup> Perl steht für *Practical Extraction and Report Language* (Wall et al., 1997, xv). Dieser Name ist Programm: Mit möglichst einfachen Mitteln sollen insbesondere solche Aufgaben möglichst einfach bewältigt werden können, bei denen große Mengen gleichförmiger Daten verarbeitet werden müssen. Eine Einführung bietet Schwarz (1993).

<sup>5</sup> Wir verstehen in diesem Kontext *hypothesis* und Erwartung oder Erwartungshaltung als synonym.

- 1) *kognitiv-physiologisch*: Diese Sichtweise bestimmt die Primitive auf der Basis von motorischen Abläufen.
- 2) *geometrisch*: Hier stehen die Eigenschaften der Bestandteile, also die graphischen Objekte einer Skizze im Vordergrund.
- 3) *semantisch*: Unter dieser Perspektive werden Kontextinformationen zur Bestimmung herangezogen. Das setzt voraus, daß der Gegenstandsbereich und die Aussage einer Skizze bekannt sind.

Eine Charakterisierung von graphischen Primitiven aufgrund einer motorischen Sichtweise geht also von „Ergebnis[sen] von standardisierten Bewegungsabläufen, die ohne Aufmerksamkeit ausgeführt werden können“ aus (Erichsen, 1997, 61). Auch wenn dies auf den ersten Blick lediglich eine Möglichkeit zu sein scheint, die Produktion und nicht die Wahrnehmung zu unterteilen, bieten sich insbesondere das An- und Absetzen des Zeichenstiftes als saliente Segmentierungssignale an; denn man kann davon ausgehen, daß diese Unterteilungen häufig durch Planungsprozesse des Zeichners bedingt sind. Die zweite Möglichkeit besteht darin, die graphischen Elemente nach geometrischen Gesichtspunkten zu klassifizieren, d.h. ein Rechteck wird z.B. als Spezialfall eines Parallelogramms interpretiert. Die semantische Perspektive schließlich bietet sich an, wenn der Kontext der Skizze bekannt ist, d.h., ist ein Programm z.B. darauf spezialisiert, Flußdiagramme zu erkennen, können die Eingabedaten auf eine sehr stark eingegrenzte Zielmenge hin interpretiert werden.

Für unsere Zwecke können wir uns nicht auf eine der Sichtweisen beschränken, sondern verwenden eine Kombination aus allen dreien. So nutzen wir das bereits erwähnte An- und Absetzen des Stiftes als Kriterium für eine erste Segmentierung des Eingabestroms, unterscheiden aber außerdem zwischen *Stroke* und *Stroking*<sup>6</sup>. Dabei bezeichnet *Stroking* das Ereignis, *Stroke* das Ergebnis, also das graphische Objekt:

DEFINITION 1: Ein *Stroking* ist das Ereignis, das aus dem Aufsetzen des Stiftes auf das Grafiktablett, der Erzeugung einer Menge von Punkten und dem Absetzen des Stiftes besteht.

DEFINITION 2: Ein *Stroke* ist die Menge von Punkten, die durch ein *Stroking* erzeugt wird.

Ein wesentlicher Teil der theoretischen Überlegungen besteht darin, vier verschiedene Repräsentationsschichten zu unterscheiden, vgl. Tappe und Habel (1998, 2f):

- 1) die Schicht der graphischen Objekte, z.B. Striche, Quadrate, Rechtecke,
- 2) die Schicht der intendierten Objekte, z.B. der Campus, die xy-Straße, die U-Bahn Station,
- 3) die Schicht der graphischen Ereignisse, z.B. ein Strich erscheint, ein Quadrat wird gezeichnet, und
- 4) die Schicht der intendierten Ereignisse, z.B. eine Ankunft an der U-Bahn Station.

Es gibt also, anschaulich gesprochen, zwei Faktoren, welche die vier Schichten erzeugen: Objekte versus Ereignisse und die graphische Darstellung versus intendierten Entitäten, auch als Realwelt bezeichnet. – Die Realwelt ist insofern intendiert, als daß ein Betrachter der Skizze nicht die Realweltobjekte selbst vor Augen hat, sondern lediglich ihre Darstellungen innerhalb der Skizze. – Für die Analyse, und insbesondere die computergestützte Analyse, kommt ein weiteres Paar von unterscheidenden Faktoren hinzu:

<sup>6</sup> *Stroke* entspricht dem Begriff „Strich“ aus Huber und Foth (1998), *Stroking* entspricht ihrem „Stroke“.

dynamische versus statische Auswertung, d.h. findet die Auswertung während des Einlesens der Daten (dynamisch, inkrementell) oder im Anschluß daran (statisch) statt. Da wir die Prozesse der Skizzenentstehung betrachten und Menschen im *online* Fall offensichtlich eine Ereigniskonzeptualisierung vornehmen, d.h. bereits während der Betrachtung das Geschehen analysieren, und wir hier ein System erstellen wollen, das sich möglichst eng an kognitive Prozesse anlehnt, halten wir uns ebenfalls an ein dynamisches Vorgehen. Die Betrachtung der Dynamik hat den Vorteil, daß man dabei auf die Betrachtung der statischen Eigenschaften nicht zu verzichten braucht, ja sie sogar benötigt, um überhaupt Objekteigenschaften ermitteln zu können. Bezieht man nur die statischen Eigenschaften in die Überlegungen ein, ohne die dynamischen zu berücksichtigen, so hat man keine Möglichkeit auf die Art und Weise der Entstehung einer Skizze und die Einflüsse der Entstehung auf das Resultat einzugehen.

Inkrementell bedeutet in diesem Zusammenhang vor allem, daß, zum einen, eine Analysemethode gewählt wird, bei der eine Sachverhaltsrepräsentation aufgebaut wird, die erst nach und nach, mit jedem Inkrement größer wird, und daß, zum anderen, der Eingabestrom in Inkremente („Stücke“) unterteilt wird. Dabei stellt sich die Frage nach der Größe eines Inkrements, d.h. in unserem Falle die Frage, wie viele Punkte eingelesen werden<sup>7</sup>, bevor sie analysiert werden. Die in Foth et al. (1999)<sup>8</sup> vorgestellte Analysemethode wartet jeweils bis zum Ende eines Strokings, bevor der Stroke auf seine Eigenschaften hin untersucht wird. Dieses Vorgehen kann aber wohl kaum von sich behaupten, kognitive Realität widerzuspiegeln; deshalb verwenden wir eine punktweise Methode, d.h. nachdem ein Punkt eingelesen wurde, wird er in bezug auf das bereits vorhandene Wissen interpretiert, und die Sachverhaltsrepräsentation wird dementsprechend verändert und / oder ergänzt.

In Foth et al. (1999) wird die in Abbildung 3 dargestellte Hierarchie zur Klassifikation bei einem stroke-basierten Ansatz vorgestellt. Dabei gilt allgemein das Prinzip, daß versucht wird, für einen Stroke eine möglichst spezielle Klasse zu finden. Ist das nicht

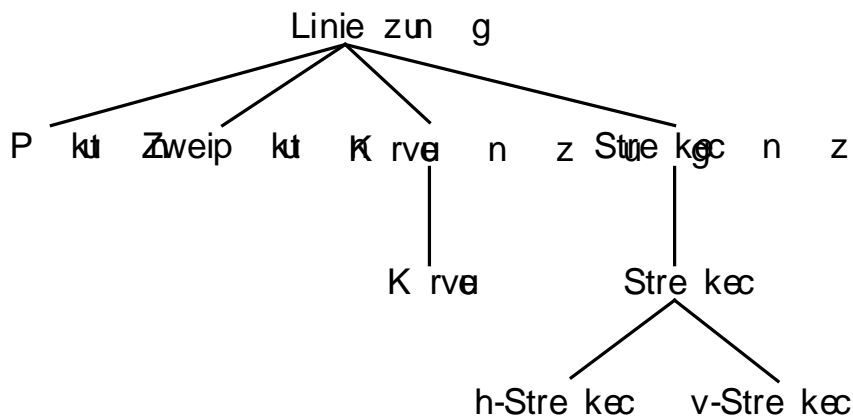


Abbildung 3: Klassifikation von Strokes

möglich, wird die nächst allgemeinere Klasse genommen; sind also beispielsweise die

<sup>7</sup> Da die Koordinatenpunkte mit einem festen Takt abgetastet werden, ist das gleichbedeutend mit der Frage, wieviel Zeit vor dem nächsten Analyseschritt vergeht.

<sup>8</sup> Der Bericht von Foth, Grochowina und Turhan (1999) wurde im Rahmen eines Projektseminars zu diesem Projekt erstellt. Das System, das wir im weiteren als Basis unserer Arbeit benutzen ist in diesem Projektseminar entstanden.

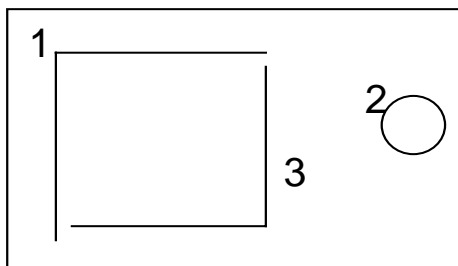
Kriterien für eine Kurve nicht erfüllt, wird versucht, den Stroke als Kurvenzug zu klassifizieren; ist auch das nicht möglich, wird er als Linienzug identifiziert. Diagonale Strecken werden so als Strecke klassifiziert.

Der allgemeinste Fall ist demzufolge der *Linienzug*, dem jeder Stroke zugeordnet werden kann, sofern er die Bedingungen für einen der spezielleren Fälle nicht erfüllt. Besteht ein Stroke lediglich aus einem (Koordinaten)Punkt, d.h. hat er die Breite und die Höhe 1, wird er als *Punkt* definiert. Objekte, die aus genau zwei Punkten bestehen, werden als *Zweipunkt* klassifiziert. Zweipunkte können viele verschiedene Interpretationen erfahren; so sind sie zwar Idealstrecken, d.h. da wir, um Strecken zu erhalten, jeweils zwei (Abtast)Punkte durch eine gerade Linie miteinander verbinden müssen, enthalten Zweipunkte keine durch Zeichengenauigkeiten entstandene Abweichungen in ihrem Verlauf. Dennoch ist in den meisten Fällen nicht beabsichtigt, eine Strecke zu zeichnen, sondern ein anderes graphisches Objekt, z.B. einen Punkt.

