| | |
|---|---|
| **Prüfer:** | Prof. Dr. Volker Claus |
| **Betreuer:** | Dipl.-Inf. Friedhelm Buchholz |

| | |
|---|---|
| **begonnen am:** | 4. Dez. 1996 |
| **beendet am:** | 3. Juni 1997 |
| **CR-Klassifikation:** | E.1, E.2, F.2.2, G.2.2, I.6 |

Diplomarbeit

# Verkehrssimulation

Axel Schwarz

Institut für Informatik
Universität Stuttgart
Breitwiesenstraße 20 – 22
D–70565 Stuttgart

# Vorwort

Simulation ist mit der heutigen Rechnerleistung ein geeignetes Mittel bei der Stadtplanung. Am Lehrstuhl Formale Konzepte von Prof. Dr. Volker Claus an der Universität Stuttgart wird ein Simulationssystem entwickelt, das dabei auftretende Fragestellungen prognostizieren kann. Die vorliegende Diplomarbeit, die unter der Betreuung von Dipl.-Inf. Friedhelm Buchholz entstand, ist ein Teilstück bei der Entwicklung dieses Simulationssystems.

Ich bedanke mich bei Prof. Dr. Volker Claus und Dipl.-Inf. Friedhelm Buchholz für die gute Betreuung.

Juni 1997,

Axel Schwarz

# Inhaltsverzeichnis

# Abbildungsverzeichnis

# Kapitel 1

# Einleitung

## 1.1 Einbettung der Arbeit

Diese Diplomarbeit ist Teil eines Forschungsprojektes, in dem ein Simulationssystem für den Individualverkehr und den öffentlichen Verkehr entwickelt wird.

Dazu wird das Verkehrsgeschehen modelliert, um Verkehrsstrukturen mit mathematischer Präzision zu erfassen und in einem Rechner verarbeiten zu können. Der Verkehr soll mit unterschiedlichen Bedingungen simuliert werden können, mit dem Ziel, verschiedene Fragestellungen der Stadtplanung zu prognostizieren.

**Beispiel 1.1:**

Eine Aufgabenstellung der Stadtplanung ist die Frage, wo eine neue U-Bahn-Linie sinnvoll sein könnte. [Richter97] beschreibt die Vorgehensweise bei der Planung einer neuen U-Bahn-Linie wie folgt:

**Zitat:**

> Ob eine neue U-Bahn-Linie sinnvoll ist oder nicht, hängt von mehreren Faktoren ab. Diese müssen eingehend untersucht und das Für und Wider einer solchen neuen U-Bahn-Linie muß gegeneinander abgewogen werden. So ist beispielsweise eine unerläßliche Voraussetzung das notwendige Fahrgastpotential und die technische Realisierungsmöglichkeit der gewählten Trassenführung. Weiterhin spielen wirtschaftliche Aspekte eine bedeutende Rolle. Nicht zuletzt ist die Finanzierbarkeit für die Einrichtung einer neuen U-Bahn-Linie maßgebend.
>
> Zusammenfassend möchte ich darauf hinweisen, daß für jede Maßnahme verkehrstechnischer Art eine sorgfältige Einzelfalluntersuchung erforderlich ist. Die für die jeweiligen Planungen maßgebenden örtlichen Rahmenbedingungen sind in der Regel historisch gewachsen und können deshalb nicht von Ort zu Ort übertragen werden. Allgemeingültige Aussagen können sich deshalb nur auf grobe Planungsprinzipien, niemals aber auf konkrete Lösungsmöglichkeiten beziehen.

Das Simulationssystem wird entwickelt, um die erwähnte Einzelfalluntersuchung zu unterstützen.

Im Rechner soll die Simulation auf der Datenstruktur THOR-Netz, einer Spezialisierung von Petrinetzen, durchgeführt werden, da hierfür ein bereits existierender Simulator von OFFIS (siehe [Wieting96] und [Schaal96]) eingesetzt werden kann.

Andererseits ist das Editieren von THOR-Netzen für den Anwender sehr unanschaulich. Daher sollen die Daten in einer benutzerfreundlicheren Repräsentation editiert werden können. Bei der Anfrage an das System, eine Simulation zu starten, soll dann automatisch die Repräsentation der Daten in die THOR-Netz-Struktur umgesetzt werden.

Dadurch gewinnt das Simulationssystem eine anwenderfreundliche Benutzeroberfläche, so daß ein marktfähiges Produkt entsteht.

## 1.2 Überblick über die Arbeit

In dieser Diplomarbeit werden in einem ersten Teil die für die interne Darstellung der Daten bei einer Simulation notwendigen Strukturen wie Petrinetze behandelt. Der zweite Teil entwickelt für die mathematische Repräsentation der Daten notwendige Modelle und Heuristiken und Algorithmen für eine Simulation.

Dabei wird das von [Schaal96] entworfene Modell um beliebige Verkehrsarten, insbesondere den Individualverkehr erweitert (Kapitel 4). Neue Aspekte sind ebenso die individuelle Steuerung einzelner Fahrer sowie die Einbeziehung außerplanmäßiger Ereignisse (Abschnitt 5.6).

Verschiedenartige Heuristiken (Abschnitt 5.3) für den Entscheidungsprozeß der Fahrer-Individuen sowie die Realisierung einer der Varianten in einem Algorithmus (Kapitel 6) stellen ein Kernstück der Arbeit dar.

Dieser Algorithmus erweitert den Dijkstra-Algorithmus für die Suche nach kürzesten Wegen in einem gerichteten Graphen. Bei der Bestwegsuche werden auch Wechsel der Verkehrsträger wie z. B. bei Park and Ride mit berücksichtigt. Ebenso wird auf verschiedene Varianten des Algorithmusses von Dijkstra für die Implementation eingegangen (Abschnitt 5.4).

Für das Gesamtverständnis der vorliegenden Diplomarbeit sind Vorkenntnisse nicht entscheidend.

# Teil I

# Allgemeine Strukturen

# Kapitel 2

# Mathematische Grundlagen

In diesem Kapitel werden die Grundbegriffe eingeführt, auf denen die in den folgenden Kapiteln besprochene Theorie aufbaut. Nach einem Abschnitt mit Begriffsklärungen folgt die Behandlung von Relationen und Wörtern.

## 2.1 Mathematische Symbole

Hier wird kurz die Bedeutung von Symbolen festgelegt, die in der Mathematik nicht einheitlich verwendet werden.

Mit „$\subset$“ wird die echte Teilmengenbeziehung zwischen zwei Mengen ausgedückt; die Gleichheit der Mengen wird dabei ausgeschlossen. Mit anderen Worten:

$$\forall A, B \text{ Mengen: } A \subset B \Rightarrow A \neq B.$$

Soll die Gleichheit der beteiligten Mengen nicht ausgeschlossen werden, wird in dieser Arbeit das Zeichen „$\subseteqq$“ verwendet in Analogon zum Zeichen „$\leqq$“ für reelle Zahlen.

Mit dem Zeichen „$\dot\cup$“ wird eine disjunkte Vereinigung angedeutet. Das heißt, daß die beteiligten Mengen leeren Schnitt haben. Aus der Schreibweise $M \dot\cup \{\infty\}$ folgt also, daß $\infty \notin M$.

Für die leere Menge schreibt man $\varnothing$. Das Symbol für die Potenzmenge einer Menge $M$ ist $\mathfrak{P}(M)$.

Mit $\mathbb{N}$ wird die Menge der natürlichen Zahlen bezeichnet. Die kleinste natürliche Zahl ist 1; $0 \in \mathbb{Z}$ ist in $\mathbb{N}$ nicht enthalten. Man definiert $\mathbb{N}_0 := \mathbb{N} \cup \{0\}$.

Sei nun $A$ eine Menge. Dann bezeichnet $|A|$ die Mächtigkeit dieser Menge. Man setzt $\aleph_0 := |\mathbb{N}|$. Das bedeutet, daß eine Menge $A$, für die $|A| < \aleph_0$ gilt, endlich ist.

Bei aussagenlogischen Formeln bedeutet „$\Leftrightarrow$“ die semantische Äquivalenz zweier Aussagen. Die textuelle Gleichheit zweier Aussagen wird mit „$\equiv$“ bezeichnet.

„Wahr“ und „falsch“ bezeichnen die beiden Wahrheitswerte. An Stelle von $x =$ wahr sagt man auch, x ist wahr, x gilt oder ähnliche sprachliche Formulierungen. Entsprechendes gilt auch für $x =$ falsch. „$=$ wahr“ kann auch weggelassen werden. Statt $x =$ falsch schreibt man auch $\neg x$.

## 2.2 Relationen und Abbildungen

**Definition 2.1:**

Seien $A$ und $B$ Mengen. R heißt eine Relation auf $(A, B)$, falls

$$\mathrm{R} \subseteqq A \times B.$$

**Definition 2.2:**

Seien $A$ und $B$ Mengen und R eine Relation auf $(A, B)$. Sei $(a, b) \in \mathrm{R}$. Dann sagt man: $a$ steht in Relation zu $b$.

Schreibweise: $a \,\mathrm{R}\, b$.

**Beispiel 2.3:**

„$\leqq$“ ist eine Relation auf $(\mathbb{Z}, \mathbb{Z})$.

**Definition 2.4:**

Sei $A$ eine Menge und R eine Relation auf $(A, A)$. Dann heißt R

(2.1) reflexiv, falls $\forall\, a \in A\colon\ a \,\mathrm{R}\, a$.

(2.2) symmetrisch, falls $\forall\, x, y \in A\colon\ (x \,\mathrm{R}\, y \Rightarrow y \,\mathrm{R}\, x)$.

(2.3) antisymmetrisch, falls $\forall\, x, y \in A\colon\ (x \,\mathrm{R}\, y \wedge y \,\mathrm{R}\, x \Rightarrow x = y)$.

(2.4) transitiv, falls $\forall\, x, y, z \in A\colon\ (x \,\mathrm{R}\, y \wedge y \,\mathrm{R}\, z \Rightarrow x \,\mathrm{R}\, z)$.

**Definition 2.5:**

Sei $A$ eine Menge und R eine Relation auf $(A, A)$. Dann heißt R eine Ordnungsrelation, falls R reflexiv, antisymmetrisch und transitiv ist. $(A, \mathrm{R})$ heißt dann eine Halbordnung.

**Definition 2.6:**

Sei $A$ eine Menge und R eine Relation auf $(A, A)$. $(A, \mathrm{R})$ sei eine Halbordnung. $(A, \mathrm{R})$ heißt eine Ordnung, falls je zwei Elemente aus $A$ vergleichbar sind, d. h. falls

$$\forall\, x, y \in A\colon\ (x \,\mathrm{R}\, y \vee y \,\mathrm{R}\, x).$$

**Definition 2.7:**

Sei $A$ eine Menge und R eine Relation auf $(A, A)$. Dann heißt R eine Äquivalenzrelation, falls R reflexiv, symmetrisch und transitiv ist.

**Definition 2.8:**
Sei $A$ eine Menge und R eine Äquivalenzrelation auf $(A, A)$. Sei $a \in A$. Dann heißt

$$[a] := \{x \in A \mid x \, \mathrm{R} \, a\}$$

die Äquivalenzklasse von $a$.

**Definition 2.9:**
Seien $A$ und $B$ Mengen und R eine Relation auf $(A, B)$. $f := (A, B, \mathrm{R})$ heißt eine Abbildung oder eine Funktion von $A$ in $B$, falls

(2.5) $\forall a \in A \, \exists b \in B\colon \ (a, b) \in \mathrm{R}$

(2.6) $\forall a \in A \, \forall b_1, b_2 \in B\colon \ ((a, b_1) \in \mathrm{R} \wedge (a, b_2) \in \mathrm{R} \ \Rightarrow \ b_1 = b_2)$

Statt $(a, b) \in \mathrm{R}$ schreibt man $f\colon \ a \mapsto b$ oder $b = f(a)$. Dann heißt $a$ Argument und $b$ Funktionswert an der Stelle $a$.
Statt $f = (A, B, \mathrm{R})$ schreibt man $f\colon \ A \to B, \forall a \in A\colon \ a \mapsto f(a)$.
R heißt der Graph von $f$; Bezeichnung: Graph$(f)$.
Mit $f\colon \ A \to B$ (ohne weitere Angabe) bezeichnet man eine Funktion $f = (A, B, R')$ für ein beliebiges geeignetes $R'$.

**Bemerkung 2.10:**
Die Bedingungen (2.5) und (2.6) stellen sicher, daß jedem Element aus $A$ ein und nur ein Element aus $B$ zugeordnet wird. Die Schreibweisen $f\colon \ a \mapsto b$ und $b = f(a)$ sind daher wohldefiniert.

**Definition 2.11:**
Seien $A$, $B_1$, $B_2$ und $C$ Mengen und $f\colon \ A \to B_1$ und $g\colon \ A \to B_2$ Funktionen. Seien $B_1 \subseteqq C$ und $B_2 \subseteqq C$. Sei R eine Relation auf $(C, C)$.
$f \, \mathrm{R} \, g \ :\Longleftrightarrow \ \forall a \in A\colon \ f(a) \, \mathrm{R} \, g(a)$

**Definition 2.12:**
Seien $A$ und $B$ Mengen und $f\colon \ A \to B$ eine Funktion von $A$ in $B$. Sei $b \in B$.
$f \equiv b \ :\Longleftrightarrow \ \forall a \in A\colon \ f(a) = b$

**Definition 2.13:**
Seien $A$ und $B$ Mengen und $f\colon \ A \to B$ und $g\colon \ A \to B$ Funktionen von $A$ in $B$.
$f \equiv g \ :\Longleftrightarrow \ \forall a \in A\colon \ f(a) = g(a)$

**Definition 2.14:**
Seien $A$ und $B$ Mengen und $f\colon \ A \to B$ eine Funktion von $A$ in $B$. Dann heißt $f$

(2.7) injektiv, falls $\forall a_1, a_2 \in A\colon \ (f(a_1) = f(a_2) \ \Rightarrow \ a_1 = a_2)$.
Bezeichnung: $f\colon \ A \rightarrowtail B$.

(2.8) surjektiv, falls $\forall b \in B \, \exists a \in A\colon \ b = f(a)$.
Bezeichnung: $f\colon \ A \twoheadrightarrow B$. $f$ heißt eine Funktion von $A$ auf $B$.

(2.9) bijektiv, falls $f$ injektiv und surjektiv ist.
Bezeichnung: $f\colon A \leftrightarrow B$.

**Definition 2.15:**

Seien $A$ und $B$ Mengen und $f\colon A \to B$ eine Funktion von $A$ in $B$. Sei $A' \subseteqq A$. Dann bezeichnet $f_{|A'}$ die Funktion $A' \to B$, für die gilt: $\forall\, a \in A'\colon\ a \mapsto f(a)$.
(Einschränkung im Definitionsbereich)

**Definition 2.16:**

Sei $B$ eine Menge. Sei

(2.10) $D = \varnothing$ und $k = 0$ oder

(2.11) $D = \{1, 2, \dots, k\}$ mit $k \in \mathbb{N}$ geeignet oder

(2.12) $D = \mathbb{N}$ und $k = \infty \notin \mathbb{N}_0$.

Eine Funktion $x\colon D \to B$ heißt eine Folge (in $B$).

Schreibweise: $x = (x_i)_{i \in D} = (x_i)_{i=1}^{k}$.

Sei $i \in D$. Statt $x(i)$ schreibt man auch $x_i$. $x_i$ heißt ein Folgenglied. $D$ heißt Indexmenge einer Folge. $i$ heißt Index in[1)] die Folge $x$.

**Definition 2.17:**

Sei $x\colon D \to B$ eine Folge. In den Fällen (2.10) und (2.11) heißt $x$ eine endliche Folge.
Schreibweise: $x = (x_i)_{i \in D} = (x_1, x_2, \dots, x_k) = (x_i)_{i=1}^{k}$.
Andernfalls heißt $x$ eine unendliche Folge.

**Bemerkung 2.18:**

Im Fall (2.10) entsteht die leere Folge. Nach obiger Definition der Schreibweise kann sie $(x_i)_{i=1}^{0}$ oder $(\,)$ geschrieben werden. Man beachte, daß es ein Folgenglied „$x_0$" nicht gibt.

## 2.3 Alphabete und Wörter

**Definition 2.19:**

Sei $A$ eine nichtleere Menge. $A$ heißt ein Alphabet, falls $A$ endlich ist, d. h.

$$|A| < \aleph_0.$$

Die Elemente von $A$ heißen Buchstaben.

[1)] Es liegt die Vorstellung zugrunde, daß der Index in die Folge hineinzeigt.

**Definition 2.20:**

Sei $A$ ein Alphabet. Sei $k \in \mathbb{N}_0$ und $x_1, x_2, \dots, x_k \in A$ Buchstaben. Die endliche Folge $(x_i)_{i=1}^k = (x_1, x_2, \dots, x_k)$ heißt Wort der Länge $k$ über dem Alphabet $A$.

Schreibweise: $x_1 x_2 x_3 \dots x_k$.

Die Menge aller Wörter über dem Alphabet $A$ wird mit $A^*$ bezeichnet.

Für $k = 0$ entsteht die leere Folge. Da zwei leere Folgen in einer Menge identisch sind, gibt es zu jedem Alphabet genau ein Wort der Länge 0.

**Definition 2.21:**

Sei $A$ ein Alphabet und $k = 0$. Die endliche Folge $(x_i)_{i=1}^k = (x_i)_{i=1}^0 = (\,)$ heißt das leere Wort über dem Alphabet $A$.

Schreibweise: $\varepsilon$.

**Beispiel 2.22:**

$A := \{1, 2, 3, 4, \diamond, \bullet\}$

Dann gilt: $\varepsilon$, 112, $12\diamond3\bullet\bullet4$, $\bullet\diamond \in A^*$, d. h. sind Wörter über dem Alphabet $A$.

**Definition 2.23:**

Sei $A$ ein Alphabet. Die Funktion

$$\begin{array}{lll} \text{„}\cdot\text{“:} & A^* \times A^* & \longrightarrow A^*, \\ & \forall (x_1x_2\dots x_n, y_1y_2\dots y_m) \in A^* \times A^*: & \\ & (x_1x_2\dots x_n, y_1y_2\dots y_m) & \longmapsto \cdot(x_1x_2\dots x_n, y_1y_2\dots y_m) \\ & & := x_1x_2\dots x_n \cdot y_1y_2\dots y_m \\ & & := x_1x_2\dots x_n y_1y_2\dots y_m, \end{array}$$

wobei $n, m \in \mathbb{N}_0$ geeignet, heißt Konkatenation.

**Beispiel 2.24:**

$A := \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

Damit gilt nun mit „ $\cdot$ “ als Konkatenation

$$\begin{aligned} 70 \cdot 565 &= 70565 \\ 1 \cdot 00 \cdot 1 &= 1001 \\ 4 \cdot \varepsilon &= 4 \\ \varepsilon \cdot \varepsilon &= \varepsilon \end{aligned}$$

Die Ziffern sind hier als Buchstaben des Alphabets $A$ zu lesen.

**Definition 2.25:**
Sei $M$ eine Menge, $e \in M$ und „$\circ$“: $M \times M \to M$. $(M, „\circ“, e)$ heißt eine Halbgruppe mit Einselement oder ein Monoid, falls gilt:

(2.13) „$\circ$“ ist assoziativ, d. h. $\forall\, x, y, z \in M\colon\ (x \circ y) \circ z = x \circ (y \circ z)$

(2.14) $\forall\, x \in M\colon\ e \circ x = x \circ e = x$

**Beispiele 2.26:**

1.) $(\mathbb{N}_0, +, 0)$ ist ein Monoid.

