Handbuch

zu

VERENA

Eine Programmbibliothek zum Rechnen mit Wahrscheinlichkeitsverteilungen

Ralf Nitzgen
Berthold Bärk
Werner Sandmann

Dritte Version, Mai 2000

Zur Urheberschaft dieses Handbuchs:

Das vorliegende Werk hat eine geistige Mutter und mehrere solcher Väter. Wesentliche Teile des Handbuchs einschließlich Abbildungen und Programmbeispielen konnten aus der Diplomarbeit von Dietmar Siepmann übernommen und modifiziert werden. Die Passagen, die sich auf die Behandlung diskret unendlicher Zufallsvariablen beziehen, sind das Verdienst von Brigitte Neuenfels. Das Vorwort orientiert sich zum Teil an einer Arbeit von J. C. Strelen. Zahlreiche Erweiterungs- und Verbesserungswünsche gegenüber den Kindertagen von VERENA wurden von Ralf Nitzgen und Brigitte Neuenfels mit großer Sorgfalt immer prompt erfüllt. Außerdem hat Ralf Nitzgen das Handbuch maßgeblich gestaltet und mit bewundernswerter Geduld stets neue Nachbesserungswünsche und andere lästige Mäkeleien meinerseits ebenso klaglos ertragen und umgesetzt wie die von Mario Heckler, der sich durch besonders pedantisches Korrekturlesen Lorbeeren erworben hat.

Berthold Bärk

Vorwort

Bei der Modellierung von Rechensystemen ist es praktisch und elegant, wenn die *Lösung* eines Modells, also quantitative Aussagen über die seine Eigenschaften beschreibenden Größen, mittels geschlossener mathematischer Ausdrücke angegeben werden kann. In diesem Fall liegt ein analytisches Modell vor. Jedoch gelingt es nicht immer, auf diese Weise ein adäquates Modell zu erstellen, so daß der Modellierer nach einer anderen Möglichkeit der Modellanalyse suchen muß. Im Bereich stochastischer Modelle sind dies häufig simulative Verfahren, zu deren Nachteilen jedoch der große Rechenaufwand gehört. Heute werden auch numerische Methoden, bei denen komplizierte mathematische Ausdrücke numerisch ausgewertet werden, in zunehmendem Maße betrachtet und erfolgreich angewandt. Dies zeigt sich u. a. in einer stark wachsenden Zahl von Publikationen zu diesem Thema.

In numerischen Modellen können wesentlich mehr Aspekte erfaßt und untersucht werden als in analytischen Modellen, erwähnt seien hier nur Modelle mit komplizierten Bedienstrategien, Blockierungen von Kunden oder endlichen Speichern für Nachrichten oder Aufträge. Vergleichbar ist dies mit anderen Gebieten der Mathematik, wo beispielsweise ein Ingenieur bei der Anwendung mathematischer Verfahren die Auswertung numerisch durchführt – man denke etwa an Differentialgleichungen.

Bei der Modellierung von Rechensystemen und auch über diesen Bereich hinaus werden häufig Markov-Modelle benutzt. Für Markovketten ist der Zustandsraum derartiger Modelle oft so groß, daß weder alle Zustandswahrscheinlichkeiten noch die Übergangswahrscheinlichkeiten berechnet und gespeichert werden können, letztgenannte selbst dann nicht, wenn die Übergangsmatrizen dünn besetzt sind. Allerdings interessieren meist auch nicht alle Zustandswahrscheinlichkeiten, sondern nur die Wahrscheinlichkeiten bestimmter Zustände oder gewisse Randwahrscheinlichkeiten, etwa für den Zustand einer ausgewählten Komponente eines untersuchten Rechensystems. Diese Wahrscheinlichkeiten von *Aggregaten*, also Mengen von Zuständen, werden auch *Makrowahrscheinlichkeiten* genannt.