Ein Stroke, der ausschließlich aus (geraden) Strecken und Ecken gebildet wird, wird als *Streckenzug* bezeichnet, enthält er außerdem keine Ecken, handelt es sich um eine *Strecke*, wobei man hier *vertikale* oder *v-Strecken* und *horizontale* oder *h-Strecken* als Spezialfälle unterscheidet. Sind keine Ecken in einem Linienzug enthalten, sprechen wir von einem *Kurvenzug*, gibt es außerdem keine Wendepunkte innerhalb eines Kurvenzuges – gibt es also nur eine Krümmung in eine Richtung –, handelt es sich um eine *Kurve*.

### 3.2 SEGMENTIERUNG UND GRUPPIERUNG

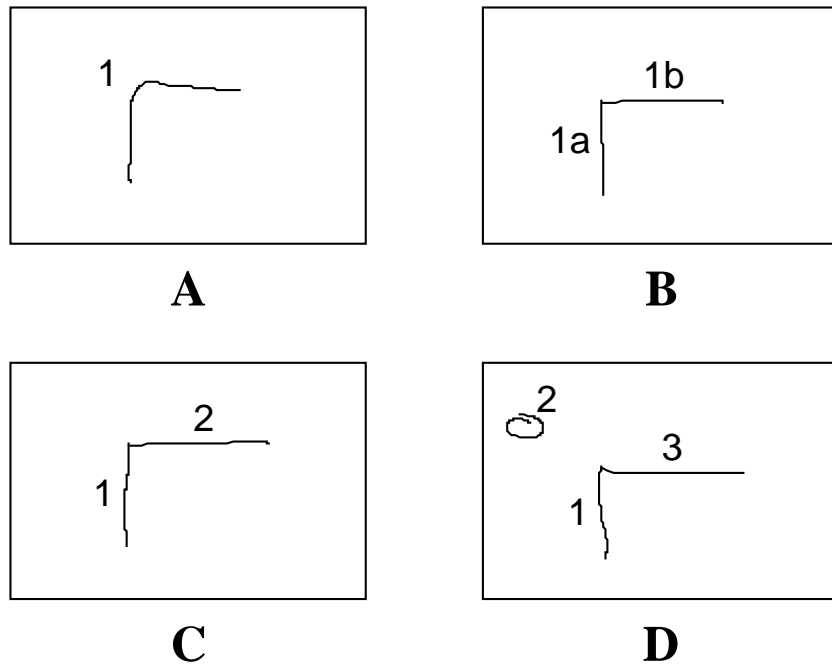
Neben den durch die Strokings vorgegebenen Segmentierungen gibt es ebenfalls die Notwendigkeit, bestimmte Elemente aufgrund anderer Kriterien zu segmentieren oder



**Abbildung 4: Eine erstes Beispiel**

zu gruppieren. Betrachten wir z.B. ein Skizzenentstehungsereignis wie in Abbildung 4, das aus drei Strokings, bzw. Strokes besteht. Nach dem Zeichnen des mit 1 bezeichneten Strokes, eines Winkels, der aus einer horizontalen und einer vertikalen Strecke besteht, kann man diesen Winkel in die beiden Teilstrecken segmentieren, wenn dies im Kontext sinnvoll ist. Nachdem 2 und 3 gezeichnet wurden, ist eine Konstellation entstanden, die es nahelegt, die durch die Ereignisse 1 und 3 entstandenen Elemente in der Schicht der graphischen Objekte zu einem Rechteck zu gruppieren, obwohl sie eindeutig *nicht* durch ein Ereignis erzeugt wurden und deswegen auch in der Schicht der graphischen Ereignisse nicht gruppiert werden. – In Guhe, Habel und Tappe (in Vorbereitung) wird ein Ereignis wie 3 als ein *Vervollständigungsereignis* charakterisiert, durch das dann eine Verbindung mit 1 hergestellt wird. –

Deutlich wird an diesem Beispiel auch ein zweiter Punkt, der bei der Analyse von Freihandskizzen berücksichtigt werden muß: Die Anfangs- und Endpunkte von Strokings bzw. Strokes liegen meistens nicht auf demselben Koordinatenpunkt, d.h. will man ein robustes, fehlertolerantes System, muß hier immer eine gewisse Abweichung berücksichtigt werden; denn menschliche Zeichner produzieren nur in den seltensten Fällen tatsächlich ideale graphische Objekte.

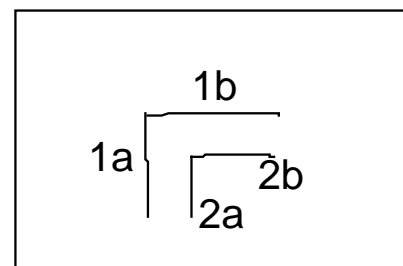


**Abbildung 5: Vier Möglichkeiten einen Winkel zu zeichnen**

Realistischere Beispiele – realistischer nicht zuletzt deswegen, weil man hier deutlich erkennen kann, daß Strecken kaum gerade sind – finden sich in Abbildung 5. Dort werden vier verschiedene Möglichkeiten gezeigt, wie ein Winkel gezeichnet werden kann, der, wie im vorangegangenen Beispiel, Teil eines Rechtecks sein kann oder der eine von zwei parallelen Streckenzügen, mit denen üblicherweise eine Straße dargestellt wird.<sup>9</sup>

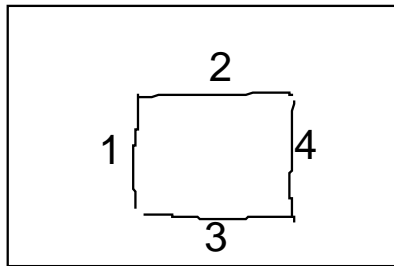
Der Regelfall, einen Winkel zu zeichnen, auch wenn wir dazu noch keine gezielte Untersuchung durchgeführt haben, ist in Abbildung 5.B ist zu sehen: In einem Stroking, doch mit einer Pause am Eckpunkt während sich der Stift auf dem Graphiktablett befindet, welche die Entstehung der zwei Abschnitte deutlich voneinander trennt. – Die kleinen Buchstaben zeigen die zwei Abschnitte des Stroking, bzw. Strokes an. – Diese Pause ist ein salienter Segmentierungspunkt, d.h. in bestimmten Analysesituationen muß der Streckenzug an dieser Stelle aufgeteilt werden, etwa dann, wenn die Zeichnung wie in Abbildung 6 fortgesetzt wird. Hier handelt es sich um eine abknickende Straße, d.h. betrachtet man die Skizze etwa auf der Ebene der intendierten Realweltobjekte, werden 1a und 2a der einen Straße, 1b und 2b einer anderen zuordnen.

Nicht immer aber gibt es bei einer systematischen Betrachtung der verschiedenen möglichen Fälle solche salienten Punkte für die Segmentierung, vgl. Abbildung 5.A. Dieses Stroking wurde ohne Pause gezeichnet, und auch die Ecke ist bei weitem nicht so deutlich ausgeprägt. Nichtsdestotrotz ist eine ähnliche Fortsetzung wie in



**Abbildung 6: Eine mögliche Fortsetzung von Abbildung 5.B**

<sup>9</sup> Straßen werden bei Skizzen, die nur einen kleinen Ausschnitt darstellen, meistens durch eine Doppellinie dargestellt; bei Skizzen für einen größeren Ausschnitt werden z.T. auch nur Einzellinien für Straßen verwendet.



**Abbildung 7: Eine mögliche Fortsetzung von Abbildung 5.C**

Abbildung 6 möglich, und die Segmentierung der Strokes, bzw. Strokes muß auf ähnliche Weise erfolgen.

Ist es bereits schwierig, die richtigen Kriterien für Segmentierungen zu finden, wird die Aufgabe bei Gruppierungen nicht einfacher. Abbildung 5.C zeigt z.B. eine Skizze, in welcher der Stift an demselben Punkt wieder aufgesetzt wurde, an dem er abgesetzt wurde. In einem anderen Kontext kann für dieses Beispiel die Notwendigkeit bestehen, die beiden Strokes – hier bezeichnet mit 1 und 2 – zu gruppieren, vgl. z.B. die in Abbildung 7 dargestellten Fortsetzung der Skizze,.

Für den menschlichen Betrachter des Resultates ist es eindeutig, daß ein Rechteck gezeichnet wurde. Grundlage dafür ist, daß alle vier Strokes, die in vier separaten Strokings, also in vier verschiedenen Ereignissen gezeichnet wurden, werden als (gerade) Strecken interpretiert, die zu einem (komplexeren) graphischen Objekt zusammengefaßt oder gruppiert werden. Diese Gruppierung kann beim Betrachter noch vor Abschluß der Entstehung geschehen, wenn bereits genug des gesamten Objektes gezeichnet wurde, um es zuzuordnen zu können, vgl. auch Abschnitt 5.4.

Ein weiteres Problem kommt hinzu, wenn zwei Strokes, die gruppiert werden müssen, zeitlich nicht aufeinander folgen, sondern zwischen ihren Entstehungsereignissen etwas anderes gezeichnet wurde, wie in Abbildung 5.D zu sehen. Allgemein ergibt sich daraus das Problem, daß das Prinzip der lokalen Entwicklung (siehe Abschnitt 3.3.1) kein universelles Gesetz, sondern nur eine Präferenz ist, d.h. bei Gruppierungsprozessen kann man sich nicht auf die zeitliche Nähe verlassen, also darauf, daß zu gruppierende Elemente direkt nacheinander ohne dazwischen liegendes Ereignis entstehen – was bei einer sequentiellen Analyse ein sehr bequemes Vorgehen wäre –, sondern muß vorrangig die räumliche Nähe betrachten.