2.) Sei $A$ ein Alphabet und „$\circ$“ die Konkatenation. Dann ist $(A, „\circ“, \varepsilon)$ ein Monoid.

**Definition 2.27:**
Sei $A$ ein Alphabet und „$\circ$“ die Konkatenation. Das Monoid $(A, „\circ“, \varepsilon)$ heißt das freie Monoid über $A$.

# Kapitel 3

# Transitionen

Petrinetze bilden die Datenstruktur, auf der ein Rechner die Verkehrssimulation durchführen wird.

Auf eine kurze Einführung in Graphen und die Behandlung von Petrinetzen folgt im Abschnitt Transitionssysteme eine Verallgemeinerung des Begriffs der Transition. Dadurch wird die theoretische Grundlage für das Simulationssystem bereitgestellt.

## 3.1 Petrinetze

### 3.1.1 Graphen

**Definition 3.1:**

Seien $V$ und $E$ Mengen. $(V, E)$ heißt ein Graph oder ein gerichteter Graph, falls

$$E \subseteqq V \times V.$$

Die Elemente von $V$ heißen Knoten.
Die Elemente von $E$ heißen Kanten.

**Bemerkung 3.2:**

$E$ ist eine Relation auf $(V, V)$.

**Definition 3.3:**

Sei $\mathfrak{G} = (V, E)$ ein Graph. $\mathfrak{G}$ heißt ein ungerichteter Graph, falls $E$ symmetrisch ist, d. h. falls

$$\forall (x, y) \in E\colon\ (y, x) \in E.$$

**Definition 3.4:**
Sei $\mathfrak{G} = (V, E)$ ein Graph. Eine Funktion C: $V \times V \to \mathbb{R}\dot{\cup}\{\infty\}$ heißt eine Bewertung des Graphen, falls
(3.1) $\forall v \in V\colon\ \mathrm{C}(v, v) = 0$
(3.2) $\forall v_1, v_2 \in V\colon\ ((v_1, v_2) \in E \Leftrightarrow \mathrm{C}(v_1, v_2) \neq \infty)$

**Definition 3.5:**
Sei $\mathfrak{G} = (V, E)$ ein Graph. Seien $V_1, V_2 \subseteqq V$. $\mathfrak{G}$ heißt $(V_1, V_2)$-bipartit, falls

$$\begin{array}{ll} & (V_1 \neq \varnothing \wedge V_2 \neq \varnothing \wedge V_1 \cup V_2 = V \wedge V_1 \cap V_2 = \varnothing) \\ \wedge & \forall (x, y) \in E\colon\ [(x \in V_1 \wedge y \in V_2) \vee (x \in V_2 \wedge y \in V_1)]. \end{array}$$

Das bedeutet, daß die Menge der Knoten in zwei Klassen $V_1$ und $V_2$ zerfällt, und es zwischen jeweils zwei Knoten aus derselben Klasse keine Kante gibt.

### 3.1.2 Definition

**Definition 3.6:**
$(S, T, F, K, W, M_0)$ heißt Petrinetz oder Stellen-Transitions-Netz (S/T-Netz), falls gilt:
(3.3) $S$ und $T$ sind endliche Mengen: $|S| < \aleph_0$, $|T| < \aleph_0$.
(3.4) $(S \cup T, F)$ ist ein $(S, T)$-bipartiter Graph.
(3.5) $K\colon\ S \to \mathbb{N} \dot{\cup} \{\infty\}$
(3.6) $W\colon\ F \to \mathbb{N}$
(3.7) $M_0\colon\ S \to \mathbb{N}_0$ mit $M_0 \leqq K$.
Die Elemente von $S$ heißen Stellen; Symbol: Kreis.
Die Elemente von $T$ heißen Transitionen; Symbol: Quadrat.
$F$ heißt Flußrelation; Symbol: Pfeil.
$K$ heißt Kapazitätsfunktion; für ein $s \in S$ heißt $K(s)$ Kapazität der Stelle $s$; Symbol: Zahl an der Stelle.
$W$ heißt Gewichtsfunktion (des Flusses); für ein $f \in F$ heißt $W(f)$ Gewicht der Kante $f$; Symbol: Zahl an der Kante.
$M_0$ heißt Anfangsbelegung oder Anfangsmarkierung; Symbol: Zahl in der Stelle.

**Definition 3.7:**
Sei $(S, T, F, K, W, M_0)$ ein Petrinetz. Eine Funktion $M\colon\ S \to \mathbb{N}_0$ mit $M \leqq K$ heißt eine Belegung oder eine Markierung des Petrinetzes.
Für ein $s \in S$ sagt man: Auf der Stelle $s$ liegen (oder befinden sich) $M(s)$ Marken.
Symbol: Zahl in der Stelle.

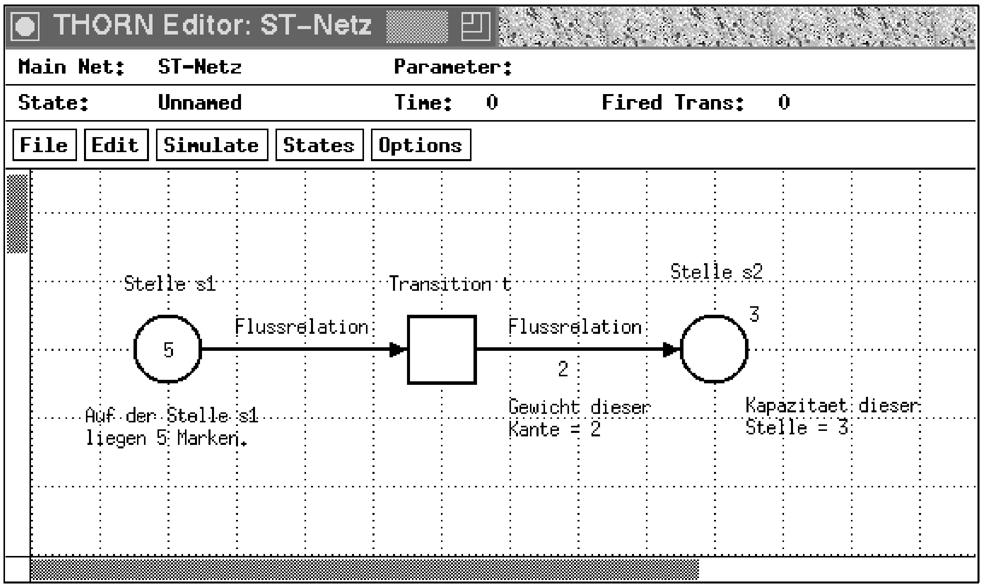



Abbildung 3.1: Symbole im Petrinetz

**Vereinbarungen 3.8:**

1.) Wenn in einem Petrinetz-Schaubild bei einer Stelle keine Kapazität angegeben ist, dann sei die Kapazität dieser Stelle $= \infty$. Man sagt, die Kapazität dieser Stelle ist unbeschränkt oder unendlich.

2.) Wenn in einem Petrinetz-Schaubild bei einer Kante kein Gewicht angegeben ist, dann sei das Gewicht dieser Kante $= 1$.

3.) Wenn in einem Petrinetz-Schaubild bei einer Stelle keine Anzahl an Marken eingetragen ist, bei anderen Stellen aber schon, dann befinde sich keine Marke auf dieser Stelle.

4.) Wenn in einem Petrinetz-Schaubild bei keiner Stelle eine Anzahl an Marken eingetragen ist, wird keine Aussage über eine Belegung mit Marken gemacht.

5.) In der Bedingung (3.7) versteht sich in der Bezeichnung „$M_0 \leqq K$“ die Relation „$\leqq$“ auf $\infty$ fortgesetzt, so daß $\forall\, n \in \mathbb{N}$: $n \leqq \infty$.

Analog in Definition 3.7.

### 3.1.3 Schaltregel

**Definition 3.9:**

Sei $(S, T, F, K, W, M_0)$ ein Petrinetz. Sei $x \in S \cup T$ und $U \subseteqq S \cup T$. Dann heißt

(3.8) $\bullet x := \{y \mid (y, x) \in F\}$ Vorbereich von $x$.

(3.9) $x\bullet := \{y \mid (x, y) \in F\}$ Nachbereich von $x$.

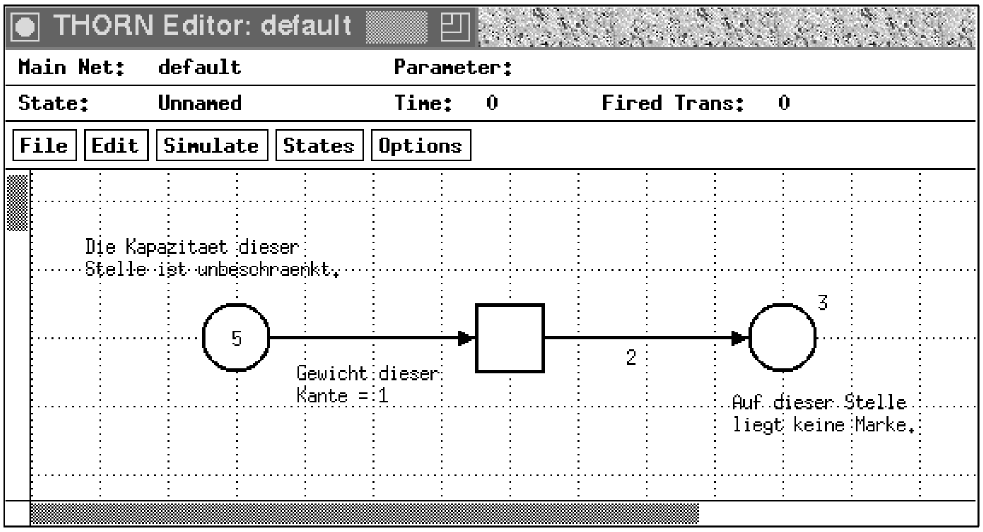



Abbildung 3.2: Standardbedeutungen im Petrinetz

(3.10) $\bullet U := \bigcup_{x \in U} \bullet x$ Vorbereich von $U$.

(3.11) $U\bullet := \bigcup_{x \in U} x\bullet$ Nachbereich von $U$.

**Definition 3.10:**

Sei $(S, T, F, K, W, M_0)$ ein Petrinetz. Sei $M$ eine Markierung und $t \in T$ eine Transition. $t$ heißt stark aktiviert unter der Markierung $M$, falls

(3.12) $\forall\, s \in \bullet t\colon\ W(s,t) \leqq M(s)$

(3.13) $\forall\, s \in t\bullet\colon\ M(s) + W(t,s) \leqq K(s)$

**Definition 3.11:**

Sei $(S, T, F, K, W, M_0)$ ein Petrinetz. Sei $M$ eine Markierung und $t \in T$ eine Transition. $t$ heißt aktiviert oder schwach aktiviert unter der Markierung $M$, falls

(3.14) $\forall\, s \in \bullet t\colon\ W(s,t) \leqq M(s)$

(3.15) $\forall\, s \in t\bullet \setminus \bullet t\colon\ M(s) + W(t,s) \leqq K(s)$

(3.16) $\forall\, s \in \bullet t \cap t\bullet\colon\ M(s) - W(s,t) + W(t,s) \leqq K(s)$

**Lemma 3.12:**

Sei $(S, T, F, K, W, M_0)$ ein Petrinetz. Sei $M$ eine Markierung und $t \in T$ eine Transition. Wenn $t$ stark aktiviert ist unter der Markierung $M$, dann ist $t$ auch schwach aktiviert unter der Markierung $M$.

**Beweis:** folgt sofort aus der Bedingung (3.13), q. e. d.

**Definition 3.13:**

Sei $(S, T, F, K, W, M_0)$ ein Petrinetz. Sei $M$ eine Markierung und $t \in T$ eine Transition. $t$ sei schwach aktiviert unter der Markierung $M$. Durch das Schalten oder Feuern von $t$ entsteht eine neue Markierung $M'$, die sogenannte Folgemarkierung. Dabei ist $M'$ folgendermaßen charakterisiert:

$$M'(s) = \begin{cases} M(s) & \text{für } s \notin \bullet t \cup t\bullet \\ M(s) - W(s,t) & \text{für } s \in \bullet t \setminus t\bullet \\ M(s) + W(t,s) & \text{für } s \in t\bullet \setminus \bullet t \\ M(s) - W(s,t) + W(t,s) & \text{für } s \in \bullet t \cap t\bullet \end{cases}$$

Bezeichnung: $M\ [t\rangle\ M'$.

**Bemerkung 3.14:**

Der starken Aktiviertheit liegt ein „zeitloses" Schaltverhalten zugrunde; der schwachen Aktiviertheit die Vorstellung, daß beim Schalten zuerst die Marken abgezogen werden, um dann in der Transition verarbeitet zu werden.

Diese Definition arbeitet mit der schwachen Aktiviertheit, da dies der Schaltregel im später eingesetzten Petrinetzsimulator (THOR) entspricht.

**Lemma 3.15:**

Das „$-$" in der Definition 3.13 ist wohldefiniert, d. h. die Operation führt aus dem Wertebereich von $M'$, das ist $\mathbb{N}_0$, nicht heraus.

**Lemma 3.16:**

Sei $(S, T, F, K, W, M_0)$ ein Petrinetz. Sei $M$ eine Markierung und $t \in T$ eine Transition. $t$ sei schwach aktiviert unter der Markierung $M$. Sei $M\ [t\rangle\ M'$. Dann ist $M'$ auch eine Markierung, d.h. $M' \leqq K$.

**Beweise:** folgen sofort aus der Definition 3.11 der schwachen Aktiviertheit, q. e. d.

**Definition 3.17:**

Sei $(S, T, F, K, W, M_0)$ ein Petrinetz. Sei $D$ Indexmenge[1)] einer Folge. Eine Folge $\mathfrak{M} = (M_i)_{i \in D}$ von Markierungen heißt ein Lauf oder eine Berechnung des Petrinetzes, falls

(3.17) $\quad D \neq \varnothing \wedge M_1 = M_0$

(3.18) $\quad \forall\, i \in D$: [ $\quad (\exists\, t \in T$: $t$ ist aktiviert unter $M_i)$

$\qquad\qquad \Rightarrow (i+1 \in D \wedge \exists\, t \in T,\ t$ ist aktiviert unter $M_i$: $M_i\ [t\rangle\ M_{i+1})$ ]

(3.19) $\quad (\exists\, k \in D\, \forall\, t \in T$: $t$ ist nicht aktiviert unter $M_k) \Rightarrow D = \{1, 2, \ldots, k\}$

Falls $|D| < \aleph_0$, heißt $\mathfrak{M}$ terminierende Berechnung des Petrinetzes. Andernfalls heißt $\mathfrak{M}$ unendliche Berechnung.

Ein Index $i \in D$ in die Folge $\mathfrak{M}$ heißt Zeitpunkt.

---

1) siehe Definition 2.16

**Satz 3.18:**
Jedes Petrinetz besitzt eine Berechnung.

**Beweis:** Sei $\mathfrak{N} = (S, T, F, K, W, M_0)$ ein Petrinetz. Definiere eine Folge von Markierungen $\mathfrak{M}' = (M_i)_{i \in \mathbb{N}}$ wie folgt:

(1) $M_1 := M_0$

(2) Sei $n \in \mathbb{N}$, $n > 1$.

*1. Fall:* $\exists\, t \in T$: *$t$ ist aktiviert unter $M_{n-1}$.* Definiere $M_n$ so, daß $M_{n-1}\,[t\rangle\, M_n$ mit einem aktivieten $t \in T$.

*2. Fall:* $\forall\, t \in T$: *$t$ ist nicht aktiviert unter $M_{n-1}$.* $M_n := M_0$.[2)]

Definiere eine Teilfolge $\mathfrak{M}$ von $\mathfrak{M}'$ wie folgt:
*1. Fall:* $\forall\, i \in \mathbb{N}\, \exists\, t \in T$: *$t$ ist aktiviert unter $M_i$.* $\mathfrak{M} := \mathfrak{M}'$.
*2. Fall: sonst.* Sei $k \in \mathbb{N}$ die kleinste Zahl, für die gilt: $\forall\, t \in T$: $t$ ist nicht aktiviert unter $M_k$. $\mathfrak{M} := (M_i)_{i \in \{1,\ldots,k\}}$.
Dann ist $\mathfrak{M}$ eine Berechnung von $\mathfrak{N}$, da für $\mathfrak{M}$ die Bedingungen (3.17), (3.18) und (3.19) nach Konstruktion erfüllt sind, q. e. d.

**Beispiel 3.19:**
Das Petrinetz in Abbildung 3.3 stellt ein Erzeuger-Verbraucher-System dar.
Dies ist ein Petrinetz $(S, T, F, K, W, M_0)$ mit
$S = \{s_1, s_2, s_3, s_4, s_5, s_6, s_7\}$
$T = \{t_1, t_2, t_3, t_4, t_5\}$
$F = \{(s_1, t_2),\ (t_2, s_2),\ (s_2, t_1),\ (t_1, s_1),\ (t_2, s_3),\ (t_2, s_4),\ (s_4, t_3),\ (t_3, s_5),\ (s_5, t_4),\ (t_4, s_6),\ (s_6, t_5),\ (t_5, s_7),\ (s_7, t_3)\}$
$K$: $s_1 \mapsto 1$, $s_2 \mapsto 1$, $s_3 \mapsto \infty$, $s_4 \mapsto 5$, $s_5 \mapsto 2$, $s_6 \mapsto 2$, $s_7 \mapsto 1$.
$W(t_2, s_4) = 3$, $W(s_4, t_3) = 2$; allen anderen Elementen aus $F$ ordnet $W$ das Gewicht 1 zu.
$M_0$: $s_1 \mapsto 1$, $s_2 \mapsto 0$, $s_3 \mapsto 0$, $s_4 \mapsto 0$, $s_5 \mapsto 0$, $s_6 \mapsto 2$, $s_7 \mapsto 0$. Die Bedingung $M_0 \leqq K$ ist erfüllt.
Zu Beginn sind die Transitionen $t_2$ und $t_5$ (sogar stark) aktiviert, da sich in deren Vorbereich genügend Marken befinden, und die Kapazität der Nachbereiche auch mit den durch Schalten von $t_2$ bzw. $t_3$ entstehenden Marken noch nicht erschöpft ist.

**Interpretation des Petrinetzes in bezug auf die Situation des Erzeuger-Verbraucher-Systems**
Steht in der folgenden Tabelle in der linken Spalte der Name einer Stelle, so wird die Bedeutung der Marken auf dieser Stelle interpretiert; steht dort der Name einer Transition, so wird – für den Fall, daß sie aktiviert ist – das Schalten dieser Transition gedeutet.

[2)] Der Wert spielt keine Rolle.