### 3.3 ÜBERGEORDNETE PRINZIPIEN

In Erichsen (1997) werden zwei Prinzipien beschrieben, die in der Tradition der Gestaltpsychologie stehen und die bei Skizzenentstehungsereignissen als übergeordnet angenommen werden können: Das *Prinzip der lokalen Entwicklung* und das *Prinzip der linearen Fortsetzung*. Diese beiden Prinzipien sind deswegen bedeutend, weil sie die Grundlage für Heuristiken oder den Aufbau von Erwartungshaltungen bei der Betrachtung der Konzeptualisierung von Skizzenentstehungsereignissen bilden können.

#### 3.3.1 Prinzip der lokalen Entwicklung

Das erste der beiden Prinzipien beschreibt Erichsen folgendermaßen:

„Ein zentrales Ergebnis der Skizzenanalyse ist, daß Skizzen grundsätzlich *lokal entwickelt* werden, d.h. Strokes nach dem Prinzip der räumlichen Nähe [...] angeordnet werden. Lokales Entwickeln bezeichnet dabei die Einteilung einer Skizze in einzelne Bereiche, die jeweils ausgeführt werden, ehe in weiteren, in der Regel angrenzenden Bereichen weitergezeichnet wird.“ (Erichsen, 1997, 79f)

In dem Beispiel aus Abbildung 7 hat dieses Prinzip die Konsequenz, daß zunächst alle vier Seiten des Rechtecks gezeichnet werden, bevor sich der Zeichner dem nächsten Objekt zuwendet. Doch besitzt das Prinzip keine universelle Gültigkeit: Die Skizze 10-

3, die wir als Beispiel für den ersten Teil der computerlinguistischen Realisierung verwenden<sup>10</sup>, vgl. Anhang A, enthält z.B. eine solche Nichtlokalität: Dort wird zunächst eine Strecke als der erste Teil der Begrenzung des intendierten Realweltobjektes CAMPUS gezeichnet. Dann jedoch entsteht ein Rechteck mit der Beschriftung ‘Damtor’, das den DAMMTORBAHNHOF repräsentiert, bevor die Strecke für den CAMPUS zu einer geschlossenen Fläche ergänzt und mit ‘Campus’ beschriftet wird.

Faßt man dieses Prinzip jedoch etwas weiter – und bezieht es auf Bereiche einer Skizze und nicht nur auf die Entstehung graphischer Objekte –, folgt der Großteil der Entstehungsprozesse diesem Prinzip. Das bedeutet, man kann bei einer automatisierten Analyse die Erwartung aufbauen, daß das nächste Entstehungsereignis in räumlicher Nähe zu dem vorigen Entstehungsereignis stattfindet.<sup>11</sup>

### 3.3.2 Prinzip der linearen Fortsetzung

Das *Prinzip der linearen Fortsetzung* (Erichsen, 1997, 81f) kann man, grob betrachtet, als die zeitliche Variante des Prinzips der lokalen Entwicklung auffassen. Die in Abschnitt 5.3.6 beschriebenen räumlichen Relationen zwischen Strokes können verwendet werden, um einen Stroke zu seinem zeitlichen Vorgänger in Beziehung zu setzen. Das Prinzip der linearen Fortsetzung besagt, daß zeitlich direkt aufeinander folgende Strokes, bevorzugt in den Relationen *parallel-zu*, *anschluß-an* oder *fortsetzung-von* zueinander stehen. Tun sie das, bilden sie eine *lineare Fortsetzung* – linear bedeutet in diesem Zusammenhang *zeitlich direkt aufeinanderfolgend* –.

Für die Analysemethode bedeutet dies, daß man, wenn ein Stroking beendet wurde, den so entstandenen Stroke bevorzugt in eine der drei genannten Relationen zu seinem Vorgänger zu setzen versucht. Dadurch wird eine Erwartungshaltung aufgebaut, die erfüllt werden kann, wenn sich tatsächlich eine dieser Relationen feststellen läßt, die aber auch enttäuscht werden kann, wenn das nicht möglich ist. Dann muß eine andere Analyse gefunden werden.

Die Parallelität zum Menschen ist offensichtlich: Die Verarbeitung erfolgt schneller, wenn die Erwartung erfüllt, als wenn sie nicht erfüllt wird; denn bei den maschinellen Berechnungen werden zunächst die zu erwartenden Fälle überprüft, bevor sie sich anderen Möglichkeiten zuwendet. Wurde eine der Erwartungen bestätigt, ist die Berechnung beendet.

## 4 Modellierung der Ereignis- und der Objektstruktur

### 4.1 DAS ZUSAMMENSPIEL VON EREIGNISSEN UND OBJEKTEN

Wie bereits dargestellt, unterscheiden wir zwischen der Ereignis- und der Objektebene. Der Grund liegt darin, daß wir davon ausgehen, daß die Art und Weise, wie eine Skizze entsteht, einen wichtigen Einfluß auf ihre Wahrnehmung und Konzeptualisierung und damit speziell auf die Verbalisierungen ausübt, siehe auch Guhe, Habel und Tappe (in

<sup>10</sup> Diese Skizze wird auch in Foth et al. (1999) verwendet. Die Strokes, von denen wir hier sprechen haben dort die Nummern 63 bis 75.

<sup>11</sup> In dem von uns vorgestellten Ansatz soll das Modul SPATIALNET die Möglichkeit bieten, solche räumlichen Beziehungen zu erkennen, vgl. Abschnitt 5.3.3.

Vorbereitung). So kann z.B. die Weise wie die Kreuzung in Abbildung 8.A entsteht als „Da kreuzen sich zwei Straßen“ verbalisiert werden, während eine Zeichenreihenfolge wie in Abbildung 8.B als „Da wird eine Kreuzung gezeichnet“ konzeptualisiert und formuliert werden kann. Noch deutlicher wird der Unterschied zwischen den beiden Beispielen, wenn zwischen den Stroking 2 und 3 noch etwas anderes gezeichnet wird. Diese beiden Kreuzungen (oder sich kreuzenden Straßen) unterscheiden sich lediglich,

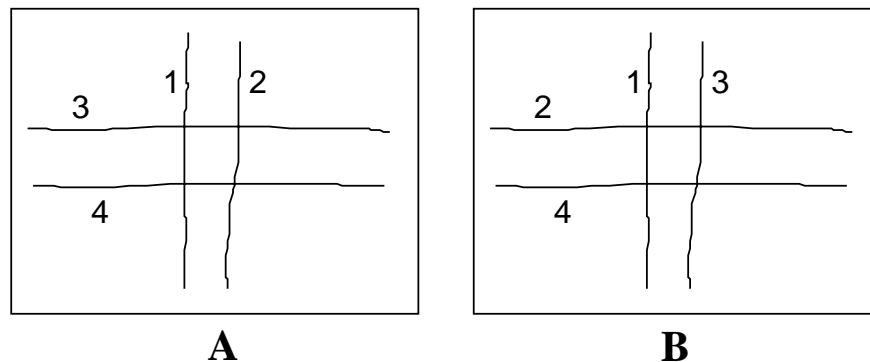


Abbildung 8: Kreuzende Straßen

liegt in der Reihenfolge ihres Entstehens und nicht in ihren Objekteigenschaften.

Zwei Möglichkeiten, wie Ereignisstrukturen über diesen beiden Beispielen aufgebaut werden können, sind in Abbildung 9 dargestellt. – Zur Strukturierung von Ereignissen siehe auch Habel und Tappe (im Druck) und Habel (1996). – In der Struktur zum Entstehungsereignis A wird die Gruppierung von je zwei Strecken zu einer Straße dargestellt. Daraus kann dann die bereits erwähnte Verbalisierung abgeleitet werden: Es sind zwei Straßen entstanden, die sich kreuzen. Aufgrund der Struktur des Entstehungsereignisses B kann eine solche Gruppierung auf einer mittleren Ebene nicht vorgenommen werden; lediglich alle vier Strecken können als die Entstehung einer Kreuzung interpretiert werden.

Betrachtet man nun die Objektstrukturen zu diesen beiden Kreuzungen, die auf der statischen Objektebene identische Strukturen haben – d.h. sie haben identische graphische und geometrische Eigenschaften –, ohne die Entstehung zu berücksichtigen, so hat man die freie Wahl zwischen zwei Datenstrukturen, die isomorph zu den in Abbildung 9 dargestellten sind. – Allerdings steht bei Objektstrukturen nicht Entstehungsereignis an der Wurzel, sondern z.B. Objekt A und Objekt B. – Die Entscheidung, ob das Ergebnis der Analyse eine Kreuzung oder zwei sich kreuzende Straßen ist, kann also nur aufgrund der Objekteigenschaften nicht getroffen werden; die Entstehungsstruktur muß berücksichtigt werden.<sup>12</sup>

<sup>12</sup> Es gibt noch die Möglichkeit, die Kreuzung als aus zwei Straßen bestehend zu analysieren, die jeweils aus zwei Strichen bestehen. Doch das löst das Problem nicht, daß es zwei unterschiedliche Konzeptualisierungen gibt, die zu zwei unterschiedlichen Verbalisierungen führen können. Selbstverständlich werden in einem zweiten Analyseschritt auch für das Entstehungsereignis A die Kreuzung und für Entstehungsereignis B die zwei Straßen ermittelt.

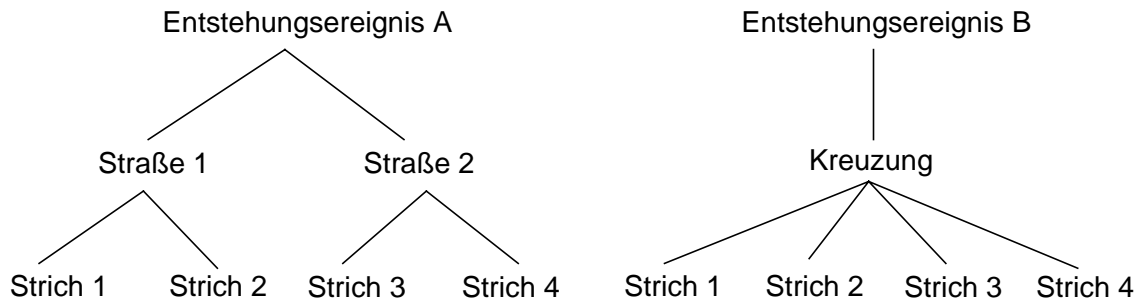


Abbildung 9: Ereignisstrukturen zu Abbildung 8

Doch muß man auch die Objektstruktur beim Aufbau der Ereignisstruktur berücksichtigen; denn erst die Objektstruktur macht es überhaupt möglich, z.B. das Entstehen eines Rechteckes zu erkennen. Nehmen wir noch einmal das Beispiel aus Abbildung 7; dort entsteht ein Rechteck durch vier Strokings, also durch vier separate Ereignisse. Nur durch die geometrischen Eigenschaften der so entstandenen Strecken läßt sich ein Entstehungsereignis für das Rechteck aufbauen. Auch daß sich die beiden Straßen in Abbildung 8 kreuzen – oder eine Kreuzung bilden –, wird erst aufgrund der Lage der Objekte zueinander deutlich und nicht durch ihre Entstehungsereignisse.

In den Datenstrukturen in der von uns im Abschnitt 5 vorgestellten Modellierung verwenden wir für Ereignisse und Objekte dieselben Namen, d.h. sowohl das Entstehungsereignis als auch das resultierende Objekt wird mit ‘Strecke’ bezeichnet. Da wir getrennte Datenstrukturen für Ereignisse und Objekte verwenden, ist die Zuordnung eindeutig.

In der obigen Darstellung haben wir eine Hierarchisierung beim Aufbau der Datenstrukturen für Objekte und Ereignisse angenommen, ohne näher darauf einzugehen; denn wir sind der Meinung, daß es für die Fälle, die wir hier betrachten, auf der Hand liegt, eine solche Hierarchisierung vorzunehmen. In Habel (1996) wird dargelegt, daß Ereignisse zu Ereignissen gruppiert werden können, die in der Hierarchie eine Stufe höher liegen und die dann selbst immer weiter gruppiert werden können, bis man einen Wurzelknoten für das gesamte Skizzenentstehungsereignis erhält. Ereignishierarchien sind also keine flachen Strukturen, in denen die Ereignisprimitive direkt unter der Wurzel hängen, sondern tiefe Strukturen, in der es Zwischenknoten gibt.

## 4.2 DIE ROLLE VON PAUSEN FÜR DIE MODELLIERUNG

Es gibt zwei verschiedene Möglichkeiten, beim Zeichnen einer Skizze Pausen zu machen: während der Zeichenstift oben und während er unten ist. Die Pausen, die entstehen, wenn der Stift oben ist, dienen gleichzeitig dazu, Strokings voneinander abzugrenzen. Die Pausen mit abgesetztem Stift befinden sich zwar innerhalb eines Strokings, können aber, wenn auch unter anderen Voraussetzungen, ebenfalls eine Grenze bedeuten, wie in Abschnitt 3.2 dargestellt.

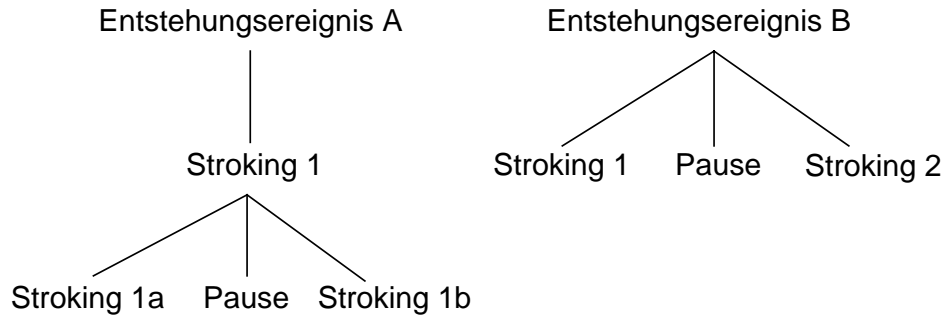


Abbildung 10: Datenstrukturen für Winkel mit Pausen der Abbildung 5.B und C

Behält man Strokings als Basiseinheiten bei, ergeben sich für die Beispiele aus Abbildung 5.B und C die Datenstrukturen wie sie in Abbildung 10 dargestellt sind. Doch ein Betrachter, also eine Versuchsperson, welche die Skizzenentstehung verbalisieren soll, sieht den Zeichenstift nicht, kann sich also darauf für die Segmentierung weder verlassen noch beziehen. – Denn der Stift wird ja auf dem Punkt wieder aufgesetzt, von dem er hochgenommen wurde. – Für eine Modellierung müßten also je nach Betrachtungsweise sowohl beide Möglichkeiten unterschieden, nämlich dann, wenn wir die Strokings betrachten, als auch zu einer zusammengefaßt werden, nämlich dann, wenn wir einen menschlichen Betrachter nachbilden.

Bedingt durch die spezielle Struktur des Beispiels, das wir für die Entwicklung verwenden (s. Abschnitt 5.1), brauchen wir zur Zeit die Entstehungsprozesse lediglich aufzuzeichnen, brauchen also noch keinen Mechanismus, um diesen Fall behandeln zu können. Eine der nächsten Überlegungen wird sein, wie diese Vereinigung auszusehen hat.

### 4.3 DIE BEHANDLUNG VON AUSRUTSCHERN

Eine wesentliche Schwierigkeit beim Klassifizieren von Freihandskizzen besteht darin, Gewolltes von Ungewolltem, Absicht von Versehen, Wesentliches von Unwesentlichem zu trennen. So wird z.B. in einem langen, geraden Stroking ein Punkt, der ein wenig von der Ideallinie entfernt liegt, kaum eine wesentliche Information enthalten, während eine Pfeilspitze aus nur drei Punkten bestehen kann, und dort ist dann jeder einzelne Punkt entscheidend. Es bietet sich deswegen an, Abweichungen von einer gerade verfolgten

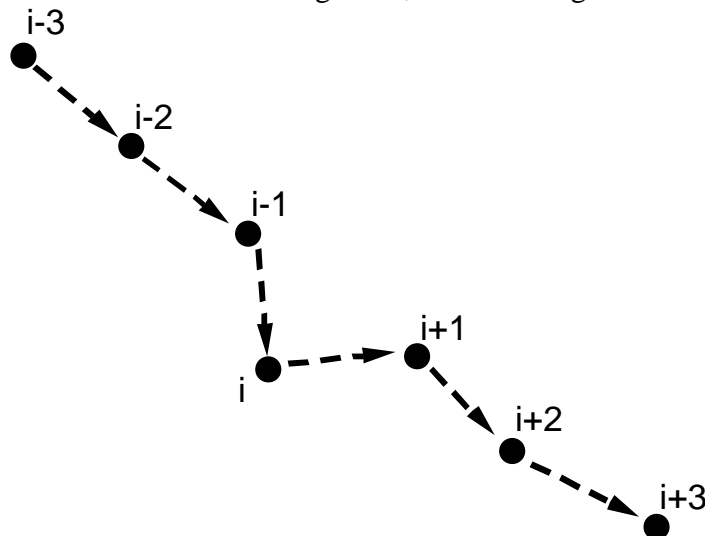


Abbildung 11: Kontextbetrachtung bei Ausrutschern

Zeichenrichtung als *Ausrutscher* aufzufassen.

Da die Abtastrate des Graphiktablets konstant ist, kann eine Betrachtung der Zeichengeschwindigkeit einigen Aufschluß bringen: Bei einer höheren Geschwindigkeit besteht ein Stroke aus weniger Punkten als ein Stroke von gleicher Ausdehnung bei einer niedrigeren Geschwindigkeit; und Abweichungen bei einer höheren Zeichengeschwindigkeit können eher als Ausrutscher interpretiert werden als Abweichungen derselben Größe bei einer niedrigeren Geschwindigkeit. Doch auch wenn die Betrachtung der Geschwindigkeit in diesem Zusammenhang ein lohnendes Forschungsfeld ist, so können wir sie hier nicht weiter in Betracht ziehen, weil wir für eine adäquate Behandlung zu weit von unserem eigentlichen Vorhaben abweichen müßten.

Noch viel wichtiger und hilfreicher als die Berücksichtigung der Geschwindigkeit ist aber eine Betrachtung des Kontextes. Nehmen wir das Beispiel aus Abbildung 11. Dort bilden die Punkte  $i$ ,  $i-1$  und  $i+1$  ein Dreieck, das man als Pfeilspitze interpretieren kann, oder auch als 'v', wenn man sich in einem Schriftkontext befindet. Nimmt man nun den hier vorhandenen Kontext und erweitert die Betrachtung auf  $i-2$  und  $i+2$ , hat man ein Konstrukt, das kaum noch an eine Pfeilspitze oder an ein Schriftsymbol erinnert. Statt dessen würde man eine Interpretation bevorzugen, die eine Strecke zwischen  $i-2$  und  $i+2$  erkennt. Der Punkt  $i$  würde als Ausrutscher aufgefaßt. Diese Interpretation erhält noch mehr Unterstützung, wenn man  $i-3$  und  $i+3$  hinzu nimmt.

Befänden sich aber noch Punkte zwischen  $i$  und  $i-1$  und zwischen  $i$  und  $i+1$ , was auch hieße, daß die Zeichengeschwindigkeit in diesem Bereich langsamer wäre – was man wiederum so interpretieren könnte, daß der Punkt  $i$  mit Absicht angesteuert wurde –, dann würde man eine Interpretation bevorzugen, welche die Ausbuchtung  $i$  nicht als Ausrutscher auffaßt.