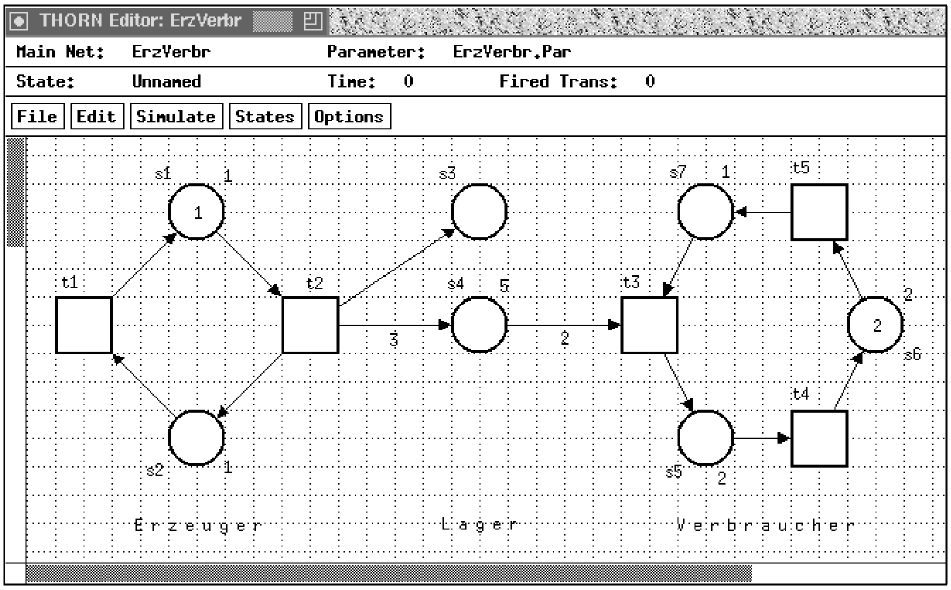



Abbildung 3.3: Erzeuger-Verbraucher-System

$s_1$: Bereitschaft zu produzieren.

$t_2$: Produktion. Es entstehen 3 Objekte (Marken), die ins Lager ($s_4$) gelegt werden. Der Erzeuger ist zur Zeit nicht in der Lage, erneut zu produzieren, angedeutet durch die Marke auf $s_2$. Außerdem wird der Produktionszähler $s_3$ um 1 weitergeschaltet.

$s_2$: Der Erzeuger hat Wartung nötig.

$t_1$: Der Erzeuger wird gewartet, z. B. durch Nachfüllen von Material.

$s_3$: Produktionszähler.

$s_4$: Lager. Die maximale Lagerkapazität beträgt 5 Objekte. Reicht die Kapazität nicht aus, um alle erzeugten Objekte aufzunehmen, kann $t_2$ nicht schalten. Sind weniger als 2 Objekte im Lager, kann $t_3$ nicht schalten.

$s_6$: Jede Marke repräsentiert einen Verbraucher. Auf $s_6$ wird die Bereitschaft angezeigt, ein Objekt zu verbrauchen. Durch $K(s_7) = 1$ soll angedeutet werden, daß immer nur ein Verbraucher auf einmal verbrauchen kann. Die Verbraucher halten sich in $s_6$ zur Verfügung, in die Warteschlange (in diesem Beispiel eine Stelle lang) aufgenommen zu werden.

$t_5$: Aufnahme in die Warteschlange.

$s_7$: Das Objekt, das als nächstes an die Reihe kommt.

$t_3$: Verbrauch. Es werden 2 Objekte aus dem Lager geholt, die verbraucht werden. Der Verbraucher ist danach nicht in der Lage, erneut zu verbrauchen, angedeutet durch die Marke auf $s_5$.

$s_5$: Der Verbraucher hat Wartung nötig. (Die Kapazitätsbeschränkung auf 2 Marken ist ohne Bedeutung, da sich aufgrund der Konstruktion des Netzes nie mehr als 2 Marken im Verbraucherkreis befinden können.)

$t_4$: Der Verbraucher wird gewartet, z. B. durch Putzen.

**Bemerkung 3.20:**
Wird die Definition 3.6 so abgewandelt, daß $K\colon\ S \to \{\infty\}$, d. h. die Kapazität aller Stellen unbeschränkt ist, dann ist die entstehende Definition eines Petrinetzes gleichmächtig zur Definition 3.6, da die Kapazität auch über sogenannte komplementäre Stellen modelliert werden kann. Auf dieser komplementären Stelle liegen so viele Marken, wie die ursprüngliche Stelle mit beschränkter Kapazität freie Plätze hat. Für ein Beispiel siehe Abbildung 3.4.
Die Kapazität dient also dazu, Petrinetze übersichtlicher gestalten zu können.

### 3.1.4 Berechnungssituationen

**Definition 3.21:**
Sei $(S, T, F, K, W, M_0)$ ein Petrinetz und $t \in T$. $t$ heißt lebendig, falls

$$\begin{array}{l} \forall (M_i)_{i\in D} \text{ Berechnung } \forall\, k \in D \\ \exists ((\tilde{M}_j)_{j\in\tilde{D}} \text{ Berechnung, } \tilde{D} \supset D,\ \forall\, i \leqq k\colon\ M_i = \tilde{M}_i) \\ \exists\, k' \in \tilde{D},\ k' > k\colon \\ t \text{ ist aktiviert unter der Markierung } \tilde{M}_{k'}. \end{array}$$

Andernfalls heißt $t$ tot.

**Bemerkung 3.22:**
Die Eigenschaft, eine Berechnung zu sein, hängt induktiv nur vom vorherigen Index ab. Statt die Existenz einer Berechnung mit gleichem Berechnungsanfang zu fordern, kann daher auch eine neue Berechnung mit $M_k$ als erstem Folgenglied gestartet werden.

**Definition 3.23:**
Sei $\mathfrak{N} = (S, T, F, K, W, M_0)$ ein Petrinetz. $\mathfrak{N}$ heißt lebendig, falls

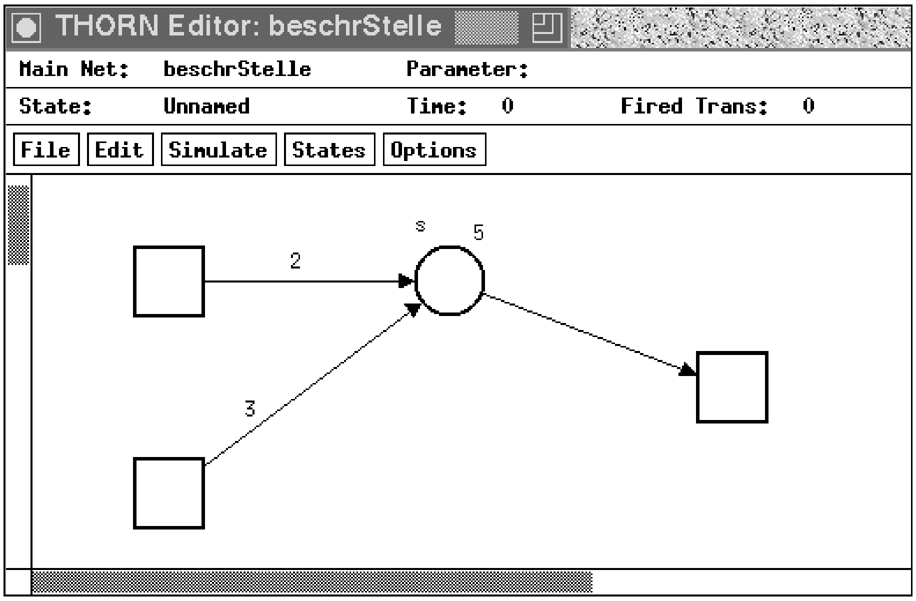
$$\forall\, t \in T\colon\ t \text{ ist lebendig.}$$

**Satz 3.24:**
Sei $\mathfrak{N} = (S, T, F, K, W, M_0)$ ein Petrinetz. Wenn $\mathfrak{N}$ lebendig ist, dann existiert eine unendliche Berechnung des Petrinetzes.

**Beweis:** Sei $\mathfrak{N}$ ein lebendiges Petrinetz. Sei $t \in T$. $t$ ist lebendig.

(1) Nach Satz 3.18 existiert eine Berechnung des Petrinetzes.

(2) Sei $(M_i)_{i\in D}$ eine Berechnung des Petrinetzes. Falls $|D| = \aleph_0$, also $D = \mathbb{N}$, ist die gesuchte Berechnung gefunden. Andernfalls kann durch die Wahl $k = |D|$ in Definition 3.21 eine Berechnung mit mächtigerer Indexmenge $\tilde{D}$ gefunden werden.




Mit Hilfe von komplementären Stellen kann auf die Verwendung von Kapazitäten verzichtet werden:

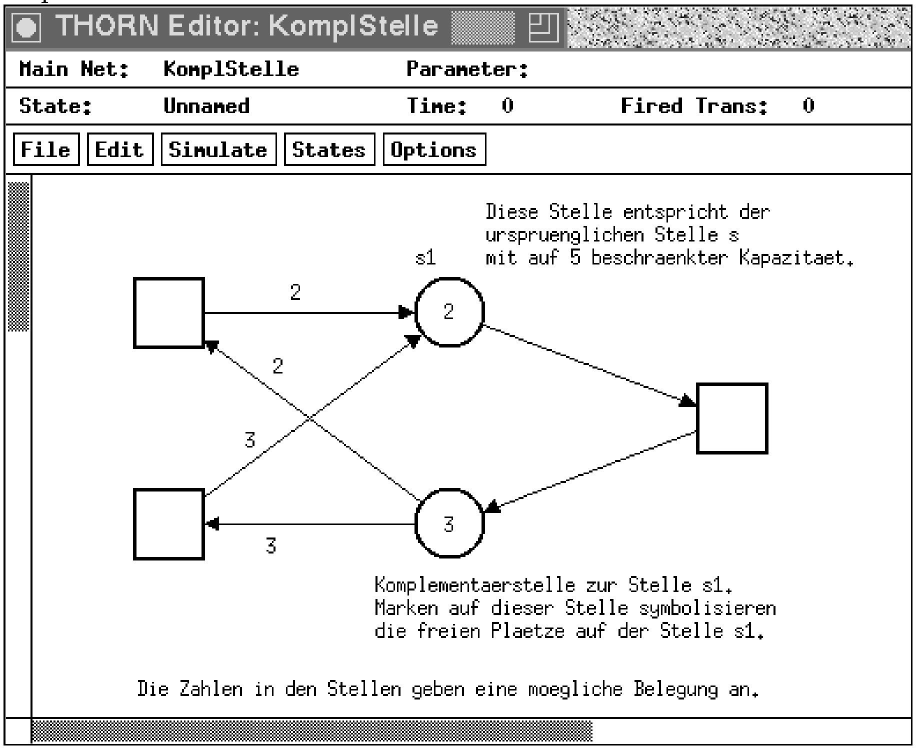



Abbildung 3.4: Komplementärbildung in Petrinetzen

Das bedeutet, daß zu jeder endlichen Berechnung eine Berechnung mit echt mächtigerer Indexmenge und gleichem Berechnungsanfang gefunden werden kann. In diesem Sinn ist also die Berechnungslänge $|D|$ unbeschränkt.

Dadurch ist die Existenz einer Berechnung mit abzählbarer Indexmenge bewiesen; hierfür kommt nach Definition der Folge (Definition 2.16) nur $D = \mathbb{N}$ in Frage. Q. e. d.

**Bemerkung 3.25:**

Die Umkehrung dieses Satzes ist falsch. Z. B. kann $T$ neben einer toten Transition weitere Transitionen enthalten, die die Berechnung fortführen.

## 3.2 Transitionssysteme

### 3.2.1 Definition

Im einfachsten Fall kann ein Transitionssystem als Modell für Zustandsübergänge definiert werden. In diesem Sinn sind auch ein endlicher, deterministischer Automat und ein Petri-Netz Transitionssysteme.

Im folgenden wird der Begriff des Transitionssystems verallgemeinert. Statt Stellen wird hier mit „Variablen" im Sinne von Programmiersprachen gearbeitet, d. h. mit Folgen im Sinne der Mathematik. Für die folgende Definition genügt es, mit den Wertebereichen dieser Folgen zu arbeiten.

**Definition 3.26:**

$(\Pi, \Sigma, \mathfrak{E}, \mathrm{T}, \Theta)$ heißt grundlegendes Transitionssystem, falls gilt:

(3.20) $\Pi = (W_1, \ldots, W_n)$, $n \in \mathbb{N}$, wobei $W_1, \ldots, W_n$ Mengen sind.

(3.21) $\Sigma \subseteqq W_1 \times W_2 \times \ldots \times W_n$

(3.22) $\mathfrak{E}$ ist eine Menge von Funktionen mit Definitions- und Wertebereichen $\Sigma$.

(3.23) $\mathrm{T} \subseteqq \{\tau \mid \tau\colon \Sigma \to \mathfrak{P}(\Sigma)\}$ mit $|T| < \aleph_0$. Jedes $\tau \in \mathrm{T}$ ist durch die Zuordnungsvorschrift

$$s \longmapsto \begin{cases} \{B_{\tau,1}(s), B_{\tau,2}(s), \ldots, B_{\tau,r_\tau}(s)\} & \text{für } \mathrm{C}_\tau \\ \varnothing & \text{sonst} \end{cases}$$

charakterisiert, wobei $\mathrm{C}_\tau$ eine aussagenlogische Formel über den Koordinaten von $\Sigma$ ist und jedes $B_{\tau,i}(s)$ die Form

$$B_{\tau,i}(s) = e_1(e_2(e_3(\ldots e_\nu(s)\ldots)))$$

mit $e_i \in \mathfrak{E}$ hat.

(3.24) $\Theta$ ist eine aussagenlogische Formel über den Koordinaten von $\Sigma$.

Eine Koordinate von $\Sigma$ heißt Variable; Bezeichnung: $\Sigma_1, \Sigma_2, \ldots, \Sigma_n$.

Ein $s \in \Sigma$ heißt Zustand.

Ein $\tau \in \mathrm{T}$ heißt Transition.

$\mathrm{C}_\tau$ heißt Aktiviertheitsbedingung der Transition $\tau$.

$\Theta$ heißt Anfangsbedingung.

**Bemerkung 3.27:**

Falls für ein $\tau$ $\mathrm{C}_\tau \equiv$ wahr ist, hat die Zuordnungsvorschrift aus (3.23) die Form

$$s \mapsto \{B_{\tau,1}(s), B_{\tau,2}(s), \ldots, B_{\tau,r_\tau}(s)\}.$$

**Definition 3.28:**

Sei $(\Pi, \Sigma, \mathfrak{E}, \mathrm{T}, \Theta)$ ein grundlegendes Transitionssystem. Sei $s \in \Sigma$ und C eine aussagenlogische Formel über den Koordinaten von $\Sigma$, den Variablen.

$s \models \mathrm{C} :\Longleftrightarrow$ C ist im Zustand $s$ erfüllt.

**Definition 3.29:**

Sei $(\Pi, \Sigma, \mathfrak{E}, \mathrm{T}, \Theta)$ ein grundlegendes Transitionssystem. Die Menge

$$\tilde{\Theta} := \{s \in \Sigma \mid s \models \Theta\}$$

heißt Menge der Anfangszustände; ein $\vartheta \in \tilde{\Theta}$ heißt Anfangszustand.

**Beispiel 3.30:**

$\Pi = (\mathbb{N}_0, \mathbb{N}_0)$, also $\Pi_1 = \mathbb{N}_0$, $\Pi_2 = \mathbb{N}_0$.

$\Sigma = \Pi_1 \times \Pi_2 = \mathbb{N}_0 \times \mathbb{N}_0$.

$$\begin{array}{lllll} \mathfrak{E} = \{ & e_1\colon & \Sigma \to \Sigma, & (a,b) \mapsto (a,b); & \\ & e_2\colon & \Sigma \to \Sigma, & (a,b) \mapsto (b,a); & \\ & e_3\colon & \Sigma \to \Sigma, & (a,b) \mapsto (a+b,a); & \\ & e_4\colon & \Sigma \to \Sigma, & (a,b) \mapsto \begin{cases} (b, a-b) & \text{für } a \geqq b \\ (0,0) & \text{sonst} \end{cases} & \}. \end{array}$$

(Der Wert $(0,0)$ spielt keine Rolle.)

$$\begin{array}{lllll} \mathrm{T} = \{ & \tau_1\colon & \Sigma \to \mathfrak{P}(\Sigma), & s \mapsto \{e_2(s), e_3(s)\}; & \\ & \tau_2\colon & \Sigma \to \mathfrak{P}(\Sigma), & s \mapsto \begin{cases} \{e_4(s)\} & \text{für } a \geqq b \\ \varnothing & \text{sonst} \end{cases} ; & \\ & \tau_3\colon & \Sigma \to \mathfrak{P}(\Sigma), & s \mapsto \{e_1(s)\} & \}. \end{array}$$

$\Theta \equiv (\Sigma_1 = 6) \wedge (\Sigma_2 = 10)$.

Dies ist ein grundlegendes Transitionssystem. Es bildet die Eigenschaften der Funktion des größten gemeinsamen Teilers zweier nichtnegativer Zahlen nach.

Mit Hilfe eines grundlegenden Transitionssystems kann ein gegebenes Petrinetz simuliert werden. Dabei ist jeder Stelle im Petrinetz eine Variable im grundlegenden Transitionssystem zugeordnet, deren Wert die Anzahl der Marken auf der korrespondierenden Stelle im Petrinetz zum Zeitpunkt $t \in \mathbb{N}$ angibt.

### 3.2.2 Ablauf

Ein grundlegendes Transitionssystem ist ein nichtdeterministisches System. Bereits die Menge der Anfangszustände kann Mächtigkeit $> 1$ haben, und evtl. kann auch in einem Zustand $s \in \Sigma$ unter mehreren Transitionen ausgewählt werden. Liegt die Transition fest, die zu einem Zeitpunkt zum Zuge kommen soll, kann evtl. noch unter mehreren Folgezuständen gewählt werden.

Interpretiert man ein grundlegendes Transitionssystem als eine Maschine, die immer, wenn eine Entscheidung notwendig wird, eine der Möglichkeiten auswählt, so bildet die Folge der Zustände, die die Maschine mit fortschreitender Zeit annimmt, einen Lauf des grundlegenden Transitionssystems.

**Definition 3.31:**

Sei $(\Pi, \Sigma, \mathfrak{E}, \mathrm{T}, \Theta)$ ein grundlegendes Transitionssystem. Eine unendliche Folge $\sigma$ in $\Sigma$ heißt ein Lauf oder eine Berechnung des grundlegenden Transitionssystems, falls gilt:

(3.25) $\sigma_1$ ist ein Anfangszustand.

(3.26) $\forall t \in \mathbb{N}: \quad [\exists \tau \in \mathrm{T}\colon\ (\tau(\sigma_t) \neq \varnothing \wedge \sigma_{t+1} \in \tau(\sigma_t))] \vee [\sigma_{t+1} = \sigma_t \wedge \forall \tau \in \mathrm{T}\colon\ \tau(\sigma_t) = \varnothing]$.

Ein Index $t \in \mathbb{N}$ in die Folge $\sigma$ heißt Zeitpunkt.

Der Übergang $(\sigma_t, \sigma_{t+1})$ heißt Transitionsschritt.

**Bemerkungen 3.32:**

1.) Für $t \in \mathbb{N}$ ist $\sigma_t$ das $n$-Tupel der Werte der Variablen zum Zeitpunkt $t$.

Der Programmiersprachenausdruck „der *neue* Wert einer Variablen“ wird durch das Fortschreiten der Zeit widergegeben; die „neuen“ Werte sind in $\sigma_{t+1}$ zu finden.

2.) Diese Definition ist so gefaßt, daß es nicht notwendig ist, daß die Transition $\forall s \in \Sigma\colon\ \tau(s) = \{s\}$ in der Menge T enthalten sein muß.

**Definition 3.33:**

Sei $(\Pi, \Sigma, \mathfrak{E}, \mathrm{T}, \Theta)$ ein grundlegendes Transitionssystem. Sei $\sigma$ ein Lauf, $\tau \in \mathrm{T}$ und $t \in \mathbb{N}$. $\tau$ heißt zum Zeitpunkt $t$ ausgewählt oder zum Zuge gekommen, falls

$$\sigma_{t+1} \in \tau(\sigma_t).$$

**Bemerkung 3.34:**

Zu einem bestimmten Zeitpunkt können mehrere Transitionen $\tau \in \mathrm{T}$ ausgewählt sein, wenn sie die geforderte Wirkung erzeugen können.