## 4.4 ERKENNEN VON KREUZUNGEN

Das Beispiel, anhand dessen wir unsere Modellierung betrieben haben, ist eine prototypische Kreuzung, vgl. Abschnitt 5.1. Kreuzungen sind aus mehreren Gründen für uns von besonderem Interesse:

- Kreuzungen spielen in Wegeskizzen eine entscheidende Rolle; denn sie dienen als wichtige Orientierungspunkte. („Hier mußt Du Deine Richtung ändern.“ oder „Hier darfst Du Deine Richtung nicht ändern, obwohl es möglich ist.“)
- In dem speziellen Kontext der Produktion von Wegeskizzen sind Kreuzungen eigenständige graphische Objekte und nicht „nur“ Darstellungen von Realweltobjekten. Dies wirft die Frage auf, wie diese Art von graphischen Objekte zu behandeln sind und welche anderen Realweltobjekte ebenfalls korrespondierende graphische Objekte haben.
- Was *ist* eigentlich eine Kreuzung? Der Platz, den die sich kreuzenden Straßen gemeinsam haben? Dieser Platz plus das „Endstück“ der einmündenden Straßen? Oft wird eine Kreuzung auch nur auf die Fahrbahnen (für Fahrzeuge) bezogen, nicht auf die gesamten Straßen. Oder wie geht man mit einer Kreuzung um, die sich auf einem großen Platz befindet? Sicherlich wird man dann nicht den gesamten Platz als Kreuzung auffassen. Dieses Problem hat Auswirkungen auf die Konzeptualisierung einer Kreuzung. Die Konzeptualisierung aber ist eine Voraussetzung für die Verbalisierung, d.h. unterschiedliche Konzeptualisierungen erzeugen unterschiedliche Verbalisierungen.
- Schließlich muß man überlegen, ob und wie man die in Abschnitt 4.1 aufgeworfene Frage beantworten kann: Wann werden Kreuzungen als Kreuzungen und wann als

sich kreuzende Straßen interpretiert?

Um diesen Fragen näher zu kommen, haben wir die prototypische Kreuzung aus Abbildung 12 gewählt. Sie besteht aus acht Strecken (die Strecken 2 und 3 können im vorliegenden Kontext zusammengefaßt werden). Solch eine Kreuzung kann man dadurch als solche erkennen, daß man bei einer (beliebigen) Strecke (z.B. 5) beginnt und versucht, eine dazu parallele zu finden (6). Ist das erfolgreich, nimmt man die gefundene Strecke und versucht einen Winkel mit einer anderen Strecke zu bilden (7). – Ein Winkel besteht aus zwei Strecken, die sich an je einem ihrer Endpunkte berühren, und die nicht die gleiche Ausrichtung<sup>13</sup> haben. – Dann versucht man erneut eine Parallele (8) zu finden. Diesen Vorgang wiederholt man so lange, bis man wieder an der ersten Strecke angekommen ist, und mit der jeweils anderen Bedingung (hier: Winkel finden) abschließt, als man angefangen hat. Mit dieser Methode kann man diese Art von Kreuzungen erkennen.

Einer der nächsten Schritte unserer Modellierung wird sein, auch noch unvollständige Kreuzungen zu erkennen. Ab einem bestimmten Punkt ihrer Entstehung können Kreuzungen nämlich eindeutig als solche identifiziert werden.

## 5 Die Modellierung

### 5.1 DAS BEISPIEL FÜR DIE ENTWICKLUNG

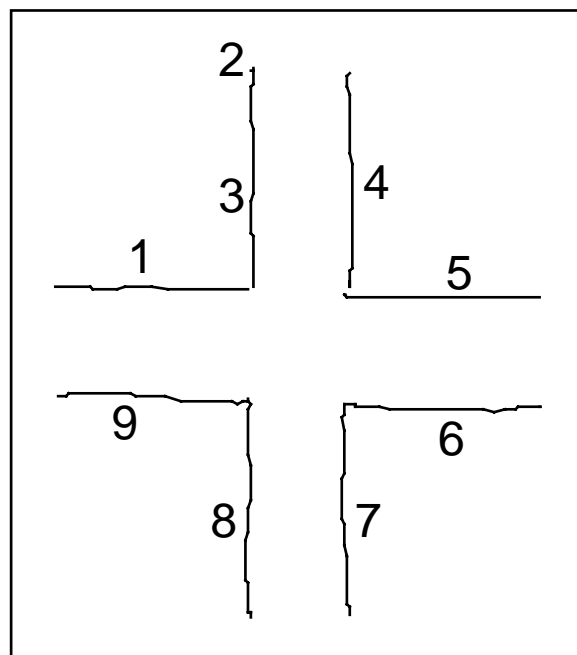


Abbildung 12: Die prototypische Beispielkreuzung

Anhand der prototypischen Kreuzung (vgl. Abbildung 12) haben wir genauer bestimmt, welche Eigenschaften und Fähigkeiten unsere Modellierung benötigt, und ein rudimen-

<sup>13</sup> Die Ausrichtung ist die Lage der Verbindungslinie zweier Koordinatenpunkte im Raum.

täres System zu konstruiert, das in der Lage ist, zumindest diese Kreuzung als eine solche zu erkennen. Die besprochenen möglichen Unterschiede in der Reihenfolge der Entstehung einer solchen Kreuzung führen schon jetzt zu unterschiedlichen Konzeptualisierungen. Auch Grundfragen der Segmentierung und Gruppierung müssen bereits sehr detailliert gelöst werden.

Die so entwickelten Fertigkeiten des Systems werden in späteren Projektabschnitten schrittweise weiter ausgebaut und erweitert. Dieses Vorgehen ist an ein Rapid-Prototyping Modell der Softwareentwicklung angelehnt. In unserer Variante bietet es den großen Vorteil, daß man sich auf die für die jeweilige Projektphase interessanten Aspekte der Modellierung konzentrieren kann.

Die Beispielkreuzung entsteht in neun Strokings, die hier zur einfacheren Bezugnahme mit Ziffern beschriftet sind, die in der Skizze nicht enthalten sind. Bis auf das Stroking 2 enden alle anderen in dem jeweiligen Eckpunkt. Das Stroking 2 besteht aus lediglich einem Punkt.<sup>14</sup> Darin zeigt sich bereits eines der in Abschnitt 3.2 dargestellten Probleme, nämlich, unter welchen Bedingungen zwei Strokings zu gruppieren sind; denn offensichtlich müssen die Strokings 2 und 3 zusammengefaßt werden – zumindest in dem Fall, wenn wir, wie im folgenden ausgeführt, diese Skizze auf das Vorhandensein von Strecken hin untersuchen. Auf der nächsten Stufe werden wir nach zusammenhängenden Paaren suchen, die jeweils aus einer horizontalen und einer vertikalen Strecke bestehen und die wir als *Winkel* bezeichnen. Schließlich überprüfen wir die Lage der Winkel zueinander und bestimmen so, ob es sich bei dem untersuchten Objekt um eine Kreuzung handelt.

Wie bereits bei der Erläuterung des Beispiels aus Abbildung 5.B dargestellt wurde, sind Pausen in Strokings, die sich leicht als eine Folge von identischen Koordinatenwerten identifizieren lassen, saliente Segmentierungspunkte – vor allem für Ecken. Aus diesem Grund stellen wir bei unserer Analyse für zwei identische, zeitlich direkt aufeinander folgende Koordinatenpunkte die Vermutung auf, daß sich hier eine Pause „ankündigt“, d.h. wir sehen diese Pause als eine Segmentgrenze an, an der, z.B. innerhalb eines Streckenzuges, zwei Teilstrecken aneinander stoßen. Es beginnt also ein neues *Segment*. Ändert sich dann auch noch die Richtung, wird die Vermutung bestätigt, und wir haben zwei Segmente eines Kurven- oder Streckenzuges erkannt; sonst muß die Vermutung zurückgewiesen werden, und die beiden Segmente, die durch die Vermutung erzeugt wurden, müssen zu einem zusammengefaßt werden.

## 5.2 ÜBERBLICK ÜBER DIE SYSTEMARCHITEKTUR

---

<sup>14</sup> Vermutlich ist dieser isolierte Punkt einer Fehlfunktion des Graphiktablets zuzuschreiben; doch das ändert nichts an der Tatsache, daß wir in der Lage sein müssen, solche Fälle zu behandeln; denn, da wir ein robustes, fehlertolerantes System anstreben, müssen wir sowohl Fehler der Hardware als auch Fehler in der Skizze selbst, also Zeichnungsfehler, ausgleichen können.

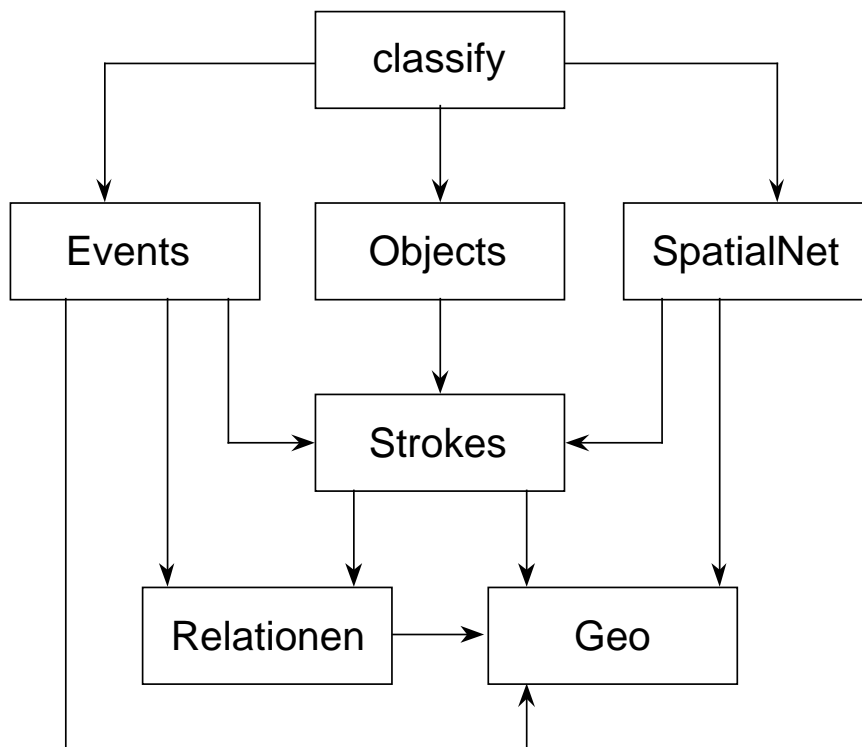


Abbildung 13: Architektur des Perl Programms

Die hier vorgestellten Arbeiten basieren auf Foth et al. (1999). Unsere Erweiterung besteht darin, daß wir die statische Analyse der fertigen Skizze eine dynamische Analyse der Skizzenentstehung erweitern.

Wir haben die Systemarchitektur so modifiziert wie es in Abbildung 13 dargestellt ist. Die Pfeile stellen Abhängigkeiten zwischen den einzelnen Modulen dar; in STROKES werden also beispielsweise Routinen aus GEO aufgerufen. CLASSIFY ist ein Programm, das von der Betriebssystemebene aufgerufen werden kann. Als Parameter wird dabei der Name der Datei angegeben, welche die Daten enthält, die mit dem Graphiktablett aufgezeichnet wurden.

CLASSIFY übergibt die Kontrolle an EVENTS, das die Daten solange Punkt für Punkt einliest, bis das Ende eines Strokings erreicht ist. Danach wird OBJECTS aufgerufen, das die Eigenschaften des gesamten Strokings bestimmt. In SPATIALNET werden zur Zeit im Rahmen einer Studienarbeit räumliche Netze implementiert. Räumliche Netze dienen dazu, graphische Objekte aufgrund ihrer räumlichen Lage zueinander in Beziehung zu setzen; das bietet den Vorteil, beim Ermitteln von Relationen zwischen Objekten den Suchraum einschränken zu können, was bislang nur unter Einbeziehung eines zeitlichen Kriteriums möglich ist, nämlich, indem man nur eine feste Anzahl (hier:10) der Vorgängerobjekte betrachtet.

Die drei Module EVENTS, OBJECTS und SPATIALNET greifen zur Klassifikation auf das Modul STROKES zu, in dem Funktionen bereitgestellt werden, mit deren Hilfe Eigenschaften eines Strokes bestimmt werden können.

GEO enthält Funktionen zur Bestimmung geometrischer Eigenschaften und RELATIONEN welche, um die Beziehungen zwischen je zwei Strokes festzustellen.

Bis auf EVENTS haben die meisten Module nur wenige Änderungen nach dem Ab-

schluß des Projektseminars erfahren. Die im Projektbericht bereits dokumentierten Eigenschaften werden wir an dieser Stelle deshalb lediglich kurz erwähnen. Der Source Code ist ausführlicher in einem eigenen Dokument für jedes Modul dargestellt.

Außer den in Abbildung 13 aufgeführten Modulen gibt es noch das Modul `DEBUGGER`, in dem man einen Debug-Level setzen kann, der die Anzahl von Statusmeldungen während der Programmausführung festlegt. Wie schon der Name sagt, dient es lediglich zum Debuggen während der Programmentwicklung und wird hier nicht weiter erwähnt.

## 5.3 DIE MODULE IM EINZELNEN (AUßER EVENTS)

### 5.3.1 Das Modul `CLASSIFY`

`CLASSIFY` übergibt zunächst die Kontrolle an `EVENTS`, das ein `Stroking` einliest und dieses als Ergebnis zurückgibt. Dann initiiert es die statische Analyse, indem es die Funktion `static` aus `OBJECTS` aufruft. Das passiert, solange sich noch Daten im Eingabestrom – das sind die Daten, die aus einer Skizzendatei ausgelesen werden – befinden.

### 5.3.2 Das Modul `OBJECTS`

`OBJECTS` ist ein kleines Modul, das für eine Liste von `Strokes` die Klassifikationsroutine `classify` aus `STROKES` aufruft, die ermittelten Werte ausgibt, dann die Relationen zwischen den `Strokes` berechnet und sie schließlich segmentiert.

### 5.3.3 Das Modul `SPATIALNET`

Das Modul `SPATIALNET` wird Datenstrukturen zur Verfügung stellen, die es ermöglichen, `Strokes` und graphische Objekte nicht aufgrund ihrer Entstehungsreihenfolge zu betrachten, sondern aufgrund ihrer räumlichen Lage zueinander. Die Funktion `GetNeighbours` soll die zu einem Objekt direkt benachbarten Items liefern. Mit der Funktion `GetInbetween` kann man die zwischen zwei Objekten liegenden Items errechnen und mit der Funktion `RangeSearch` hat man die Möglichkeit, sich Entitäten anzeigen zu lassen, die in einem bestimmten Radius liegen.

### 5.3.4 Das Modul `STROKES`

Den Kern des Moduls `STROKES` bildet die Funktion `classify`, in der die geometrischen Eigenschaften, wie etwa Größe, Schwerpunkt, Ausrichtung und Ecken, eines `Strokes` errechnet werden, und der `Stroke` wird einer Klasse, z.B. 'Kurvenzug', zugeordnet. Interessant ist noch die Routine `segment`, die eine Segmentierung eines `Strokes` vornimmt, d.h. sie unterteilt den `Stroke` an seinen Ecken in Segmente.

### 5.3.5 Das Modul `GEO`

`GEO` stellt zwei Arten von Routinen zur Verfügung: solche, um Abstände zwischen Objekten zu ermitteln, und solche, um die Lage von Strecken im Raum zu bestimmen. In die erste Kategorie fällt z.B. die Routine `point2point`, die den kartesischen Abstand zwischen zwei Punkten berechnet, in die zweite `Winkel`, welche die Lage einer Strecke relativ zur Abszisse bestimmt.

### 5.3.6 Das Modul `RELATIONEN`

In dem Modul `RELATIONEN` werden Funktionen bereitgestellt, um die folgenden geometrischen, räumlichen Beziehungen zwischen `Strokes` zu bestimmen, die aus Erichsen (1997, 80f) stammen:

- *nahe-bei*: Zwei Strokes stehen in der Relation *nahe-bei*, wenn sie nur einen geringen Abstand zueinander aufweisen.
- *innerhalb-von*: Ein Stroke befindet sich innerhalb eines anderen. Zu den genauen Bedingungen für diese Relationen – die u.a. von der Verwendung der Eigenschaft *geschlossen* abhängen – vgl. in Foth et al. (1999).
- *parallel-zu*: Strokes können nicht nur parallel zueinander sein, wenn sie gerade sind, sondern bei jeder beliebigen Form. Wir definieren *parallel-zu* allerdings nicht symmetrisch, weil Symmetrie nur bei der Betrachtung von (unendlichen) Geraden sinnvoll ist; wir aber untersuchen hier die Relationen zwischen (endlichen) Strecken, bzw. Linienzügen. Insbesondere kann ein längerer Linienzug nicht parallel zu einem kürzeren sein.
- *anschluß-an*: Ein Stroke befindet sich im *anschluß-an* einen anderen, wenn sie einen gemeinsamen Punkt<sup>15</sup> haben, sich aber nicht schneiden. Das ist insbesondere dann der Fall, wenn dieser gemeinsame Punkt der Endpunkt des ersten Strokes und der Anfangspunkt des zweiten ist.
- *fortsetzung-von*: Ein Stroke ist die *fortsetzung-von* einem anderen, wenn die beiden in der Relation *anschluß-an* stehen und sie außerdem dieselbe Ausrichtung haben.

## 5.4 DAS MODUL EVENTS

Das Modul EVENTS enthält die Teile des Systems, welche die dynamische Verarbeitung der Ereignisse vornehmen. Seine Aufgabe ist die Segmentierung und Strukturierung, also die beiden ersten Teilprozesse der Konzeptualisierung (s.o.). Das Modul EVENTS ist das modifizierte Modul FILES aus Foth et al. (1999). Dieses hatte die Aufgabe, die Graphiktablett-Dateien einzulesen und daraus eine Abfolge von Strokes und Pausen zu extrahieren.

Wir haben uns von der Untersuchung ganzer Strokings und Strokes gelöst und sind zu einer feineren Betrachtung übergegangen: Wir warten nicht mehr bis zum Ende eines Strokings, sondern bewerten die entstehenden Strukturen beim inkrementellen Einlesen eines Punktes sofort neu.

Ziel ist es, die gleichförmige „Vermehrung von Bildpunkten“ zu segmentieren, um so Segmente als Ereignisse individuieren zu können. Diese Segmente und die Pausen dazwischen werden als Ereignisprimitive in die hierarchische Ereignisstruktur „Sketch-Generating-Event“ aufgenommen. Ebenso werden die Segmente als Basiseinheiten der Objektstruktur „Segment“ festgehalten. Diese werden zu komplexeren Objekten gruppiert und tragen damit dazu bei, komplexere Ereignisstrukturen erkennen zu können (vgl. Abschnitt 4.1).

### 5.4.1 Die Hauptroutine `nextstroke`

Das Kernstück des Moduls EVENTS ist die Funktion `nextstroke`, die von `CLASSIFY` aus aufgerufen wird. Als erstes wird mit der Funktion `getPoint` ein Zahlentripel aus der Graphiktablett-Datei gelesen: die zu einem Abtastzeitpunkt gehörenden beiden Pixel-Koordinaten und der Stiftstatus. Dann erfolgt die eigentliche Verarbeitung dieses Tripels, wobei sowohl Objekt als auch Ereignisstrukturen aufgebaut werden. Das wird solange wiederholt, bis das Ende eines Strokings erreicht ist oder die Datei mit den Daten des Graphiktablets zu Ende ist. Dann gibt `nextstroke` die Kontrolle wieder an

<sup>15</sup> „Gemeinsamer Punkt“ heißt bei Freihandskizzen, die wir hier betrachten, immer, daß zwei Punkte entweder identisch, oder aber räumlich sehr nahe zueinander sind; dabei sind die Bedingungen für „sehr nahe“ Gegenstand einer künftigen Untersuchung.

das Hauptprogramm zurück.