### 3.2.3 Berechnungssituationen

**Definition 3.35:**

Sei $(\Pi, \Sigma, \mathfrak{E}, \mathrm{T}, \Theta)$ ein grundlegendes Transitionssystem. Sei $s \in \Sigma$ und $\tau \in \mathrm{T}$.

(3.27) $\tau$ heißt aktiviert unter $s$, falls $\tau(s) \neq \varnothing$.
Andernfalls heißt $\tau$ blockiert unter $s$.

(3.28) $\tau$ heißt untätig unter $s$, falls $\tau(s) = \varnothing \vee \tau(s) = \{s\}$.
Andernfalls heißt $\tau$ tätig unter $s$.

**Definition 3.36:**

Sei $(\Pi, \Sigma, \mathfrak{E}, \mathrm{T}, \Theta)$ ein grundlegendes Transitionssystem. Sei $s \in \Sigma$ und $\mathrm{M} \subseteqq \mathrm{T}$.

(3.29) M heißt aktiviert unter $s$, falls $\exists\, \tau \in \mathrm{M}\colon\ \tau(s) \neq \varnothing$.
Andernfalls heißt M blockiert unter $s$.

(3.30) M heißt untätig unter $s$, falls $\forall\, \tau \in \mathrm{M}\colon\ \tau(s) = \varnothing \vee \tau(s) = \{s\}$.
Andernfalls heißt M tätig unter $s$.

**Definition 3.37:**

Sei $(\Pi, \Sigma, \mathfrak{E}, \mathrm{T}, \Theta)$ ein grundlegendes Transitionssystem und $s \in \Sigma$. $s$ heißt ein Endzustand oder terminiert, falls T untätig unter $s$ ist, d. h.

$$\forall\, \tau \in \mathrm{T}\colon\ \tau(s) = \varnothing \vee \tau(s) = \{s\}.$$

**Definition 3.38:**

Sei $(\Pi, \Sigma, \mathfrak{E}, \mathrm{T}, \Theta)$ ein grundlegendes Transitionssystem und $\sigma$ ein Lauf. $\sigma$ heißt terminierende Berechnung des grundlegenden Transitionssystems, falls $\sigma$ einen Endzustand enthält, d. h.

$$\exists\, t \in \mathbb{N}\, \forall\, \tau \in \mathrm{T}\colon\ \tau(\sigma_t) = \varnothing \vee \tau(\sigma_t) = \{\sigma_t\}.$$

**Definition 3.39:**

Sei $(\Pi, \Sigma, \mathfrak{E}, \mathrm{T}, \Theta)$ ein grundlegendes Transitionssystem. Sei $\sigma$ ein Lauf und $t \in \mathbb{N}$. Der Transitionsschritt $(\sigma_t, \sigma_{t+1})$ heißt emsig, falls $\sigma_t \neq \sigma_{t+1}$.

Andernfalls heißt der Transitionsschritt untätig.

**Satz 3.40:**

Sei $(\Pi, \Sigma, \mathfrak{E}, \mathrm{T}, \Theta)$ ein grundlegendes Transitionssystem. Sei $\sigma$ ein Lauf und $t \in \mathbb{N}$. Wenn der Transitionsschritt $(\sigma_t, \sigma_{t+1})$ emsig ist, dann existiert eine ausgewählte tätige Transition, und jede ausgewählte Transition ist tätig.

**Beweis:** Sei $t \in \mathbb{N}$ und $(\sigma_t, \sigma_{t+1})$ ein emsiger Transitionsschritt, also $\sigma_t \neq \sigma_{t+1}$.

(1) In Formel (3.26) ist für das betrachtete $t \in \mathbb{N}$ also die zweite Alternative falsch; daher muß die erste Alternative wahr sein. Das dortige $\tau \in \mathrm{T}$ ist also ausgewählt. Es folgt, daß $\tau(\sigma_t) \neq \varnothing$. Da $\sigma_{t+1} \in \tau(\sigma_t)$ und $\sigma_{t+1} \neq \sigma_t$, folgt, daß $\tau(\sigma_t) \neq \{\sigma_t\}$. Also ist $\tau$ tätig.

(2) Sei $\tau \in \mathrm{T}$ zum Zeitpunkt $t \in \mathbb{N}$ ausgewählt. Da $\sigma_{t+1} \in \tau(\sigma_t)$, ist $\tau(\sigma_t) \neq \varnothing$; und da $\sigma_{t+1} \neq \sigma_t$, folgt, daß $\tau(\sigma_t) \neq \{\sigma_t\}$. Also ist $\tau$ tätig, q. e. d.

**Bemerkung 3.41:**

Die Umkehrung dieses Satzes ist falsch. Es kann eine tätige Transition geben, die die einzige Transition ist, die ausgewählt ist, und der Transitionsschritt trotzdem untätig sein; ein emsiger Transitionsschritt kann nicht garantiert werden.

Man betrachte beispielsweise eine Transition $\tau$ und einen Zustand $s$, mit $\tau(s) = \{s, s'\}$, $s' \neq s$. $\tau$ ist eine tätige Transition. Da $s \in \tau(s)$, ist $(s, s)$ ein Transitionsschritt; er ist untätig.

**Bemerkung 3.42:**

Eine terminierende Berechnung hat nur endlich viele Zeitpunkte, an denen T im aktuellen Zustand tätig ist. Dies bedeutet, daß ab einem bestimmten Zeitpunkt nur noch untätige Transitionsschritte folgen.

**Korollar 3.43:**

Ein Lauf, der nur endlich viele emsige Transitionsschritte enthält, muß keine terminierende Berechnung sein.

**Beweis:** Es folgen zwar ab einem bestimmten Zeitpunkt nur noch untätige Transitionsschritte. Jedoch könnte ein $\tau \in \mathrm{T}$ tätig unter dem Zustand sein, an dem die Berechnung stagniert. Im vorliegenden Lauf wird dieses $\tau$ aber entweder nicht ausgewählt, oder es wird ausgewählt und hat untätige Transitionsschritte zur Folge, q. e. d.

**Korollar 3.44:**

Ein Lauf, der keine terminierende Berechnung ist, d. h. für den für alle Zeitpunkte T im aktuellen Zustand tätig ist, kann evtl. nur endlich viele emsige Transitionsschritte enthalten.

**Beweis:** Es ist nicht ausgeschlossen, daß ein untätiges $\tau \in \mathrm{T}$ existiert, das ständig zum Zuge kommt. Es ist ebenfalls möglich, daß eine tätige Transition einen untätigen Transitionsschritt zur Folge hat. Q. e. d.

**Definition 3.45:**

Sei $(\Pi, \Sigma, \mathfrak{E}, \mathrm{T}, \Theta)$ ein grundlegendes Transitionssystem und $\sigma$ ein Lauf. $\sigma$ heißt unendliche Berechnung des grundlegenden Transitionssystems, falls es unendlich viele emsige Transitionsschritte enthält, d. h.

$$|\{t \in \mathbb{N} \mid \sigma_{t+1} \neq \sigma_t\}| = \aleph_0.$$

Mit Satz 3.40 folgt dann, daß dieser Lauf auch unendlich viele Zeitpunkte hat, an denen T im aktuellen Zustand tätig ist.

## 3.3 Zweck

Transitionen in Form von Petrinetzen und grundlegenden Transitionssystemen dienen dazu, die Semantik von Systemen zu definieren. Sie sind ein Hilfsmittel, um die Semantik z. B. einer Programmiersprache formal fassen zu können.

**Beispiel 3.46:**

Eine der Variablen in einem grundlegenden Transitionssystem kann als Programmzähler dienen. Ein Sprung im Programm wird durch das Umsetzen dieser Variablen dargestellt.

Die Semantik einer Programmiersprache baut dabei induktiv auf der Semantik ihrer Konstrukte auf.

Damit steht ein Werkzeug zur Verfügung, Problemstellungen des Software-Engineerings wie z. B. die Korrektheit präzise definieren zu können.

Ebenso kann mit Hilfe von Transitionen ein Gerechtigkeitsprinzip formuliert werden.

**Forderung 3.47:**

Für jede Transition, die ständig aktiviert ist, muß es einen Zeitpunkt geben, an dem sie zur Ausführung kommt.

In einem Simulationssystem ist es wesentlich, daß alle parallel laufenden Prozesse gleichberechtigt abgearbeitet werden. Das Programmodul, das diese Fairneß erzeugt, heißt Scheduler.

Weitere Informationen zum Thema Petrinetze kann den Veröffentlichungen [Starke90] und [Reisig86] entnommen werden. Das Buch [Winskel94] enthält näheres über das Gebiet der formalen Semantik.

Das Verkehrssimulationssystem, auf das mit dieser Diplomarbeit hingearbeitet wird, wird mit einer Diskretisierung der Zeit arbeiten. Die Zustände, die sich von Zeitschritt zu Zeitschritt ändern, entsprechen den Variablen des grundlegenden Transitionssystems.

Für die Simulation wird der THORN-Petrinetzsimulator[3)] eingesetzt werden.

[3)]siehe [WIETING96] und [SCHAAL96, Kapitel 5]

# Teil II

# Verkehrsmodellierung

# Kapitel 4

# Verkehrswege

Nachdem die Vorbereitungen nun abgeschlossen sind, kann mit der Modellierung begonnen werden.

## 4.1 Problematik

Gesucht ist ein allgemeines Wegemodell, das in mathematischer Präzision folgende Probleme bewältigt:

1.) Die Verkehrswege eines beliebigen Gebietes sollen modelliert werden. Dabei soll sowohl der Individualverkehr als auch der öffentliche Verkehr auf Straßen und Schienen berücksichtigt werden.

2.) Das Modell soll leicht auf andere Verkehrsträger erweiterbar sein, z. B. auf den Schiffs- oder Flugzeugverkehr. Auch soll es möglich sein, die Beschränkungen, denen manche Fahrzeuge unterliegen (z. B. Gefahrguttransporte auf Gefällstrecken), zu berücksichtigen.

3.) Verschiedenartige Informationen (z. B. Straßenkarte – Schienennetz) sollen ohne gegenseitigen Einfluß verarbeitet werden können, damit nicht Änderungen an einem Wegesystem Änderungen an anderen Systemen nach sich ziehen.

4.) Es soll lokal modelliert werden. Z. B. darf das Einfügen einer Haltestelle nicht Änderungen an allen anderen Straßen nach sich ziehen.

5.) Das Modell muß flexibel und einfach sein, damit Änderungen (z. B. das Einfügen einer neuen S-Bahn-Linie) schnell erfolgen können.

6.) Es soll eine hierarchische Arbeitsweise ermöglicht werden; jedes beliebige Gebiet soll verfeinert werden können.

In späteren Kapiteln wird das Wegemodell dieses Kapitels zu einem Verkehrsmodell erweitert, indem Objekte hinzugefügt werden, die sich auf den hier beschriebenen Wegen bewegen.

## 4.2 Ansätze

Anstelle von nulldimensionalen Punkten werden als Aufenthaltsorte von beweglichen Objekten Gebiete, also Flächen verwendet. Dadurch ist es möglich, jedes Gebiet sukzessive zu verfeinern, indem in ihm neue Gebiete deklariert werden.

Bei einer Hierarchisierung müssen Straßen u. s. w. an Untergebiete des verfeinerten Gebietes angebunden werden. Solange die Anbindung nicht angepaßt worden ist, kann der Weg von jedem Untergebiet aus direkt benutzt werden. Mit dieser Regelung kann auch mit inkonsistenten Daten eine vernünftige Simulation gestartet werden.

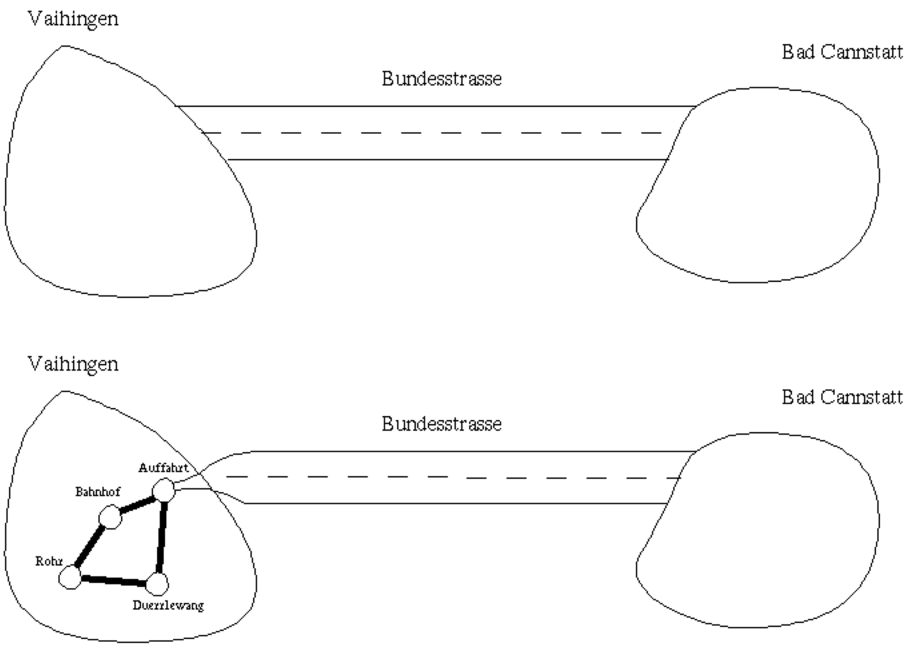



Abbildung 4.1: Anbindung von Wegstücken bei der Verfeinerung eines Gebietes

Das ganze Modell kann als Verfeinerung eines einzigen Gebietes gesehen werden. Das kann z. B. eine Stadt sein. Dennoch ist es immer möglich, weitere Städte hinzuzufügen.

Die Information für eine Verfeinerung stammt aus Karten (Plänen). Das sind Graphen, die für jedes Gebiet, das mit dem gerade verfeinerten Gebiet durch einen Weg verbunden ist, einen Knoten besitzen; hinzu kommen die Knoten für die Verfeinerung. Es sollten soviel Graphen zur Verfügung stehen, daß alle in das Verfeinerungsgebiet hineinführenden Wege an Untergebiete angeschlossen werden.

Im Beispiel der Abbildung 4.1 wurde der Graph in Abbildung 4.2 für die Verfeinerung verwendet.

Gebiete und Wege erhalten Beschriftungen, so daß sich Verkehrsmittel orientieren können.

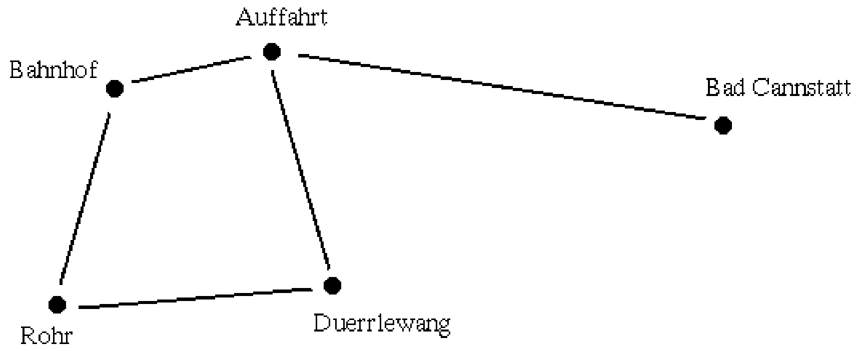



Abbildung 4.2: Verfeinerungsgraph

## 4.3 Definitionen

**Definition 4.1:**

Eine Menge $\mathfrak{G} = \{G \mid G \subseteqq \mathbb{R} \times \mathbb{R},\ G \neq \varnothing\}$ heißt Menge der Gebiete, falls gilt:

(4.1) $\forall G \in \mathfrak{G}$: $G$ ist einfach zusammenhängend[1].

(4.2) $\forall G \in \mathfrak{G}\, \exists$ Kreis $K$ (mit endlichem Radius): $G \subset K$, d. h. $G$ ist beschränkt.

(4.3) $\forall G_1, G_2 \in \mathfrak{G}, G_1 \neq G_2$: $(G_1 \cap G_2 = \varnothing) \vee (G_1 \subset G_2) \vee (G_2 \subset G_1)$

Ein $G \in \mathfrak{G}$ heißt Gebiet[2].

**Lemma 4.2:**

Sei $\mathfrak{G}$ eine Menge der Gebiete und $G \in \mathfrak{G}$. Dann ist auch $\mathfrak{G} \setminus \{G\}$ eine Menge der Gebiete.

**Beweis:** In (4.1), (4.2) und (4.3) sind weniger Elemente bzw. Paare zu überprüfen. Die übriggebliebenen Bedingungen sind von den weggefallenen unabhängig und bleiben daher gültig, q. e. d.

**Lemma 4.3:**

Sei $\mathfrak{G}$ eine Menge der Gebiete und $G \in \mathfrak{G}$. Sei $\forall G' \in \mathfrak{G}$: $G' \not\subset G$. Seien $G_1, G_2, \ldots, G_n \subset G$ einfach zusammenhängend. Dann ist auch $\mathfrak{G} \cup \{G_1, G_2, \ldots, G_n\}$ eine Menge der Gebiete.

---

[1] Eine Menge heißt einfach zusammenhängend, wenn ihr Rand zusammenhängend ist. Weitere Grundlagen können einem Topologielehrbuch wie z. B. [Rinow75] entnommen werden.

[2] Hier wird nicht gefordert, daß ein Gebiet offen sein muß; es besteht keine Gemeinsamkeit mit dem Gebietsbegiff der Topologie.

**Beweis:** Da $G \in \mathfrak{G}$, folgt nach (4.2), daß $G \subset K$ für einen geeigneten Kreis $K$. Da $G_1, G_2, \ldots, G_n \subset G$, folgt, daß $G_1, G_2, \ldots, G_n \subset K$.
Die anderen Bedingungen sind durch die Voraussetzungen direkt gegeben, q. e. d.

**Definition 4.4:**
Sei $\mathfrak{G}$ eine Menge der Gebiete und $G \in \mathfrak{G}$. Sei $\forall G' \in \mathfrak{G}$: $G' \not\subset G$. Seien $G_1, G_2, \ldots, G_n \subset G$ einfach zusammenhängend. $G$ heißt nicht verfeinert bezüglich $\mathfrak{G}$. Die Menge $\mathfrak{G} \cup \{G_1, G_2, \ldots, G_n\}$ heißt Verfeinerung des Gebietes $G$.

**Definition 4.5:**
Sei $\mathfrak{G}$ eine Menge der Gebiete. Eine Menge $\mathfrak{W} \subseteqq \{(G_1, G_2) \mid G_1, G_2 \in \mathfrak{G}\}$ heißt Menge der Wegstücke, falls gilt:

$$\forall (G_1, G_2) \in \mathfrak{W}\colon\ G_1 \cap G_2 = \varnothing.$$

Das Tupel $(\mathfrak{G}, \mathfrak{W})$ heißt ein Wegstückemodell.

**Definition 4.6:**
Sei $(\mathfrak{G}, \mathfrak{W})$ ein Wegstückemodell und $G \in \mathfrak{G}$. Die Menge

$$N_G := \{G' \in \mathfrak{G} \mid (G, G') \in \mathfrak{W} \vee (G', G) \in \mathfrak{W}\}$$

heißt Menge der Nachbarn von $G$.