Jedes neu gelesene Zahlentripel wird in Beziehung zu den bereits verarbeiteten Tripeln gesetzt. Ein Vergleich des aktuellen Stiftstatus zu dem des zuvor gelesenen Tripels führt zu einer Einordnung in einen von vier Bearbeitungsmodi (vgl. auch Tabelle 1):

- 1) der Stift war oben und bleibt oben: man befindet sich innerhalb einer Pause
- 2) der Stift war unten und bleibt unten: man befindet sich innerhalb eines Strokings
- 3) der Stift war unten aber ist jetzt oben: ein Stroking ist zu Ende
- 4) der Stift war oben aber ist jetzt unten: ein Stroking beginnt

Stift ist	oben	unten
Stift war		
oben	1	4
unten	3	2

**Tabelle 1: Die vier Bearbeitungsmodi von nextstroke**

### Modi 1 und 4

In Modus 1 wird die Routine `registerPause` aufgerufen, welche lediglich die Zeitintervalle zählt, während derer die Pause andauert. In Modus 4 ist gerade eine Pause zu Ende gegangen, die als abgeschlossenes Ereignis in die Ereignishierarchie eingetragen wird. Die für uns wesentliche Aufgabe der Segmentierung und Strukturierung findet in den beiden Modi 2 und 3 statt.

Um in dem „Strom der wahrgenommenen Eindrücke“ Ereignisse ausmachen zu können, ist es notwendig, Segmentgrenzen zu finden. Das augenfälligste Kriterium für eine Segmentgrenze liegt in Modus 3 vor und ist das Absetzen des Stiftes. Für den Beobachter einer Skizzenentstehung beginnt aber auch dann ein neues Ereignis, wenn innerhalb eines Strokings auf einer Stelle verharret wird und nach einiger Zeit – ohne Absetzen des Stiftes – weiter gezeichnet wird. Ebenso bildet eine Richtungsänderung innerhalb eines Strokings eine Segmentgrenze, selbst wenn sie nicht durch ein Verharren am Eckpunkt begleitet wird. Die letzten beiden Kriterien können im Modus 2 vorliegen.

### Modus 2

In Modus 2 wird geprüft, ob einer der beiden eben angesprochenen Fälle vorliegt, d.h. ob eine Pause innerhalb eines Strokings oder eine Richtungsänderung stattfindet. Zunächst wird getestet, ob sich der aktuelle Punkt vom vorherigen unterscheidet oder ob ein Verharren auf derselben Stelle, also eine Pause mit aufgesetzten Stift vorliegt. In diesem Falle ermittelt die Funktion `registerPause` die Zeitdauer dieser Pause. Da eine Segmentgrenze vorliegt, wird ein Eintrag in die Ereignishierarchie vorgenommen. Dazu machen wir die vereinfachende Annahme, daß – in der eingeschränkten Domäne der Kreuzungserkennung – in einem Stroking entweder eine vollständige Strecke entsteht (die als „Strecken-Entstehungs-Ereignis“ in die Ereignishierarchie eingetragen wird) oder Strukturen, die aus Strecken zusammengesetzt sind (und als „Strecken-zug-Entstehungs-Ereignis“ in die Ereignishierarchie eingetragen werden). Wird in Modus 2 (also innerhalb eines Strokings) eine Segmentgrenze festgestellt, kann es sich nicht mehr um ein einzelnes Strecken-Entstehungs-Ereignis handeln; es muß ein Strecken-zug-Entstehungs-Ereignis sein. Dabei kann das gerade beendete Subereignis den Anfang eines Strecken-zug-Entstehungs-Ereignisses bilden oder ein bereits begonnenes Strecken-zug-Entstehungs-Ereignis weiterführen. Im ersten Fall wird ein neues Hauptereignis „Strecken-zug-Entstehungs-Ereignis“ in die Ereignishierarchie eingefügt. Die eigentliche Eintragung des Segmentes als Subereignis des Strecken-zug-Entstehungs-Ereignisses in die Ereignishierarchie, sowie die Auswertung und Einordnung des Segmentes in die be-

stehenden Objektstrukturen findet in beiden Fällen in der Routine `registerSegment` statt, vgl. Abschnitt 5.4.2. Ebenso wie das Segment wird die Pause als nächstes Subereignis des Streckenzugs in die Ereignishierarchie eingetragen.

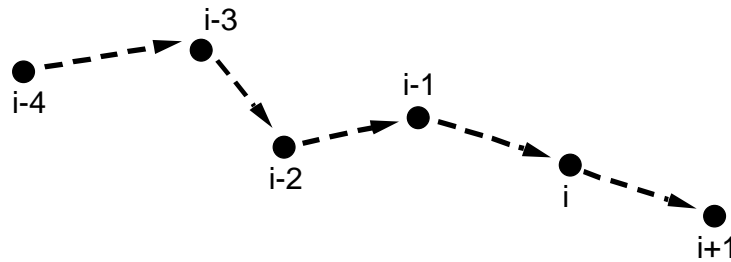


Abbildung 14: Eckenerkennung

Nach dem Test auf Pausen folgt der Test auf eine Richtungsänderung innerhalb des Strokings. Unter Annahme von geraden Segmenten geht es hier ausschließlich darum, eine Ecke zu erkennen. Hierzu vergleichen wir die Ausrichtung der Strecke zwischen dem aktuellen Punkt  $i$  und dem letzten Punkt  $i-1$  jeweils mit der Ausrichtung der Strecke zwischen  $i-1$  und  $i-2$ , zwischen  $i-2$  und  $i-3$  sowie zwischen  $i-3$  und  $i-4$ , vgl. Abbildung 14. Weicht die aktuelle Ausrichtung wesentlich von allen drei letzten Ausrichtungen ab, so haben wir die Erwartung, daß eine Ecke vorliegen könnte. Wird diese neue Ausrichtung – also die Strecke zwischen  $i$  und  $i-1$  – auch zwischen dem aktuellen und dem folgenden Punkt  $i+1$  beibehalten, so bestätigt sich unsere Erwartung und es liegt eine Ecke vor, und zwar am Punkt  $i-1$ . Kehrt die nächste Ausrichtung jedoch zu einer der drei letzten zurück, so handelt es sich nur um einen Ausrutscher. Wurde eine Ecke erkannt, so haben wir ebenfalls eine Segmentgrenze gefunden. Es findet ein Eintrag in die Ereignishierarchie statt, und das Segment wird in der Routine `registerSegment` ausgewertet und in die bestehenden Strukturen eingeordnet.

### Modus 3

In Modus 3 geht ein Stoking zu Ende, d.h. es liegt in jedem Fall eine Segmentgrenze vor. Dabei sind zwei Möglichkeiten zu unterscheiden: Gibt es außer Anfang und Ende keine weiteren Segmentierungspunkte, liegt ein Strecken-Entstehungs-Ereignis vor. Andernfalls handelt es sich um das letzte Subereignis eines Streckenzug-Entstehungs-Ereignisses. Die Eintragung des Segments in die Ereignishierarchie, sowie die Auswertung und Einordnung des Segmentes in die bestehenden Objektstrukturen findet in beiden Fällen in der Routine `registerSegment` statt.

Mit dem Ende eines Strokings wird die Einleseschleife von `nextstroke` beendet, abschließende Zuweisungen werden vorgenommen, z.B. werden die Datenstrukturen bereits für ein neues Stoking vorbereitet, und die Programmkontrolle wird an das Hauptprogramm, also das Modul `CLASSIFY` zurückgegeben.

#### 5.4.2 Die Routine `registerSegment`

In der Routine `registerSegment` werden abgeschlossene Segmente in die Datenstrukturen eingetragen. Zum einen werden sie in die hierarchische Ereignisstruktur „Sketch-Generating-Event“ eingeordnet, zum anderen als Element einer linearen Liste, der Objektstruktur „Segment“, angefügt. Diese Elemente bezeichnen wir im folgenden als Objekt-Segmente, kurz O-Segmente.

Beim Eintragen in die Ereignishierarchie wird das Segment entweder als Subereignis zu dem aktuellen Hauptereignis „Streckenzug“ eingegliedert, oder der bereits ange-

legte Schlüssel „Strecke“ wird mit den Werten des Stroking ergänzt, wenn es sich um ein vollständiges Stroking handelt. In der Objektstruktur wird das Segment unter einer fortlaufenden Nummer eingeordnet, und, um eine Zuordnung mit der Ereignisstruktur zu ermöglichen, wird diese Nummer beim zugehörigen Ereignis eingetragen.

Die genaue Evaluierung der Objektstruktur erfolgt in der Routine `evalSegment`. Hier werden die geometrischen Eigenschaften des O-Segments bestimmt, wofür die Funktion `classify` aus dem Modul STROKES zuständig ist (siehe Foth et al. (1999)). Des Weiteren werden O-Segmente, die stetig mit gleicher Ausrichtung fortgesetzt werden, zu einem Objekt zusammengefaßt. Dies betrifft vor allem den Fall, daß nach einer Pause in derselben Richtung weitergezeichnet wird. Die Pause wird als Segmentgrenze erkannt, und danach beginnt ein neues Ereignis. Für die Objektebene wird jedoch nur eine Strecke in der gleichen Richtung fortgesetzt; der Beobachter nimmt nicht zwei getrennte, sondern lediglich ein Objekt wahr. Deswegen findet hier eine Verschmelzung der beiden durch eine solche Pause getrennten O-Segmente statt, die zum Verlust der Einzelmerkmale der bisherigen O-Segmente führt.

Kriterium für diese Konkatenation ist eine starke räumliche Nähe zweier aufeinanderfolgender O-Segmente mit gleicher Ausrichtung. Hierbei werden auch O-Segmente zugelassen, die sich aus entgegengesetzten Richtungen treffen. Dies ist gerechtfertigt, weil es sich bei der Konkatenation lediglich um eine Veränderung der Objektstruktur, nicht aber der Ereignisstruktur handelt.

Die Zusammenfassung von O-Segmenten wird in der Subroutine `concatenate` durchgeführt. Dort werden die Merkmale des neuen Gesamt-O-Segmentes aus den Merkmalen der beiden Einzel-O-Segmente kombiniert.

#### 5.4.3 Die Routine `crossroadsDetection`

Nachdem wir mit der Bestimmung der O-Segmente die grundlegenden graphischen Bausteine bereitgestellt haben, können wir die Objekte zu komplexeren gruppieren. Die Routine `crossroadsDetection` befaßt sich mit der Gruppierung von O-Segmenten zu komplexeren Strukturen, die Objekte in der realen Welt – wie Straßen oder Straßenecken – repräsentieren. Wir machen bei der Betrachtung der Beispielkreuzung die Annahme, daß eine ideale Kreuzung aus acht O-Segmenten besteht. Auf einer höheren Granularitätsebene setzt sich eine ideale Kreuzung aus vier Winkeln – wie denjenigen aus Abbildung 5 – sowie aus vier Parallelen – das sind jeweils zwei parallele Strecken – zusammen. Sowohl ein Winkel als auch eine Parallele ist damit eine Gruppierung aus jeweils zwei O-Segmenten.

Die Routine `crossroadsDetection` versucht aus O-Segmenten Winkel und Parallelen aufzubauen. Ziel ist es, später diese höheren Strukturen wiederum zu gruppieren und schließlich eine Kreuzung daraus zusammensetzen zu können. Solange die räumlichen Netze noch nicht zur Verfügung stehen, halten wir uns an das Prinzip der lokalen Fortsetzung und gehen davon aus, daß zeitlich nahe beieinander liegende Strokes auch räumlich nahe beieinander liegen. Dies realisieren wir durch einen beschränkten Puffer mit den Eigenschaften eines Stacks, in dem sich die O-Segmente befinden, die noch keinen Partner für einen Winkel gefunden haben. Ein neu eingelesenes O-Segment versucht nun mit dem obersten (jüngsten) O-Segment auf dem Stack einen Winkel zu bilden. Gelingt dieses, so wird dieses oberste Stackelement vom Stack entfernt und steht folgenden O-Segmenten nicht mehr als Gruppierungspartner zur Verfügung. Kann das aktuelle O-Segment nicht mit dem obersten Stackelement gruppiert werden, wird eine Gruppierung mit dem nächsten (zweitjüngsten und so fort) Stackelement versucht. Ist es überhaupt nicht möglich, einen Partner für das aktuelle O-Segment

zu finden, wird es auf dem Stack abgelegt und ist damit neues oberstes Stackelement. Führt die Komplexbildung gleich mit dem obersten Stackelement zum Erfolg, haben wir eine Gruppierung von zwei aufeinander folgenden O-Segmenten. Ein Verbalisierer der Skizzenentstehung könnte dann z.B. äußern : „Es wird eine Straßenecke gezeichnet.“ Damit müßten also auch die beiden Entstehungsereignisse der v-Strecke und der h-Strecke zu einem neuen Ereignis „Eckenentstehung“ gruppiert werden. Wir haben in diesem Fall eine Korrespondenz zwischen Objekt- und Ereignisstruktur. Bis jetzt beschränken wir uns jedoch auf eine Betrachtung der Objektstrukturen und bilden keine Ereigniskomplexe. Die Ermittlung von Parallelen erfolgt analog.

## 6 Ausblick

Das in diesem Bericht Dargestellte bildet das Grundgerüst für den ersten Teil der computerlinguistischen Modellierung des Projektes. Damit haben wir die für den zweiten Teil notwendigen Datenstrukturen aufgebaut. Der nächste Schritt muß nun darin bestehen, diese Datenstrukturen in referentielle Netze zu übertragen, um dort die Bildung einer Gesamt-Sachverhaltsrepräsentation vorzunehmen. Der Hauptaspekt wird dabei der Aufbau von Ereignishierarchien sein; denn bislang haben wir lediglich Ereignisprimitive aufgezeichnet. Vor allem zwei Punkte sind entscheidend:

- der Aufbau von Erwartungshaltungen<sup>16</sup>, also die Entwicklung von Kriterien, unter welchen Umständen z.B. das Konzept KREUZUNG aktiviert wird. Dabei sind zwei Randbedingungen zu berücksichtigen:
  1. Erwartungshaltungen können enttäuscht werden, die Ereignisstruktur muß also revidiert werden können.
  2. Obwohl die Ereignishierarchie umstrukturiert werden kann, d.h. daß bestimmte Elemente gelöscht werden, während andere hinzukommen, können – zumindest beim menschlichen Erkennen – (Spuren von) Erinnerungen an frühere Ereignis- und Objektstrukturen vorhanden bleiben. Wir jedoch haben nur eine Ereignis- und Objektstruktur, können also solche „Erinnerungen an frühere Zustände“ nicht behalten.
- Außerdem müssen wir untersuchen, ob wir eine Modellierung der Ereignisvervollständigung wählen. Wenn also die Entstehung eines Objektes durch andere Entstehungsereignisse unterbrochen ist, haben wir zwei Möglichkeiten:
  - 1) Wir lassen zeitlich nicht zusammenhängende Entstehungsereignisse zu, nehmen also Entstehungsereignisse an, die aus Subereignissen bestehen, die durch andere Ereignisse unterbrochen werden können; sie wären also isomorph zu Entstehungsereignissen, die nicht unterbrochen werden.
  - 2) Ein Ereignis, das vervollständigt, was ein anderes Ereignis begonnen hat, wird nicht als „normales“ Ereignis konzeptualisieren, sondern als Vervollständigungsereignisse, d.h. daß es lediglich über einen Querverweis mit dem ersten Ereignis verbunden wird.

---

<sup>16</sup> Unter Erwartungshaltung verstehen wir eine Annahme darüber, wie – aufgrund der bisherigen Daten – die nächsten Daten des Eingabestroms aussehen werden, ohne daß wir sicher sein können, daß diese (zukünftigen) Daten unserer Annahme auch entsprechen.

## Anhang A. Die Skizze 10-3

---



---

### Abbildungsverzeichnis

Abbildung 1: Modularisierung der Sprachproduktionskomponenten nach Levelt (1989, 9)	3
Abbildung 2: Der Datenfluß im Gesamtsystem	5
Abbildung 3: Klassifikation von Strokes	8
Abbildung 4: Eine erstes Beispiel	9
Abbildung 5: Vier Möglichkeiten einen Winkel zu zeichnen	10
Abbildung 6: Eine mögliche Fortsetzung von Abbildung 5.B	10
Abbildung 7: Eine mögliche Fort- setzung von Abbildung 5.C	11
Abbildung 8: Kreuzende Straßen	13
Abbildung 9: Mögliche Ereignisstrukturen zu Abbildung 8	14
Abbildung 10: Datenstrukturen für Winkel mit Pausen der Abbildung 5.B und C	15
Abbildung 11: Kontextbetrachtung bei Ausrutschern	15
Abbildung 12: Die prototypische Beispielkreuzung	17
Abbildung 13: Architektur des Perl Programms	19
Abbildung 14: Eckenerkennung	23

## Literatur

- Erichsen, Mark. (1997) *Wissensbasierte Analyse des graphischen Inventars von Freihandzeichnungen*. Diplomarbeit. Technischer Bericht Nr.1 des Projektes „Konzeptualisierungsprozesse in der Sprachproduktion“. Universität Hamburg.
- Eschenbach, Carola. (1988) *SRL als Rahmen eines textverarbeitenden Systems*. GAP Arbeitspapiere 3. Hamburg.
- Foth, Kilian; Grochowina, Stephan; Turhan, Anni-Yasmin. (1999) *Abschlußbericht der Projektseminars „Konzeptualisierungsprozesse in der Sprachproduktion*. Seminarbericht im Arbeitsbereich Wissens- und Sprachverarbeitung des Fachbereichs Informatik an der Universität Hamburg.
- Guhe, Markus; Habel, Christopher; Tappe, Heike. (in Vorbereitung) *Interdependent Conceptualizations of Objects and Events: Natural Language Generation from Graphical Input*.
- Habel, Christopher. (1986) *Prinzipien der Referentialität: Untersuchungen zur propositionalen Repräsentation von Wissen*. Berlin: Springer.
- Habel Christopher. (1996) *Räumliche Repräsentationsformate – eine Fallstudie*. in: R. Meyer-Klabunde, C. von Stutterheim (Hrsg.) *Proceedings of a Workshop on Conceptual and Sematntuc Knowledge in Language Production*. Heidelberg/Mannheim; Arbeiten aus dem Sonderforschungsbereich 245 „Sprache und Situation“; Bericht Nr. 92; S. 57-86.
- Habel, Christopher; Tappe, Heike. (im Druck) *Processes of segmentation and linearisation in describing events*.
- Huber, Steffen; Foth, Kilian. (1998) *Repräsentation von Skizzen in SRL und Prolog*. Arbeitsbereich Wissens- und Sprachverarbeitung des Fachbereichs Informatik an der Universität Hamburg.
- Levelt, Willem J.M. (1989) *Speaking: From Intention to Articulation*. Cambridge, MA: MIT Press.
- Marcus, Mitchell P. (1980) *A Theory of Syntactic Recognition for Natural Language*. Cambridge, MA: MIT Press.
- Schwarz, Randal L. (1993) *Learning Perl*. Sebastopol, CA: O'Reilly.
- Tappe, Heike; Habel Christopher. (1998) *Verbalization of Dynamic Sketch Maps: Layers of Representation and their Interaction*. Langfassung der Posterpräsentation auf der Cognitive Science Conference 1998. *Verbalization of Dynamic Sketch Maps: Layers of Representation in the Conceptualization of Drawing Events*. Madison, Wisconsin, August 1998.
- Wall, Larry; Christiansen, Tom; Schwarz, Randal L. (1997) *Programmieren mit Perl*. Köln: O'Reilly.