**Definition 4.7:**
Sei $(V, E)$ ein Graph. Sei $(\mathfrak{G}, \mathfrak{W})$ ein Wegstückemodell. Eine Abbildung $h\colon V \to \mathfrak{G}$ heißt Einbettung des Graphen $(V, E)$ in das Wegstückemodell $(\mathfrak{G}, \mathfrak{W})$, falls gilt:

(4.4) $h$ ist injektiv.

(4.5) $\forall (v_1, v_2) \in E\colon\ (h(v_1), h(v_2)) \in \mathfrak{W}$

Schreibweise: $h\colon (V, E) \to (\mathfrak{G}, \mathfrak{W})$; $h((v_1, v_2)) := (h(v_1), h(v_2))$.

Die Elemente von $V$ und $E$ können so strukturiert sein, daß sie Informationen zur Orientierung von Verkehrsmitteln tragen, beispielsweise die Eigenschaft, eine Haltestelle oder ein Parkplatz zu sein. Diese Information kann über die Umkehrabbildung $h^{-1}\colon \mathfrak{G} \to \mathfrak{P}(V)$, $h^{-1}(G) = \{v \in V \mid h(v) = G\}$ benutzt werden.

Ebenso können Verkehrsbeschränkungen verordnet werden wie z. B. Abbiegeverbote. Solche Verordnungen können im Fall von Ampeln auch zeitabhängig sein. Damit ist eine Möglichkeit gegeben, das Verhalten einer grünen Welle zu simulieren.

Die Folge der Wegstücke, die ein Verkehrsmittel auf seiner Reise von einem Gebiet zu einem anderen mit fortschreitender Zeit betritt, bildet einen Weg.

**Definition 4.8:**
Sei $(\mathfrak{G}, \mathfrak{W})$ ein Wegstückemodell und $(G, G') \in \mathfrak{W}$. Sei $(V, E)$ ein Graph und $h$: $V \to \mathfrak{G}$. Sei $n \in \mathbb{N}$. Sei $(v_i, u_i) \in E$, so daß $(h(v_i), h(u_i)) = (G_i, G')$ mit $G_i \in \mathfrak{G}$ und $G_i \subset G$ für $i = 1, \ldots, n$. Das Wegstückemodell

$$(\mathfrak{G}, (\mathfrak{W} \setminus (G, G')) \cup (G_1, G') \cup \ldots \cup (G_n, G')) \tag{4.6}$$

heißt Anbindung des Weges $(G, G')$ an die Untergebiete $G_1, G_2, \ldots, G_n$ von $G$.

**Bemerkung 4.9:**
(4.6) ist ein Wegstückemodell, da aus $(G, G') \in \mathfrak{W}$ folgt, daß $G \cap G' = \varnothing$. Da $\forall\, i$: $G_i \subset G$, folgt, daß $\forall\, i$: $G_i \cap G' = \varnothing$. Damit ist $(\mathfrak{W} \setminus (G, G')) \cup (G_1, G') \cup \ldots \cup (G_n, G')$ eine Menge der Wegstücke und (4.6) ein Wegstückemodell.

Später werden Verkehrsmittel eingeführt, die sich im Wegstückemodell $(\mathfrak{G}, \mathfrak{W})$ bewegen. Ein $G \in \mathfrak{G}$, das nicht verfeinert ist bezüglich $\mathfrak{G}$, stellt einen einzigen, atomaren Aufenthaltsort dar. Ebenso sind Wege Aufenthaltsorte.

**Definition 4.10:**
Sei $(\mathfrak{G}, \mathfrak{W})$ ein Wegstückemodell. Dann heißt die Menge

$$\{G \in \mathfrak{G} \mid \text{G ist nicht verfeinert bezüglich } \mathfrak{G}\} \cup \mathfrak{W}$$

die Menge der Aufenthaltsorte.

**Definition 4.11:**
Sei $(\mathfrak{G}, \mathfrak{W})$ ein Wegstückemodell. Seien $(V_1, E_1), (V_2, E_2), \ldots, (V_n, E_n)$ Graphen und $\forall\, i = 1, \ldots, n$: $h_i$: $(V_i, E_i) \to (\mathfrak{G}, \mathfrak{W})$.

$$(((V_1, E_1), h_1), ((V_2, E_2), h_2), \ldots, ((V_n, E_n), h_n))$$

heißt Information des Wegstückemodells $(\mathfrak{G}, \mathfrak{W})$.

Gehen von einem Gebiet $G \in \mathfrak{G}$, das verfeinert ist, keine Wege mehr aus, d. h. $N_G = \varnothing$, kann $G$ nach Lemma 4.2 aus $\mathfrak{G}$ entfernt werden, falls $G$ keine noch gebrauchte Information trägt. Dies ist z. B. dann der Fall, wenn die Gebiete einen Namen haben, der hierarchisch nach der Schachtelung der Gebiete aufgebaut ist.

Wird ein $G$ aus $\mathfrak{G}$ entfernt, ist jedoch zu berücksichtigen, daß die Information des Wegstückemodells konsistent bleibt, d. h. weiterhin die Bedingungen (4.4) und (4.5) beachtet; nötigenfalls sind die Graphen anzupassen.

Eine Entfernung kann sinnvoll sein, wenn sich die Einbettung eines gesamten Graphen durch Verfeinerung erübrigt hat. Das entsprechende Tupel $((V_i, E_i), h_i)$ wird dann aus der Information entfernt.

## 4.4 Betrachtung

In den vorangegangenen Definitionen wurde eine allgemeine Struktur von Verkehrswegen vorgestellt. Die Definitionen sind so allgemein gehalten, daß damit beliebige Arten von Verbindungen beschrieben werden können.

**Beispiel 4.12:**
$\mathfrak{G} = \{G_1, G_2, G_3, G_4\}$
$\mathfrak{W} = \{(G_1, G_2), (G_2, G_3), (G_2, G_4)\}$
Graph $(V, E)$ mit $V = \{v_1, v_2, v_3\}$, $E = \{(v_1, v_2), (v_2, v_3)\}$
$h$: $(V, E) \to (\mathfrak{G}, \mathfrak{W})$ mit $\forall\, i = 1, 2, 3$: $h(v_i) = G_i$ ist eine Einbettung von $(V, E)$ in $(\mathfrak{G}, \mathfrak{W})$, siehe Abbildung 4.3.
$e_1 := (v_1, v_2)$ ist strukturiert und trägt die Eigenschaft „ist Straße“.
$v_2$ ist strukturiert und trägt die Eigenschaften „ist eine Haltestelle der Buslinie 80“ und „Kapazität 100“, d. h. höchstens 100 Fußgänger haben auf $G_2$ Platz.

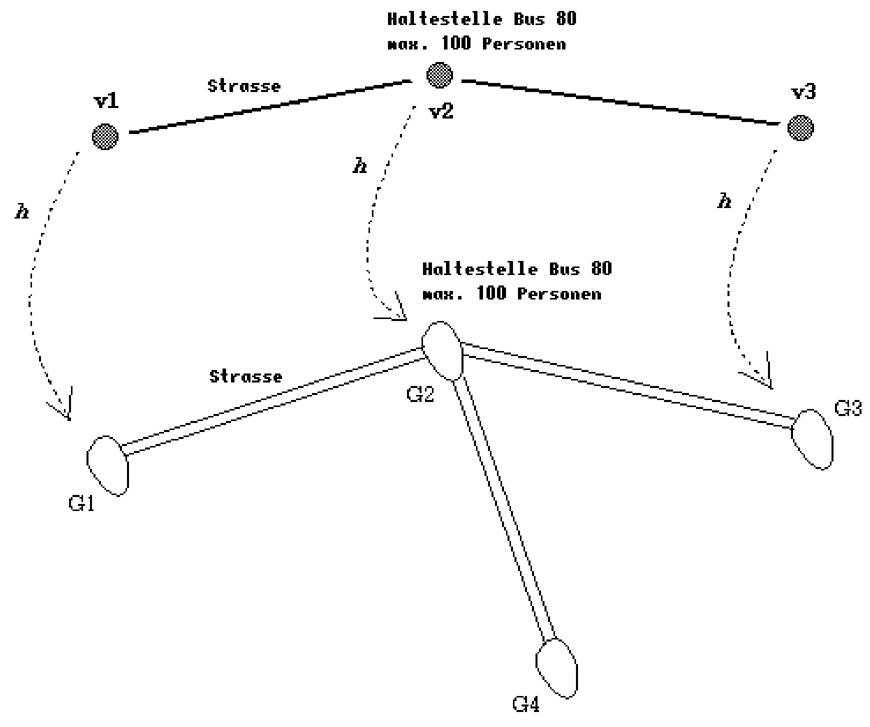



Abbildung 4.3: Beispiel einer Einbettung

An einem Wegstückemodell können nach Definition 4.11 mehrere Graphen beteiligt sein; z. B. kann jeder Graph Informationen für jeweils eine bestimmte Wegeart bereitstellen.

Fahrtrassen unterschiedlicher Verkehrsarten zwischen zwei Orten werden jeweils durch verschiedene Kanten wiedergegeben; die Menge der Gebiete muß entsprechend fein sein. Auch Sonderfahrspuren z. B. für Busse und Radfahrer werden durch eigene Kanten dargestellt.

Ein bestehendes Modell kann leicht um neue Strukturen ergänzt werden, indem neue Graphen in die Ebene eingebettet werden. Dadurch wird eine hierarchische Vorgehensweise unterstützt, da eine zunächst grobe Modellierung nach und nach verfeinert werden kann.

Ferner ist die Anzahl der verschiedenen Verkehrsarten in einem Modell unbeschränkt und kann jederzeit vergrößert werden.

# Kapitel 5

# Verkehrsmittel

Im vorherigen Kapitel wurde ein Wegstückemodell vorgestellt. Auf diesen Wegen sollen sich nun Objekte aufhalten und bewegen können. Diese Objekte werden hier als Verkehrsmittel bezeichnet. In diesem Sinne sind z. B. Fußgänger oder S-Bahnen Verkehrsmittel.

Verkehrsmittel lassen sich nach der Art ihrer Bewegungsmotivation klassifizieren. Während manche zu festen Zeiten an bestimmten Orten sein sollten (Fahrplan) und der Weg an allen Kreuzungen fest vorgegeben ist, können sich andere Verkehrsmittel freier im Wegstückemodell bewegen (Individualverkehr) und in jeder Situation ihre Vorgehensweise überdenken (dynamische Routensuche, Verkehrsmittelwahl).

Zum Einsatz in der Routensuche wird in Abschnitt 5.4 auf Bestwegalgorithmen eingegangen. In Kapitel 6 wird hieraus ein Algorithmus entwickelt, der die Bestwegsuche auch über Möglichkeiten erstreckt, andere Verkehrsmittel benutzen zu können. So wird z. B. bei der Routensuche ein Wechsel vom Individualverkehr zum öffentlichen Verkehr oder umgekehrt mit untersucht, wobei der Besitz eines eigenen PKW mit berücksichtigt wird.

Schließlich werden, um eine realistische Simulation zu erhalten, auch unvorhersehbare Ereignisse mit einbezogen (Abschnitt 5.6).

Dieses Kapitel soll noch möglichst allgemein gehalten werden. Eine der vielen Möglichkeiten, die Konzepte zu instanziieren, wird im nächsten Kapitel vorgestellt.

## 5.1 Fahrplangebundene Verkehrsmittel

Der Fahrweg von fahrplangebundenen Verkehrsmitteln wird bereits bei der Vorbereitung einer Simulation festgelegt. Während der Simulation wird dann ein derartiges Verkehrsmittel folgendermaßen gesteuert:

*1. Fall: Das Verkehrsmittel steht nicht an einer Haltestelle.* Es folgt dem ihm vorgegebenen Weg, wobei örtliche Bedingungen wie Höchstgeschwindigkeit oder Stau zu berücksichtigen sind.

*2. Fall: Das Verkehrsmittel steht an einer Haltestelle.* Unter Berückichtigung der Platzkapazität können andere Objekte (z. B. Fußgänger) ein- und aussteigen. Wenn dieser Vorgang abgeschlossen ist und die im Fahrplan vorgegebene Zeit erreicht ist, fährt das Verkehrsmittel auf dem programmierten Weg weiter.

## 5.2 Gesteuerte Verkehrsmittel

Gesteuerte Verkehrsmittel werden dadurch charakterisiert, daß sie sich im Wegemodell frei bewegen können, die Bewegungsmotivation jedoch von einem anderen Verkehrsmittel (Fahrer) kopiert wird.

**Beispiel 5.1:**

Ein Auto ist ein gesteuertes Verkehrsmittel. Es kann 5 Fußgänger (Personen) mitnehmen. Einer der mitgenommenen Personen ist als Fahrer ausgezeichnet. Er entscheidet über den Verlauf der Fahrt und ob und wo weitere Personen ein- oder aussteigen dürfen; die Kapazitätsgrenze muß dabei beachtet werden.

Die Steuerung gesteuerter Verkehrsmittel läßt sich also reduzieren auf die Steuerung der Fahrer.

**Beispiel 5.2:**

Zwei Personen machen sich gemeinsam auf die Reise. Der Wille wird ohne Beschränkung der Allgemeinheit auf eine Person konzentriert. Diese entscheidet über den Verlauf der Reise. Die andere Person kopiert deren Willen.

Dabei ist die Kapazitätsgrenze gemeinsam genutzter Verkehrsmittel zu beachten.

## 5.3 Routensuche

In diesem Abschnitt werden Heuristiken für den Entscheidungsprozeß der Steuerung entwickelt. Er betrifft Verkehrsmittel, denen ein eigener Wille modelliert werden soll. Um den Sprachgebrauch zu vereinfachen, wird hier der Entscheidungsprozeß aus der Sicht eines Autofahrers beschrieben, der mit einem Auto auf einer Straße fährt.

Es muß modelliert werden, wie sich ein Fahrer aufgrund der ihm zur Verfügung stehenden Informationen entscheiden kann, welchen Weg er zu seinem Ziel wählt.

Ein Fahrer kann – falls er möchte – auf folgende Kenntnisse zurückgreifen:

- Er kennt seinen Start- und Zielort.

- Er kennt zu jedem Zeitpunkt seinen momentanen Aufenthaltsort. Man kann interpretieren, daß er entweder mitschreibt, welche Wege er von seinem Startpunkt aus gefahren ist oder er im Besitz eines Positionsbestimmungssystems wie z. B. dem satellitengestützten Global Positioning System (GPS) ist.

- Er kennt je nach seiner Ortskundigkeit einen Teil des Wegestückemodells. Es ist auch möglich, daß er das gesamte Wegestückemodell kennt; dann kann man interpretieren, daß er im Besitz einer Landkarte ist.

– Er sieht die Belastung der von seinem Aufenthaltsort wegführenden Straßen. Dies beinhaltet auch die Sperrung einer Straße.[1)]

– An jeder Kreuzung kann spezifiziert sein, welche Straßen (Kanten) zusätzlich zu den angrenzenden sichtbar sind, d. h. auch deren Belastung sieht der Fahrer und kann entsprechend darauf reagieren. Man kann interpretieren, daß der Fahrer durch Wechselanzeigen informiert wird. Dort kann z. B. angezeigt werden, ob eine weiterführende Straße frei ist oder nicht; vgl. Abbildung 5.1.

– Es ist möglich, daß manche Fahrer zu einem bestimmten Zeitpunkt unabhängig von deren Aufenthaltsorten einige Straßenkanten einsehen können. Man kann interpretieren, daß eine Meldung im Radio-Verkehrsfunk durchgegeben worden ist. Die Fahrer, die diese Information nicht bekommen, empfangen den Verkehrsfunksender zur Zeit nicht.

– Ein Fahrer kennt den bisherigen Verlauf der Fahrt und hat noch alle Informationen, die er auf seiner Fahrt gewinnen konnte.


Abbildung 5.1: Eine Wechselanzeige des Verkehrsleitsystems der Stadt Köln

Gesucht sind Algorithmen, die angeben, wie Fahrer auf diese Informationen reagieren können. Es ist wichtig, daß jeder Fahrer lokal mit seinen Informationen für sich entscheiden kann; das bedeutet, daß es keine globale Koordination der Entscheidungen gibt. Dies modelliert den eigenen, freien Willen der Fahrer.

In diesem Abschnitt wird auf einen Bestwegalgorithmus zurückgegriffen; geeignete Bestwegalgorithmen werden in Abschnitt 5.4 vorgestellt.

---

[1)] Um diesen Abschnitt kurz halten zu können, ist, wie angekündigt, der Sprachgebrauch vereinfacht worden. Hier kann z. B. genauso der Ausfall einer S-Bahn-Linie gemeint sein.

Als Maß für die Bewertung eines Wegstückes für den Bestwegalgorithmus kann die theoretisch kürteste Fahrtzeit (bei leerer Straße) genommen werden. Dies ist günstig, weil dadurch die Höchstgeschwindigkeit und die Länge des Wegstückes berücksichtigt werden und sich die Bewertung während der Simulation nicht ändert, wodurch Vorabberechnungen ermöglicht werden.

Im folgenden werden nun verschiedene Heuristiken untersucht. Es ist zu beachten, daß der Fahrer nicht alle Informationen, die ihm zur Verfügung stehen, auch benutzen muß. Dadurch begründen sich Unterschiede in der Laufzeit und im Speicherplatzbedarf bei der Ausführung der Algorithmen auf einem Rechner.

### 5.3.1 Kein situationsbedingtes Überdenken

Der Fahrer bestimmt zu Beginn seiner Fahrt seinen Bestweg. Er hat keinerlei Informationen über den Zustand einzelner Straßenstücke. Er bestimmt also den Bestweg unter der Annahme, daß er überall freie Fahrt haben wird. Falls diese Annahme während seiner Fahrt einmal verletzt wird, wartet er.

Der Rechenaufwand ist bei diesem Verfahren am geringsten: pro Fahrt genau ein Aufruf des Bestwegalgorithmusses (bei Fahrtantritt).

Zu speichern ist während jeder Fahrt der dadurch fest programmierte Weg; weitere Informationen werden nicht verwendet und müssen daher auch nicht gespeichert werden.

Probleme mit Kreisfahrten können bei diesem Verfahren naturgemäß nicht auftreten.

Dies ist eine Heuristik, die mit geringem Rechenaufwand eine Simulation gestattet. Der Algorithmus ist einfach und daher im Vergleich zu den folgenden Vorgehensweisen in kurzer Zeit und fehlerunanfällig implementierbar.

Es ist jedoch nicht gesichert, daß es für jeden Fahrer einen Zeitpunkt gibt, an dem er sein Ziel erreicht. Da kein Fahrer seine Fahrtroute überdenkt, ist der Fall möglich, daß manche Fahrer gegenseitig aufeinander warten.

Dieses Verfahren kann als Prototyp bei der Implementation des Gesamtsystems dienen. Wenn das System dann stabil läuft, kann zu realitätsnäheren Heuristiken übergegangen werden.

### 5.3.2 Dynamische Routensuche

Das hier vorgestellte Verfahren beschreibt, daß nicht nur zu Beginn einer Fahrt Entscheidungen getroffen werden. Hier können sich durch neue Informationen, die am momentanen Aufenthaltsort zur Verfügung stehen, andere Richtungsentscheidungen ergeben.

Falls der zu Beginn der Fahrt berechnete Bestweg an einer bestimmten Stelle überlastet ist oder wenn der Fahrer durch den Radio-Verkehrsfunk oder mittels Wechselanzeigen über entfernte Stauungen informiert wird, wird von der momentanen Stelle aus mit Hilfe des Bestwegalgorithmusses ein neuer Weg zum Zielort berechnet. Dabei werden überfüllte Kanten zunächst nicht berücksichtigt. Eine Kante kann beispielsweise als überfüllt gelten, falls sie zu mehr als 90 % ausgelastet ist; es sei denn, an der Kante ist lokal ein anderer Wert für die Überlastung angegeben. Der Wert für die globale Grenzschwelle kann aus einer Benutzereingabe beim Simulationsstart stammen.

Die maximale Belastbarkeit einer Kante muß bei jeder Kante mit angegeben werden. Dieser Wert kann nicht automatisch berechnet werden, weil in diesen Wert die Breite der Straße (Anzahl der Fahrspuren) mit eingeht.

Dieser Versuch, eine neue Route zu berechnen, kann jedoch scheitern. Er scheitert, falls alle wegführenden Kanten überlastet sind. Er wird ebenso als gescheitert betrachtet, falls zwar ein neuer Bestweg berechnet wird, aber dessen Länge die Länge der alten Route um mehr als einen bestimmten Faktor $\alpha$ übersteigt. Ein günstiger Faktor $\alpha$ kann durch Simulationen ermittelt werden.

Falls also der erste Ansatz nicht zum Erfolg geführt hat, muß der Fahrer seine Strategie ändern. Er muß sich für eine Straße mit hoher Auslastung entscheiden.

Prinzipiell könnte an dieser Stelle durch sukzessives Streichen überlasteter Kanten und anschließenden Aufrufen des Bestwegalgorithmusses die Kante mit geringster Auslastung berechnet werden, die in Richtung Ziel führt. Dies bringt jedoch $n$ Aufrufe des Bestwegalgorithmusses mit sich, wobei $n$ die Anzahl der wegführenden Kanten ist.

Falls dieser hohe Rechenaufwand nicht gewünscht wird, kann sich der Fahrer in dieser Situation auch dazu entschließen, seinem ursprünglichen Bestweg zu folgen. Dies kostet keinen Rechenaufwand.

Um der Gefahr einer Verklemmung[2)] zu begegnen, erhöht jeder Fahrer nach einer gewissen Zeit den Faktor $\alpha$. Man kann interpretieren, daß, je länger er wartet, er desto mehr bereit ist, längere Umwege zum Ziel zu akzeptieren. Die Funktionsvorschrift, die die Erhöhung des Faktors $\alpha$ beschreibt, wird experimentell ermittelt werden. Dabei kann zunächst von einem linearen Anwachsen ausgegangen werden.

Dieses Verfahren führt pro Fahrer nicht mehr Daten mit als das in 5.3.1 vorgestellte.

Der Fall, daß keine der wegführenden Straßen überlastet ist, ist der Normalfall. Um zu verhindern, daß während einer Simulation häufig der Bestwegalgorithmus mit den gleichen Eingabedaten aufgerufen wird, kann der Benutzer je nach verfügbarem Speicherplatz für einige Teilgebiete die kürzesten Verbindungen unter normalen Bedingungen (keine Überlastung) bereits vorab berechnen lassen. Zusätzlich können auch einige während der Simulation berechneten Ergebnisse abgespeichert werden. Während eines Simulationslaufes muß dann vor einem Aufruf des Bestwegalgorithmusses überprüft werden, ob das Resultat schon einmal berechnet worden ist.

Die Anregung zu dieser Vorgehensweise stammt aus [SERWILL94]. Im Unterschied zu [SERWILL94, Seite 79] wird das beschriebene Vorgehen jedoch nicht iteriert, um eine Grenzverteilung der Straßenbelastung zu erhalten. Jeder Fahrer entscheidet sich für seinen Weg. Wenn sich dabei mehrere Fahrer gleichzeitig für einen leeren Weg entscheiden und dieser danach überfüllt ist, dann bleibt jedem Fahrer nur übrig, mit der neuen Situation zurecht zu kommen. Dies entspricht der realen Verkehrssituation. Die Entscheidung fällen Individuen, denen der Gesamtüberblick fehlt.

Weil ein Fahrer nach jeder Kreuzung seine Vergangenheit vergißt, besteht die Gefahr, daß er wiederholt zwischen zwei Punkten hin- und herfährt, da jeweils in einer anderen Fahrtroute eine Alternative gesehen wird und er vergessen hat, daß diese auch überfüllt ist.

---

[2)] Deadlock: gegenseitiges aufeinander warten

### 5.3.3 Fahrer mit Gedächtnis

Dieses Verhalten des Fahrers kann nur geändert werden, indem dem Fahrer ein Gedächtnis gegeben wird, mit dem er sich verschiedene Informationen aus dem bisherigen Fahrtverlauf merken und bei Entscheidungen berücksichtigen kann. Je mehr Speicherplatz pro Fahrer zur Verfügung steht, desto realitätsnähere Algorithmen können implementiert werden.

Die dynamische Routensuche aus 5.3.2 kann auf folgende Arten erweitert werden:

– Falls der Fahrer Kenntnis über seine bisherige Fahrtstrecke hat, kann modelliert werden, daß er, falls es möglich ist, höchstens $\beta$ mal über eine bestimmte Kreuzung fahren möchte. Dazu liest er vor jeder Entscheidung seinen Fahrtverlauf und nimmt beim Aufruf des Bestwegalgorithmusses jene Kanten aus, deren Zielort er schon $\beta$ mal betreten hat.

  Falls dabei keine Kanten übrigbleiben und er somit eingesperrt wäre, gibt er die Suche nach einem neuen Bestweg auf.

  Die Zahl $\beta$ wird vom Benutzer beim Simulationsstart eingegeben.

– Mit folgender Strategie kann Rechenzeit gespart werden: Es ist nicht unbedingt notwendig, an jeder Kreuzung nach einem besseren Weg zu suchen. Daher kann dem Fahrer vorgeschrieben werden, nach einem Routenwechsel dieser neuen Route mindestens $\gamma$ Kreuzungen zu folgen.

  Dies hat aber den Nachteil, daß evtl. eine günstige Alternativroute übersehen wird. Daher wird nur dann $\gamma > 1$ gewählt werden, falls nicht genügend Rechenkapazität zur Verfügung steht und deswegen die Simulation vergröbert werden soll.

– Die Realitätsnähe läßt sich wesentlich verbessern, falls sich der Fahrer jede überlastete Kante, die er auf seiner Reise gesehen hat, merken kann. Dies erfortert 1 Bit Speicherplatz pro Kante im Straßengraphen und Fahrer.

  **Beispiel 5.3:**
  Bei 100 000 Fahrern und 10 000 Kanten in Straßengraphen liegt der Speicherbedarf in der Größenordnung von 100 Megabyte bis 200 Megabyte, hier 120 Megabyte.

  Jede überfüllte Kante, die gesehen wird, wird markiert, d. h. als überfüllt gekennzeichnet. Markierte Kanten bleiben dauerhaft markiert (es sei denn, der Fahrer löscht eine Markierung) und sind von der Bestwegsuche ausgeschlossen.

  Hier kann sich der Fahrer Stauungen merken; somit werden Kreis- und Rückfahrten vermieden.

### 5.3.4 Wechsel der Verkehrsmittelart

Die bisher vorgestellten Konzepte können so angewendet werden, daß ein Wechsel der Verkehrsmittelart berücksichtigt wird. Dabei ist zu beachten:

- Ein Wechsel vom Individualverkehr zum öffentlichen Verkehr erfordert einen freien Parkplatz, falls alle Personen aus dem Verkehrsmittel aussteigen wollen.
- Ein Wechsel vom öffentlichen Verkehr zum Individualverkehr erfordert ein Verkehrsmittel. Falls das eigene Fahrzeug (siehe Abschnitt 5.5) nicht zur Verfügung steht, wird die Reise durch Taxikosten verteuert.

Diese Zusatzkosten müssen im Bestwegealgorithmus berücksichtigt werden. Eine günstige Formel zur Berechnung der Gesamtkosten kann durch Tests gewonnen werden.

### 5.3.5 Zusammenfassende Bewertung

Da sich die vorgestellten Heuristiken nicht gegenseitig ausschließen, sondern sich ergänzen, ist es möglich, bei der Realisierung des Gesamtsystems mit einfach zu implementierenden Algorithmen zu beginnen und nach erfolgreichen Laufzeittests verbessernde Varianten nachzurüsten.

Für das endgültige Simulationssystem wäre es wünschenswert, alle vorgestellten Ideen zu verwirklichen.

## 5.4 Algorithmen zur Bestwegsuche

Im vorigen Abschnitt wurden Algorithmen zur Bestwegsuche eingesetzt. Diese werden hier besprochen.

In der Literatur finden sich viele Quellen, in dene Algorithmen zur Bestwegsuche beschrieben und diskutiert werden, so z. B.

- [AHU87, Kapitel 6, insbesondere Seite 205]
- [Mack96, Kapitel 4]
- [Brandtstädt94, Kapitel 5]
- [OW90]
- [FT87] über Fibonacci-Heaps
- [DP85] über die A*-Strategie
- [DGST88] über die Parallelisierung mit Hilfe von Relaxed Heaps

Im folgenden werden verschiedene Bestwegalgorithmen vorgestellt. Bei der Beschreibung der Algorithmen wird eine programmiersprachenähnliche Notation verwendet. Für die zeitabhängigen Folgen nach Bemerkung 3.32(1) wird daher hier die Variablenschreibweise der Programmiersprachen verwendet.

### 5.4.1 Der Algorithmus von Dijkstra mit Adjazenzmatrix: $O(n^2)$

Dies ist die einfachste Form eines Algorithmusses zur Bestwegsuche.
Eingabe:

- Graph $(V, E)$
- nichtnegative Bewertung C: $E \to \mathbb{R}_0^+$
- Startknoten $\mathrm{S} \in V$
- Zielknoten $\mathrm{Z} \in V$

Ausgabe:

- Folge von Kanten: $(e_i)_{i \in D}$ in E, die den besten Weg enthält.
  Falls die Indexmenge $D \neq \varnothing$ ist, soll also in Richtung $e_1$ weitergegangen werden.
  Falls $D = \varnothing$, führt vom Startknoten kein Weg zum Zielknoten.

Lokale Variablen:

- K: $V \to \mathbb{N}$ (Kosten)
- P: $V \to V$ (Vorgängerfunktion für die Wegbestimmung)
- $\mathrm{B} \subseteqq V$ (Knoten, zu denen der kürzeste Weg bekannt ist)

Berechnungsvorschrift:

```
B := {S};
for all i ∈ V do
      K(i) := C(S, i);
      P(i) := S;
End for;
while Z ∉ B do
      Wähle w ∈ V \ B, so daß K(w) minimal ist.
      B := B ∪ {w};
      for all v ∈ V \ B do
            if K(w) + C(w, v) < K(v) then
                  K(v) := K(w) + C(w, v);
                  P(v) := w;
            End if;
      End for;
End while;
if K(Z) = ∞
      then Indexmenge D := ∅;
      else
            Indexmenge D := {1, 2, ..., k}, wobei k so bestimmt ist, daß P^{∘k}(Z) = S
            ist, wobei „∘ k“ die k malige Hintereinanderausführung von P bedeuten
            soll.
```

Die Folge für die Ausgabe ist:

$e_1 := (\mathrm{P}^{\circ k}(\mathrm{Z}), \mathrm{P}^{\circ(k-1)}(\mathrm{Z}))$

$e_2 := (\mathrm{P}^{\circ(k-1)}(\mathrm{Z}), \mathrm{P}^{\circ(k-2)}(\mathrm{Z}))$

$\vdots$

$e_{k-1} := (\mathrm{P}^{\circ 2}(\mathrm{Z}), \mathrm{P}(\mathrm{Z}))$

$e_k := (\mathrm{P}(\mathrm{Z}), \mathrm{Z})$

End if;

Wegen der zweifach geschachtelten Schleife hat dieser Algorithmus im schlimmsten Fall eine Laufzeit von $O(|V|^2)$.

### 5.4.2 Dijkstra mit Fibonacci-Heaps: $O(n \log n)$

Die Laufzeit des Algorithmusses aus 5.4.1 läßt sich noch asymptotisch verbessern. Der Effizienzgewinn kann dadurch erreicht werden, daß in der inneren Schleife nicht alle $v \in V \setminus \mathrm{B}$ behandelt werden.

Vielmehr läßt man sich vom Startknoten S aus eine äquidistante Welle ausbreiten, bis der Zielknoten erreicht ist. Die Knoten zerfallen nun in jedem Schleifendurchlauf in drei Klassen:

1. Knoten, zu denen ein kürzester Weg bereits bekannt ist; genannt gewählte Knoten.
2. Knoten, die gerade in Bearbeitung sind, sich also gerade auf der Welle befinden; genannt Randknoten.
3. Knoten, die noch unerreicht sind; genannt unerreichte Knoten.

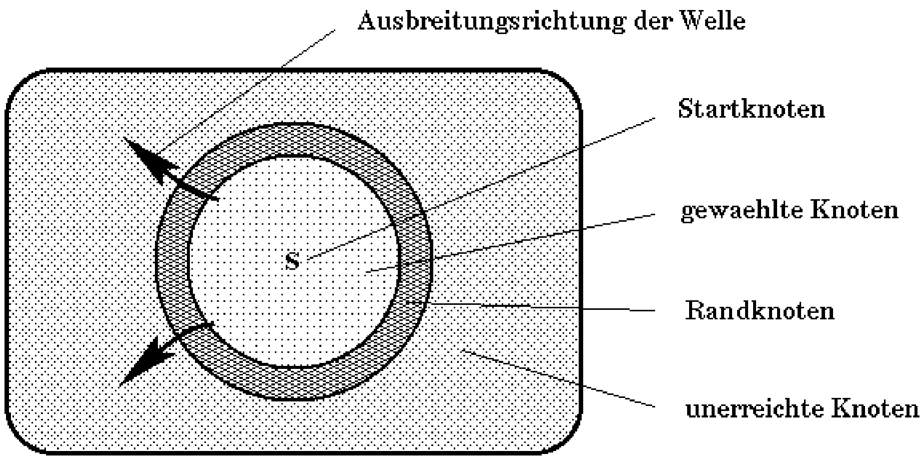



Abbildung 5.2: Die Welle äquidistanter Entfernungen

Die Steigerung der Effizienz rührt daher, daß nur auf den Randknoten gearbeitet wird und sich diese in einer Datenstruktur anordnen lassen, die einen Zugriff in $O(\log n)$ erlaubt.

Diese Datenstruktur ist der Fibonacci-Heap. Sie wird definiert in [FT87]. Die erwähnte Dijkstra-Version ist in [OW90] zu finden.

Statt mit Fibonacci-Heaps kann mit gleichem asymptotischen Aufwand auch mit normalen Heaps gearbeitet werden. Eventuelle längere Laufzeiten können durch einen erheblich einfacheren Implementierungsprozeß gerechtfertigt werden.

### 5.4.3 Heuristik bei der Bestwegsuche: $A^*$

Der bisherige Ansatz berücksichtigt nicht, wo das Ziel der Suche ist. Aus den Randknoten wird immer derjenige mit minimalem Abstand herausgesucht, auch wenn er vom Ziel wegführt. Aus [DP85] über die Optimalität des $A^*$-Verfahrens stammt die Idee, die äquidistante Welle nicht kreisförmig wachsen zu lassen, sondern durch eine Änderung der Distanzfunktion die Welle so zu deformieren, daß das Ziel schneller erreicht wird.

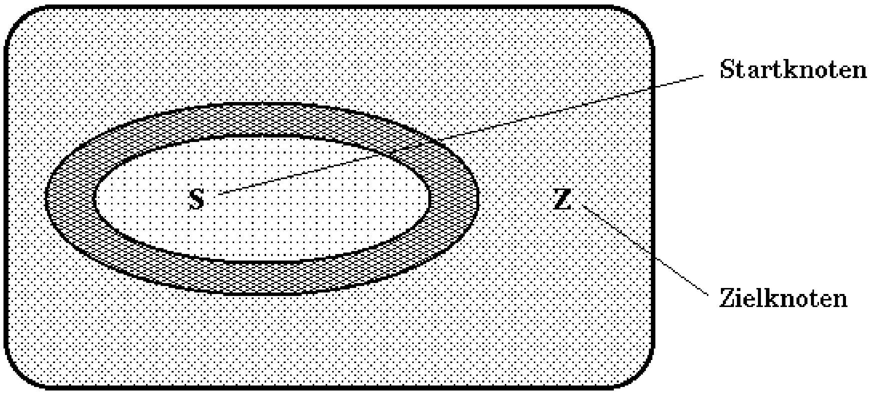



Abbildung 5.3: Die Deformation der Welle in Richtung Ziel

Als Distanzfunktion kann die Funktion $f(v) = g(v) + h(v)$, $v \in V$ gewählt werden, wobei $g$ die schon bekannten Kosten von S zu $v$ sind und $h$ eine untere Schranke der Reisekosten von $v$ zum Zielknoten ist.

Falls der Graph mit der theoretisch kürzesten Fahrtzeit bewertet ist, kann für $h$ die Flugzeit auf der Luftlinie genommen werden. Dabei sollte die Fluggeschwindigkeit höher sein als alle sonst gefahrenen Geschwindigkeiten.

Ist der Graph mit Entfernungen bewertet, kann für $h$ die euklidische Entfernung von $v$ zum Ziel genommen werden.

Diese Funktion $f$ bremst die Ausbreitung des Randes in den Richtungen, die vom Ziel wegführen. Dadurch wird die Ablaufgeschwindigkeit beschleunigt; die optimale Route wird dennoch gefunden, falls die Untere-Schranken-Bedingung an $h$ eingehalten wird.

### 5.4.4 Parallelisierung durch Relaxed Heaps

In [DGST88] wird die Datenstruktur Relaxed Heap vorgestellt. Sie ist als Ersatz zum Fibonacci-Heap im Dijkstra-Algorithmus konzipiert. Der Grundgedanke ist, gewisse Störungen in der Heapstruktur nicht sofort zu beheben, mit dem Vorteil, daß bei keiner Operation ein Worst-Case eintreten kann.

Wird die innere Schleife des Dijkstra-Algorithmusses auf einer Mehrprozessormaschine parallelisiert, so daß jeder Prozessor zu jedem Zeitpunkt die gleiche Instruktion auf seinen Daten ausführt, ist diese Eigenschaft entscheidend für den Geschwindigkeitsgewinn. Denn wenn nach einer Operation möglicherweise große Umstrukturierungen erforderlich werden, müssen die anderen Prozessoren so lange untätig warten.

## 5.5 Wahl der Verkehrsmittelart

Es gibt Verkehrsmittel, die sich auf ihrer Reise entscheiden können, andere Verkehrsmittel zu benutzen, vgl. Beispiel 5.1. Aus Gründen der sprachlichen Einfachheit werden sie hier als Personen bezeichnet.

Zusätzlich zum im vorigen Abschnitt erörterten Problem, den freien Willen zur Wahl des Weges zu modellieren, wird hier nun darauf eingegangen, wie ein freier Wille zur Benutzung oder Nichtbenutzung eines anderen Verkehrsmittels dargestellt werden kann.

Eine Person hat je nach ihrem Besitz oder ihren Gewohnheiten gewisse Präferenzen für eine bestimmte Verkehrsmittelwahl und wird versuchen, ihre Reise mit diesem Verkehrsmittel zu beginnen.

**Beispiel 5.4:**

Der Besitzer eines Autos wird sich eher für eine Fahrt mit dem Auto entscheiden.

Diese Präferenz kann bei dem Beginn einer Reise durch Würfeln festgelegt werden. Die dabei verwendete Verteilung ist so parametrisiert, daß die Gesamtheit aller Reisen von allen Personen der vorliegenden Gesamtverteilung der Verkehrsmittelwahlen entspricht.

Durch verschieden parametrisierte Simulationsläufe kann getestet werden, mit welchen Werten möglichst viele Reisen ohne Überlastungen durch das Verkehrssystem geschleust werden können.

**Beispiel 5.5:**

Angenommen, im Ganzen werden 60 % aller Fahrten mit dem Auto durchgeführt, Dann sollte sich jede Person bei jeder Reise mit einer Wahrscheinlichkeit von 0,6 dafür entscheiden, die Fahrt mit dem Auto zu beginnen.

Der wahrscheinlichkeitstheoretische Hintergrund hierfür ist das Kolmogoroffsche starke Gesetz der großen Zahlen. Das Würfeln bei jeder Person und Reise kann als Folge unabhängiger, identisch verteilter, (trivialerweise) integrierbarer, reeller Zufallsvariablen $X_n$ angesehen werden. Damit gilt

$$\frac{1}{n}\sum_{k=1}^{n} X_k \to \mathrm{E}X_1 \quad (n \to \infty) \quad \text{fast sicher.}$$

Das heißt, daß im Schnitt die gewünschte Gesamtverteilung entsteht.

## 5.6 Unfälle und Zufälle

Ein Hauptzweck eines Simulationssystems ist es, herauszufinden, wie sich der Verkehr in außergewöhnlichen Situationen verhält.

**Beispiel 5.6:**

Welche Ersatzrouten werden verstärkt belastet, wenn eine Verbindung wegen eines Unfalls oder einer Betriebsstörung ausfällt?

Für die Steuerung solcher Zufälle gibt es zwei Möglichkeiten:

- Der Anwender des Simulationssystems gibt vor oder während der Simulation an, welche Kanten nicht benutzbar sind; nach einem Ausfall auch, welche wieder benutzbar sind.
- Die unvorhersehbaren Ereignisse werden durch einen Zufallsmechanismus eingebracht. Dieser Zufallszahlengenerator soll so parametrisiert werden, daß derartige Ereignisse äußerst selten eintreten.

# Kapitel 6

# Implementation der dynamischen Routensuche

In diesem Kapitel werden die Konzepte aus den vorausgegangenen Kapiteln konkretisiert. Es wird ein Algorithmus vorgestellt, die eine der Strategien zur Routensuche testet. Darin enthalten ist die Datenstruktur zur Repräsentation der Verkehrswege und Verkehrsmittel.

Die implementierte Bestwegstrategie realisiert die dynamische Routensuche, wie sie in 5.3.2 vorgestellt worden ist.

## 6.1 Eigenschaften des Programms

Das Programm, abgedruckt in Anhang B, enthält ein fest codiertes Netz und läßt zur Demonstration fest vorgegebene Reisen ausführen. Darin enthalten ist ein Wechsel der Verkehrsträger, konkret ein Einsteigen in ein bereitgestelltes Auto.

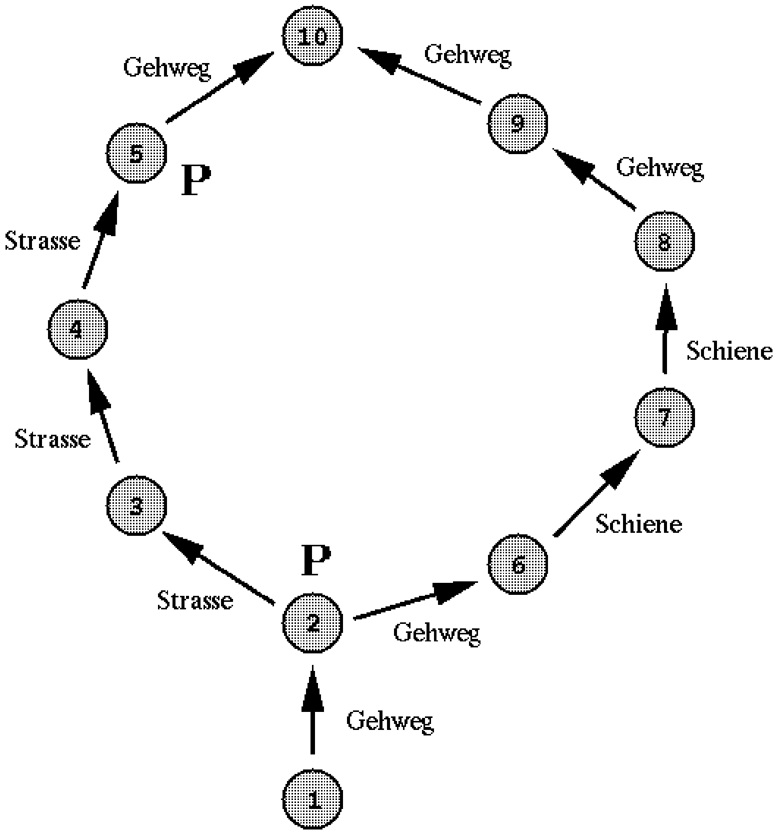
Abbildung 6.1 zeigt die codierten Verkehrswege.

Das Programm setzt eine Person Nr. 1 auf Knoten 1; das Reiseziel ist Knoten 10. Auf Knoten 2 (über den der Weg auf jeden Fall führt) steht ein Auto, das jedoch der Person Nr. 1 nicht gehört. Die Person wählt die Route über die Knoten 6, 7, 8 und 9.

Dann wird eine Person Nr. 2 auf Knoten 1 gesetzt, das Reiseziel ist wieder der Knoten 10. Die Person Nr. 2 ist Besitzer des Autos, das immer noch auf dem Knoten 2 geparkt ist. Diese Person fährt die Route über die Knoten 3, 4 und 5 mit dem Auto. Das Auto wird am Knoten 5 wieder abgestellt; die Kante (5,10) läuft die Person zu Fuß.

Danach wird die Ausgangsposition wiederhergestellt, indem das Auto wieder auf Knoten 2 gesetzt wird und die Person Nr. 2 auf den Knoten 1; abermals mit dem Reiseziel Knoten 10. Die eben beschriebene Reise wird erneut gestartet, mit dem Unterschied, daß vorher die Kante (2,3) überlastet gesetzt wird.

Die Person Nr. 2 versucht nun, ihrem Wunschweg zu folgen. Sie erkennt zwar die Ausweichroute, jedoch ist ihr der Umweg momentan zu groß, weil er sich um mehr als den Faktor $\alpha$, der bei der Person zu Beginn auf $\alpha = 1{,}1$ gesetzt wurde, unterscheidet. Da sich an der Situation, daß die Straße (2,3) überlastet ist, nichts ändert, wird die Person im Lauf der Zeit immer bereiter, längere Umwege in Kauf zu nehmen und erhöht daher den Faktor $\alpha$.




Die Zahlen in den Kreisen geben die Nummer des Knotens an.
P bedeutet, daß auf diesem Knoten ein Auto geparkt werden kann.

Abbildung 6.1: Testnetz

Schließlich entscheidet sie sich (bei $\alpha = 1{,}8$), den Umweg über die Knoten 6, 7, 8 und 9 zu nehmen.

Das Protokoll dieser Reisen ist nach dem Programmcode im Anhang B abgedruckt.

## 6.2 Datenstruktur

Im Programm[1)] haben die Knoten, Kanten und Verkehrsmittel folgende Strukturelemente:

– Knoten

- Nummer: Nummer des Knotens.
- Kapazität: Anzahl der Verkehrsmittel, die sich auf diesem Knoten aufhalten können.

[1)]in `Netz.def`

  - Auslastung: Anzahl der Verkehrsmittel, die sich momentan auf diesem Knoten aufhalten.
  - Parkplatz: Anzahl der Parkplätze, die momentan auf diesem Knoten frei sind.
  - Parkplatzgebühren: Parkplatzgebühren pro Verkehrsmittel, das auf diesem Knoten parkt.

- Kante
  - Typ: Art der Kante, z. B. Straße, Schiene oder Gehweg.
  - Kapazität: Anzahl der Verkehrsmittel, die sich auf dieser Kante aufhalten können.
  - Auslastung: Anzahl der Verkehrsmittel, die sich momentan auf dieser Kante aufhalten.
  - Anfangspunkt: weist auf den Knoten, mit dem die Kante anfängt.
  - Endpunkt: weist auf den Knoten, mit dem die Kante aufhört.
  - Länge.
  - Höchstgeschwindigkeit.
  - Bewertung: Graphbewertung. Wird hier gesetzt zu Länge / Höchstgeschwindigkeit.

- Verkehrsmittel
  - Typ: Art des Verkehrsmittels, z. B. Person, Auto oder Zug.
  - Mitnahmekapazität: Anzahl der Personen, die das Verkehrsmittel befördern kann.
  - Auslastung: Anzahl der Personen, die momentan von diesem Verkehrsmittel befördert werden.
  - Wunschweg: verkettete Liste von Kanten, die den Weg enthält, den das Verkehrsmittel momentan verfolgt.
  - maximale Geschwindigkeit: Höchstgeschwindigkeit des Verkehrsmittels.
  - Fortbewegungsmittel.
  - Aufenthaltsort (ein Knoten oder eine Kante).
  - Entfernung bis Kantenende.
  - Typ gerade gefahrene Kante: enthält die Art der gerade verlassenen Kante, z. B. Straße, Schiene oder Gehweg.
  - Besitz: Verkehrsmittel, z. B. ein bestimmtes Auto.
  - Zielort.
  - bisherige Reisekosten.
  - alpha: Faktor, um wieviel der Kandidat für den neuen Weg höchstens länger sein darf als der bisher favorisierte Weg, um akzeptiert zu werden.

## 6.3 Heuristik zur Routensuche

Im Programm wird die in 5.3.2 vorgestellte Variante zur Routensuche, die dynamische Routensuche, implementiert. Die Erweiterung auf einen Fahrer mit Gedächtnis (in 5.3.3) bleibt der zukünftigen Arbeit überlassen.

Die letzte der durchgeführten Testreisen (siehe Abschnitt 6.1) zeigt, wie sich eine Person dynamisch für eine neue Route entscheidet.

Der Algorithmus trägt folgende Grundzüge:[2)]

– Eine neue Reise wird wie folgt erzeugt:

> Wunschweg $(e_i)_{i \in D}$ :=Bestweg(Start,Ziel);
> if $D = \varnothing$ then
>> „Es gibt keinen Weg vom Startort zum Zielort; die Fahrt kann nicht stattfinden.“
>
> End if;

– Ein Verkehrsmittel, das durch dynamische Routensuche gesteuert wird, führt einen Zeitschritt der Reise durch:

> if das Verkehrsmittel hält sich auf einer Kante auf then
>> if Entfernung bis zum Kantenende $> 0$ then
>>> Bewege dich in Abhängigkeit von einem evtl. benutzten Fortbewegungsmittel, der eigenen und der am Weg vorgegebenen Höchstgeschwindigkeit und der relativen Auslastung des Weges ein gewisses Stück auf der Kante weiter.
>>
>> End if;
>> if Ende der Kante erreicht then
>>> Aufenthaltsort := Endpunkt der gerade durchfahrenen Kante.
>>> Den nächsten Knoten (Nahziel) aus dem Wunschweg holen.
>>
>> End if;
>
> End if;
> if das Verkehrsmittel hält sich auf einem Knoten auf then
>> if Zielknoten erreicht
>>> then Fahrt löschen.
>>> else
>>>> if Überlast(die Kante, die zum nächsten Knoten führt) then
>>>>> Durch einen Aufruf des Bestwegalgorithmusses eine Alternative suchen.
>>>>> if (Es gibt eine Alternative) and (Sie unterscheidet sich in den Reisekosten um nicht mehr als um den individuellen Faktor $\alpha$ von der bisherigen Route)
>>>>>> then Akzeptiere die Alternativroute.

[2)]`PROCEDURE Reisebeginn` und `PROCEDURE Zeitschritt` im Modul Fahrer else

Verwerfe die Alternativroute und verfolge weiterhin den bisher favorisierten Weg.

Erhöhe den Faktor $\alpha$.

End if;

End if;

Falls ein Wechsel des Verkehrssystems notwendig wird, führe diesen durch.

Aufenthaltsort := die Kante, die zum nächsten Knoten führt.

End if;

End if;

## 6.4 Bestwegealgorithmus

Der implementierte Algorithmus[3)] zur Bestwegsuche basiert auf dem Algorithmus von Dijkstra mit Adjazenzmatrix-Speicherverwaltung, wie er in Abschnitt 5.4.1 vorgestellt worden ist. Im fertigen Simulationssystem sollte eine Variante mit geringerer asymptotischer Laufzeit zum Einsatz kommen.

Folgende Modifikationen gegenüber dem Algorithmus aus 5.4.1 sind dabei vorgenommen worden:

- Eingabe:
  - Graph $(V, E)$
  - nichtnegative Bewertung C: $E \to \mathbb{R}_0^+$
  - Startknoten S $\in V$
  - Zielknoten Z $\in V$
  - Überlastfunktion: Kante $\to$ $\{\text{wahr}, \text{falsch}\}$.

    Dadurch wird die Kenntnis des Fahrers von der Überlastung einer Kante übergeben.

    Die übergebene Funktion muß den Wahrheitswert wahr zurückgeben, falls der Fahrer von einer Überlastung der Kante weiß. Es muß der Wahrheitswert falsch zurückgegeben werden, falls der Fahrer weiß, daß die Kante nicht überlastet ist oder er keine Information über die Überlastung der Kante hat.
  - Kantentyp, auf der der Fahrer zum Startknoten S gelangt ist; zu Beginn einer Reise kann „Gehweg“ angegeben werden.
- Ausgabe:
  - Folge von Kanten: $(e_i)_{i \in D}$ in E, die den besten Weg enthält.

    Falls $D = \varnothing$, führt vom Startknoten kein Weg zum Zielknoten.

---

[3)]in Modul Bestwegsuche

- Vor jedem Aufruf der Funktion, die zu einer Kante deren Bewertung aufruft, wird mit Hilfe der übergebenen Überlastfunktion überprüft, ob die Kante als überlastet gilt. In diesem Fall wird die Bewertung zu unendlich gesetzt, andernfalls wird die Bewertung des Graphen genommen.
- Die Kosten für eine Kante $(i, j)$ sind: Wechselkosten(Kantentyp, auf dem zum Knoten $i$ gefahren wurde,Typ$(i, j)$) + $C(i, j)$.

Das aufgerufene Unterprogramm hat die folgende Spezifikation; die Implementierung ist im Modul Bestwegsuche zu finden, abgedruckt im Anhang B.

```
PROCEDURE Wechselkosten (
  wer:        VerkehrsmittelZeiger;
  wo:         KnotenZeiger;
  von, nach: KanteArt
): REAL;

(*
  Kosten, die fuer das Verkehrsmittel "wer" bei einem Wechsel von der
  Knotenart "von" zu der Knotenart "nach" an der Stelle "wo" entstehen.
*)
```

Zusätzlich zum Algorithmus von Dijkstra vollzieht dieser Algorithmus eine Transformation der Graphbewertung, indem die Wechselkosten mit einbezogen werden.

Bei der Verifikation dieses Algorithmusses kann man sich auf der Verifikation des Algorithmusses von Dijkstra in der Literatur stützen; die Bewertung ist nach der Transformation nach wie vor positiv.

# Kapitel 7

# Ausblick

Diese Diplomarbeit hat ein Modell vorgestellt, wie der Individualverkehr in eine Verkehrssimulation einbezogen werden kann, indem die Verkehrsteilnehmer einzeln nach dem Prinzip der dynamischen Routensuche gesteuert werden.

In der zukünftigen Arbeit wird in Richtung eines marktfähigen Produktes weitergearbeitet werden. Dazu sind noch verschiedene Bausteine notwendig:

- Eine graphische Benutzeroberfläche zur Eingabe von Daten und zur Überwachung der Simulation muß realisiert werden.
- Aus einer benutzerfreundlichen Eingabe muß automatisch die Graphstruktur erzeugt werden.
- Die Fortbewegung von Verkehrsmitteln, die nicht dynamisch gesteuert werden, also insbesondere öffentliche Verkehrsmittel, muß implementiert werden. Das Einsteigen in solche Verkehrsmittel läßt sich dann genauso realisieren wie der Einsteigevorgang in ein Auto; allerdings ist danach die Bewegung der Person vom öffentlichen Verkehrsmittel zu kopieren.
- Ampeln inklusive einer grünen Welle können eingearbeitet werden.
- Die dynamische Routensuche sollte um die Konzepte aus 5.3.3 erweitert werden, so daß sich die Fahrer Informationen aus dem bisherigen Fahrtverlauf merken und verwenden können. Dies ist wünschenswert, da mit dieser Strategie Kreisfahrten vermieden werden.

  Hier kommt eine Überlastfunktion zum Einsatz, die die dem Fahrer bekannte Information aus der Datenbank holt und zurückgibt.
- Der Bestwegealgorithmus sollte optimiert werden, um eine geringere asymptotische Laufzeit erreichen zu können.
- Es ist ein Scheduler zu implementieren, der die zur Verfügung stehende Rechenzeit fair auf die einzelnen Fahrer verteilt.
- Es werden Laufzeittests zur Bestimmung und Optimierung der verwendeten Konstanten stattfinden.

- Um Simulationsergebnisse zu erhalten, ist es erforderlich, Daten aus einer Verkehrserhebung eingeben zu können, damit das System realitätsnah Verkehr erzeugen und durch den Verkehrsgraphen schleusen kann.

Damit wird ein Verkehrssimulationssystem entstehen, das beliebige Verkehrsstrukturen anwenderfreundlich nachbildet.

# Anhang A

# Ein verworfenes Verkehrsmodell

Dieses Verkehrsmodell wurde aus später genannten Gründen nicht weiterverfolgt. Im Anschluß an die Definition folgen Erläuterungen und die Gründe für die Ablehnung.

## A.1 Definition

Wegen der Vielzahl an auftretenden Variablen werden im folgenden zum Teil Bezeichner an Stelle von Buchstaben verwendet. Für die Projektion auf eine Koordinate wird dann die Punktschreibweise eingesetzt; vergleichbar zu Records in Pascal oder Ada.

**Definition A.1:**

Eine endliche, nichtleere Menge heißt eine Menge der Zeitpunkte.

**Beispiel A.2:**

Die Menge $\{0, 1, 2, \ldots, 23\} \times \{0, 1, 2, \ldots, 59\}$ ist eine Menge der Zeitpunkte.

**Definition A.3:**

Eine Menge $\mathfrak{A}$ mit $|\mathfrak{A}| = 6$ heißt Menge der Verkehrsträger. Die Elemente von $\mathfrak{A}$ heißen Fußgänger, Auto, Bus, Bahn, Straße, Schiene.

**Definition A.4:**

Sei „Zeit" eine Menge der Zeitpunkte. Sei $\mathfrak{A}$ eine Menge der Verkehrsträger. Ein Tupel $(\mathfrak{F}, \mathfrak{B})$, wobei $\mathfrak{F} = \{F_1, F_2, F_3, \ldots\}$ und $\mathfrak{B} = \{B_1, B_2, B_3, \ldots\}$, heißt ein Verkehrssystem, falls gilt:

(1) $F_i$ ($i \in \mathbb{N}$) heißt ein festes Verkehrsobjekt und hat folgende Eigenschaften: $F_i$ = (Typ, darf_weiterleiten, Ort, Parkplatz, Bushaltestelle, Anschlußstücke, Bahnhof, Wegweiser, Kapazität, Auslastung), wobei

(a) Typ $\in$ {Straße, Schiene}
(b) darf_weiterleiten $\subseteqq$ {Fußgänger, Auto, Bus, Bahn}
(c) Ort $\in \mathbb{R} \times \mathbb{R}$
(d) Parkplatz $\in$ {wahr, falsch}
(e) Bushaltestelle $\in$ {wahr, falsch}
(f) Anschlußstücke $\subseteqq \mathfrak{F}$
(g) Bahnhof $\subseteqq \mathfrak{F}$
(h) Wegweiser $\subseteqq \mathfrak{F} \times$ Anschlußstücke
(i) Kapazität: darf_weiterleiten $\to \mathbb{N}_0$
(j) Auslastung: Zeit $\times$ darf_weiterleiten $\to \mathbb{N}_0$
(k) $\forall\, i \in \mathbb{N}$: $[F_i.\text{Typ} = \text{Straße}$
$\Longrightarrow F_i.\text{darf\_weiterleiten} = \{\text{Fußgänger, Auto, Bus}\}]$
$\forall\, i \in \mathbb{N}$: $[F_i.\text{Typ} = \text{Schiene}$
$\Longrightarrow F_i.\text{darf\_weiterleiten} = \{\text{Bahn}\}]$
(l) $\forall\, i \in \mathbb{N}$: $[F_i.\text{Typ} = \text{Schiene}$
$\Longrightarrow F_i.\text{Bushaltestelle} = \text{falsch}]$
(m) $\forall\, i \in \mathbb{N}\, \forall\, F \in F_i.\text{Anschlußstücke}$:
$(F.\text{Typ} = F_i.\text{Typ} \wedge F_i \in F.\text{Anschlußstücke})$
(n) $\forall\, i \in \mathbb{N}\, \forall\, H \in F_i.\text{Bahnhof}$:
$(H.\text{Typ} \neq F_i.\text{Typ} \wedge F_i \in H.\text{Bahnhof})$
(o) $\forall\, t \in$ Zeit: $F_i.\text{Auslastung}(t, \cdot) \leqq F_i.\text{Kapazität}$

(2) $B_i$ ($i \in \mathbb{N}$) heißt ein bewegliches Verkehrsobjekt und hat folgende Eigenschaften: $B_i$ = (Typ, darf_sich_bewegen_auf, Fußgängerkapazität, Auslastung, Aufenthaltsort, will_mitnehmen, will_aussteigen_lassen, Richtung, Verzögerung, Steuerung), wobei

(a) Typ $\in$ {Fußgänger, Auto, Bus, Bahn}
(b) darf_sich_bewegen_auf $\in$ {Auto, Bus, Bahn, Straße, Schiene}
(c) Fußgängerkapazität $\in \mathbb{N}_0$
(d) Auslastung: Zeit $\to \mathbb{N}_0$
(e) Aufenthaltsort: Zeit $\to \mathfrak{F}$
(f) will_mitnehmen: $\mathbb{N}_0 \times \mathfrak{F} \times \mathfrak{B} \to$ {wahr, falsch}
1. Koordinate: Fußgängerkapazität
2. Koordinate: Aufenthaltsort
3. Koordinate: Objekt, das einsteigen will
(g) will_aussteigen_lassen: Auslastung $\times\ \mathfrak{F} \to$ {wahr, falsch}
2. Koordinate: Aufenthaltsort
(h) Richtung: Zeit $\to \mathfrak{F}$
(i) Verzögerung $\in \mathbb{N}$
(j) Steuerung: $\mathfrak{F} \times \mathfrak{F} \times$ Zeit $\to$ {Auto, Bus, Bahn, Straße, Schiene} $\times\ \mathfrak{F}$
1. Koordinate Definitionsbereich: Aufenthaltsort

2. Koordinate Definitionsbereich: Richtung
2. Koordinate Wertebereich: neue Richtung

(k) $\forall\, i \in \mathbb{N}$: $[B_i.\text{Typ} = \text{Fußgänger}$
$\Longrightarrow B_i.\text{darf\_sich\_bewegen\_auf} = \{\text{Auto, Bus, Bahn, Straße}\}]$
$\forall\, i \in \mathbb{N}$: $[B_i.\text{Typ} \neq \text{Fußgänger}$
$\Longrightarrow B_i.\text{darf\_sich\_bewegen\_auf} = \{\text{Straße}\}]$

(l) $\forall\, i \in \mathbb{N}$: $[B_i.\text{Typ} = \text{Fußgänger}$
$\Longrightarrow B_i.\text{Fußgängerkapazität} = 0]$
$\forall\, i \in \mathbb{N}$: $[B_i.\text{Typ} = \text{Auto}$
$\Longrightarrow B_i.\text{Fußgängerkapazität} = 5]$
$\forall\, i \in \mathbb{N}$: $[B_i.\text{Typ} = \text{Bus}$
$\Longrightarrow B_i.\text{Fußgängerkapazität} = 70]$
$\forall\, i \in \mathbb{N}$: $[B_i.\text{Typ} = \text{Bahn}$
$\Longrightarrow B_i.\text{Fußgängerkapazität} = 1000]$

(m) Sei $B \in \mathfrak{B}$ mit $B.\text{Typ} \neq \text{Fußgänger}$.
$\forall\, i \in \mathbb{N}$: $B_i.\text{will\_mitnehmen}(\,\cdot\,,\,\cdot\,, B) \equiv \text{falsch}$.

(n) $\forall\, i \in \mathbb{N}$: $B_i.\text{will\_mitnehmen}(B_i.\text{Fußgängerkapazität},\,\cdot\,,\,\cdot\,) \equiv \text{falsch}$

(o) $\forall\, i \in \mathbb{N}$: $B_i.\text{will\_aussteigen\_lassen}(0,\,\cdot\,) \equiv \text{falsch}$

(p) $\forall\, t \in \text{Zeit}$: $B_i.\text{Auslastung}(t) \leqq B_i.\text{Fußgängerkapazität}$

(3) Sei $F \in \mathfrak{F}$.
$\forall\, i \in \mathbb{N}$: $[B_i.\text{Typ} = \text{Auto}$
$\Longrightarrow (F.\text{Parkplatz} \Leftrightarrow B_i.\text{will\_aussteigen\_lassen}(1, F))]$

(4) Sei $F \in \mathfrak{F}$ und $B \in \mathfrak{B}$ mit $B.\text{Typ} = \text{Fußgänger}$.
$\forall\, i \in \mathbb{N}\, \forall\, k < B_i.\text{Fußgängerkapazität}$: $[B_i.\text{Typ} = \text{Bus}$
$\Longrightarrow (F.\text{Bushaltestelle} \Leftrightarrow B_i.\text{will\_mitnehmen}(k, F, B))]$

(5) Sei $F \in \mathfrak{F}$ und $k > 0$.
$\forall\, i \in \mathbb{N}$: $[B_i.\text{Typ} = \text{Bus}$
$\Longrightarrow (F.\text{Bushaltestelle} \Leftrightarrow B_i.\text{will\_aussteigen\_lassen}(k, F))]$

(6) Sei $F \in \mathfrak{F}$ und $B \in \mathfrak{B}$ mit $B.\text{Typ} = \text{Fußgänger}$.
$\forall\, i \in \mathbb{N}\, \forall\, k < B_i.\text{Fußgängerkapazität}$: $[B_i.\text{Typ} = \text{Bahn}$
$\Longrightarrow (F.\text{Bahnhof} \neq \varnothing \Leftrightarrow B_i.\text{will\_mitnehmen}(k, F, B))]$

(7) Sei $F \in \mathfrak{F}$ und $k > 0$.
$\forall\, i \in \mathbb{N}$: $[B_i.\text{Typ} = \text{Bahn}$
$\Longrightarrow (F.\text{Bahnhof} \neq \varnothing \Leftrightarrow B_i.\text{will\_aussteigen\_lassen}(k, F))]$

**Bemerkungen A.5:**

1.) Das Verkehrssystem muß nicht zusammenhängend sein. D. h., es wird nicht verlangt, daß ein bestimmtes Straßenstück von jedem anderen aus erreichbar sein muß.

2.) Die Ortskoordinaten bei festen Verkehrsobjekten sind nur für die Graphik von Bedeutung. Sie sollten so gewählt werden, daß miteinender verbundene Streckenstücke auch in der Graphik aneinander angrenzend liegen.

3.) Nach der Definition ist es möglich, daß auch ein Schienenstück ein Parkplatz ist. Dies soll bedeuten, daß dieser Ort eine Endhaltestelle sein kann.

## A.2 Erläuterungen

In diesem Modell ist es nicht erforderlich, daß im Schienenverkehr die Fahrtrichtungen getrennt modelliert werden. Es genügt, eine Folge von Schienenstücken, die von einem Bahnhof zum nächsten wie eine Kette miteinander verbunden sind. Die Kapazität pro Schienenstück kann 1 betragen, wenn ein Vertauschen der Aufenthaltsorte zweier sich gegenüberstehenden Bahnen zu einem Zeitpunkt erlaubt wird.

Die Kapazität eines Straßenstücks kann je nach gewünschter Eigenschaft der Straße gesetzt werden.

Jedes einzelne bewegte Verkehrsobjekt hat eine Mindestaufenthaltszeit auf einem Straßen- bzw. Schienenstück („Verzögerung"). Dies entspricht der Geschwindigkeit und der Länge des Straßenstücks. Sofern die Kapazität der angrenzenden Straßenstücke es erlaubt, können dann zu einem Zeitpunkt mehrere bereite bewegte Verkehrsobjekte auf ein angrenzendes Straßen- bzw. Schienenstück hüpfen.

Für ein festes Verkehrsobjekt (Straße oder Schiene) gibt die Menge der Anschlußstücke an, mit welchen anderen festen Verkehrsobjekten des gleichen Typs es verbunden ist. Die Menge Bahnhof gibt dagegen an, mit welchen festen Verkehrsobjekten des jeweils anderen Typs eine direkte Verbindung besteht.

Parkplatz und Bushaltestelle sind Eigenschaften eines Straßenstücks. Bewegliche Verkehrsobjekte können diese erkennen und ihr Verhalten danach ausrichten. Hinzu kommt, daß nur an einem Parkplatz alle Personen aus einem Auto aussteigen dürfen.

Über die Wegweiser kann sich ein bewegliches Verkehrsobjekt orientieren, welches der möglichen Anschlußstücke zu seinem gewünschten Ziel führt.

Funktionen, die von der Zeit abhängen, deuten die Eigenschaft einer „Variablen" an, vgl. Bemerkung 3.32(1).

Die Funktion Steuerung in einem beweglichen Verkehrsobjekt kann vom gesamten Verkehrssystem $(\mathfrak{F}, \mathfrak{B})$ abhängen. Das bedeutet, daß in die Steuerung des Individualverkehrs Informationen über die Überlastung einer Strecke oder den Ausfall einer Verbindung mit einfließen können.

## A.3 Kritik

Ein Nachteil dieses Modells ist, daß eine Änderung in einem Teilsystem (z. B. Schienensystem) nicht auf dieses eine System beschränkt bleibt, sondern Anpassungen im ganzen Verkehrssystem nach sich zieht.

**Beispiel A.6:**

Beim Einrichten eines neuen Bahnhofs wird beim anliegenden Straßenstück ein Rückwärtsverweis auf das Schienenstück erforderlich.

Eine Hierarchisierung ist durch eine Kapazitätserhöhung von Aufenthaltsorten möglich. Aber auch hier muß bei einer Verfeinerung in umliegende Objekte eingegriffen werden, z. B. um Anschlußstücke neu zu setzen.

Ferner ist es unmöglich, ohne strukturelle Änderung an der Definition und damit an allen bereits definierten Objekten andere Verkehrsträger wie Flugzeuge oder Schiffe nachträglich hinzuzumodellieren.

# Anhang B

# Test der Algorithmen (Listings)

Das nachstehende Programm testet die Algorithmen zur Bestwegsuche und zur dynamischen Routensuche. Anhand fest vorgegebener Daten wird der Verlauf einer Reise demonstriert.

Das Programm ist in MODULA-2 geschrieben; seine Dokumentation ist in Kapitel 6 zu finden. Ohne Änderung des Codes liefert der Aufruf des Programms stets das gleiche Ergebnis. Dieses ist nach dem Programmcode abgedruckt.

Die Module hängen wie in Abbildung B.1 angegeben voneinander ab.

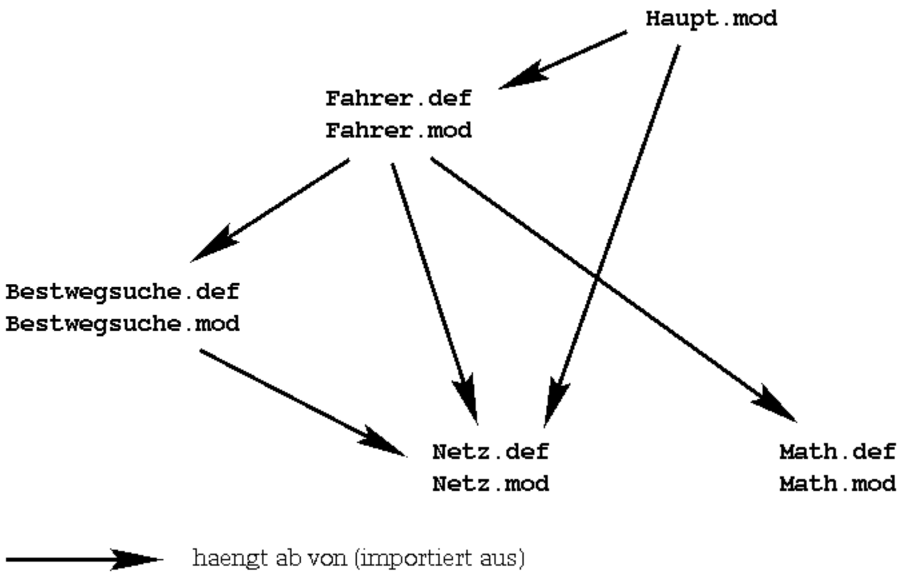



Abbildung B.1: Abhängigkeitsgraph der Module

Die Dateien sind in dieser Reihenfolge abgedruckt:

`Bestwegsuche.def`, `Bestwegsuche.mod`, `Fahrer.def`, `Fahrer.mod`, `Haupt.mod`, `Netz.def`, `Netz.mod`, `Math.def`, `Math.mod`, Ausgabe des Programms.

# Quellenverzeichnis


[AHU87] Alfred Aho, John E. Hopcroft und Jeffrey D. Ullman. *Data Structures and Algorithms.* Verlag Addison-Wesley, 1987.

[BAUER78] Heinz Bauer. *Wahrscheinlichkeitstheorie und Grundzüge der Maßtheorie.* 3. Auflage. Verlag Walter de Gruyter Berlin, New York, 1978.

[BO93] Ronald V. Book und Friedrich Otto. *String-Rewriting Systems.* Springer-Verlag, 1993.

[BRANDTSTÄDT94] Andreas Brandtstädt. *Graphen und Algorithmen.* Verlag B. G. Teubner Stuttgart, 1994.

[CLAUS95] Volker Claus. *Vorlesung Netze und Prozesse.* Universität Stuttgart, Institut für Informatik, 1995.

[CR] *Computing Reviews,* Volume 38, Number 1, 1997.

[DGST88] James R. Driscoll, Harold N. Gabow, Ruth Shrairman und Robert E. Tarjan. Relaxed Heaps: An Alternative to Fibonacci Heaps with Applications to Parallel Computation. *Communications of the ACM,* Volume 31, Number 11, Seiten 1343 – 1354, 1988.

[DIEKERT90] Volker Diekert. *Combinatorics on Traces.* Springer-Verlag, 1990.

[DIEKERT94] Volker Diekert. *Vorlesung Diskrete Mathematik.* Universität Stuttgart, Institut für Informatik, 1994.

[DP85] Rina Dechter und Judea Pearl. Generalized Best-First Search Strategies and the Optimality of A$^*$. *Journal of the Association for Computing Machinery (ACM),* Volume 32, Number 3, Seiten 505 – 536, 1985.

[FT87] Michael L. Fredman und Robert Endre Tarjan. Fibonacci Heaps and Their Uses in Improved Network Optimization Algorithms. *Journal of the Association for Computing Machinery (ACM),* Volume 34, Number 3, Seiten 596 – 615, 1987.

[MACK96] Marcus Mack. *Untersuchung von effizienten Algorithmen zur Bestimmung der k-kürzesten Wege innerhalb von ÖPNV-Verkehrsnetzen.* Diplomarbeit Nr. 1374. Universität Stuttgart, Institut für Informatik, 1996.

[MT87] Alistair Moffat und Tadao Takaoka. An all pairs shortest path algorithm with expected time $O(n^2 \log n)$. *SIAM Journal on Computing,* Volume 16, Number 6, Seiten 1023 – 1031, 1987.

[OW90] Thomas Ottmann und Peter Widmayer. *Algorithmen und Datenstrukturen.* B. I. Wissenschaftsverlag Mannheim, Wien, Zürich, 1990.

[REISIG86] Wolfgang Reisig. *Petrinetze – Eine Einführung.* 2. Auflage. Springer-Verlag, 1986.

[RICHTER97] Herr Richter. Brief vom 7. April 1997. Stadt Köln, Amt für Straßen und Verkehrstechnik.
Amt für Straßen und Verkehrstechnik, Hollweghstraße 22 – 26, 51103 Köln. Telefon: (02 21) 2 21-78 33. Zeichen: 663/2.

[RINOW75] W. Rinow. *Lehrbuch der Topologie.* Deutscher Verlag der Wissenschaften Berlin, 1975.

[SCHAAL96] Markus Schaal. *Verkehrsmodellierung.* Diplomarbeit Nr. 1373. Universität Stuttgart, Institut für Informatik, 1996.

[SCHÖLLER97] Wolfgang Schöller. Persönliches Gespräch am 11. März 1997. Universität Stuttgart, Institut für Navigation.

[SERWILL94] Dirk Serwill. *DRUM – Modellkonzept zur dynamischen Routensuche und Umlegung.* Institut für Stadtbauwesen, RWTH Aachen, 1994.
RWTH Aachen, Mies-van-der-Rohe-Straße 1, 52056 Aachen.

[STARKE90] Peter H. Starke. *Analyse von Petri-Netz-Modellen.* Verlag B. G. Teubner Stuttgart, 1990.

[UNKHOFF97] Herr Dr. Unkhoff. Telefonat am 2. Mai 1997. Stadt Stuttgart, Tiefbauamt.
Landeshauptstadt Stuttgart, Amt 66, Postfach 10 60 34, 70049 Stuttgart. Telefon: (07 11) 2 16-27 92. Zeichen: 66-3-20.

[WEIGAND79] Michael M. Weigand. *Algorithmen zur Bestimmung k-kürzester Wege in einem Graphen.* Dissertation. Universität Stuttgart, Institut für Informatik, 1979.

[WIETING96] Ralf Wieting. *Handbuch zur THORN-Entwicklungsumgebung.* OFFIS, 1996.
OFFIS, Escherweg 2, 26121 Oldenburg.
`http://www.offis.uni-oldenburg.de/projekte/DNS/`

[WINSKEL94] Glynn Winskel. *The Formal Semantics of Programming Languages – An Introduction.* The MIT Press, 1994.

# Register

**Erklärung**

Hiermit versichere ich, diese Arbeit
selbständig verfaßt und nur die
angegebenen Quellen benutzt zu haben.

---

(Axel Schwarz)