In unserer Arbeitsgruppe untersuchen wir nun Verfahren, die nur diese Aggregate und Makrowahrscheinlichkeiten betrachten und die versuchen, die Modelle nur hinsichtlich dieser zu analysieren, ohne dabei alle Wahrscheinlichkeiten für den Zustand des gesamten Modells zu berechnen. Da das Modell auf diese Weise nicht ganz genau erfaßt wird, kommt es zu Verfahrensfehlern, deren Auswirkungen auf die Genauigkeit der erzielten Resultate zu untersuchen ist. In diesem Zusammenhang ist es häufig erforderlich, Analyseprogramme für verschieden feine Aggregationen zu erstellen, um die Güte der Ergebnisse unterschiedlicher Ansätze zu vergleichen.

Es hat sich als sehr zweckmäßig erwiesen, die Zustände der zu modellierenden Komponenten durch Zufallsvariablen zu beschreiben und die Analysealgorithmen mit diesen Zufallsvariablen und Operationen auf ihnen zu formulieren. Eine solche Operation ist beispielsweise die Faltung der Verteilungen zweier zu addierender Zufallsvariablen, falls diese unabhängig voneinander sind. Wir bezeichnen diese Sprechweise bei der Formulierung von Analysealgorithmen als *Stochastische Modelliersprache* (SMS). Eine formale verbindliche Sprachdefinition von SMS liegt bisher nicht vor, ist jedoch geplant. Derweil sei auf zahlreiche Beispiele in den im Literaturverzeichnis aufgeführten Arbeiten von Strelen bzw. Strelen/Bärk und den ebenfalls dort genannten Diplomarbeiten verwiesen.

Bei der Programmierung von Analysealgorithmen zeigt sich, daß gewisse Operationen auf Verteilungen von Zufallsvariablen häufig wiederkehren und beträchtlicher Programmieraufwand für die Erzeugung und Initialisierung von Verteilungen von Zufallsvariablen entsteht. Auf Anregung von J. C. Strelen wurde mit VERENA ein Werkzeug geschaffen, das für diese und andere Aufgaben zahlreiche Hilfsmittel in Form von Datentypen und Operationen auf Objekten dieser Typen bereitstellt. Angestrebt wird bei mit VERENA erstellten Analyseprogrammen eine möglichst genaue Entsprechung mit den in SMS formulierten Analysealgorithmen. So wird eine sehr gute Lesbarkeit der Analyseprogramme erzielt, und die Anzahl der Fehlerquellen, die üblicherweise ein langes und damit potentiell unübersichtliches Analyseprogramm enthält, wird erheblich reduziert. Zudem ist es so möglich, sehr schnell neue Analyseprogramme zu erstellen.

Die erste Version von VERENA wurde wesentlich von Dietmar Siepmann im Rahmen seiner Diplomarbeit mitgestaltet und implementiert. Erweiterungen bis zum derzeitigen Stand von VERENA haben Brigitte Neuenfels mit ihrer Diplomarbeit und Ralf Nitzgen vorgenommen. Als Einführung in die Vorgehensweise unserer Arbeitsgruppe und damit auch die Grundlagen für VERENA sei die Arbeit von Strelen über iterative Analyse von Markov-Modellen mit alternierender Aggregation und Disaggregation empfohlen.

VERENA hat unserer Arbeitsgruppe bisher die Arbeit schon erheblich erleichtert, und wir stellen dieses Werkzeug anderen Interessenten gerne zur Verfügung.

Bonn, im Januar 1993

Berthold Bärk

Vorwort zur dritten Version

Die aktuelle Version von VERENA enthält als zusätzliche Erweiterungen Prozeduren zur Berechnung von Minima und Maxima stetiger Zufallsvariablen. An einigen Stellen wurden zudem kleinere Modifikationen bereits bestehender Prozeduren vorgenommen. Diese Erweiterungen entstanden bereits im Rahmen meiner Diplomarbeit, wurden jedoch erst jetzt in die VERENA-Bibliothek integriert. Die nun vorliegende dritte Version des Handbuchs ist daher inhaltlich um die Beschreibung dieser Erweiterungen ergänzt worden.

Weiterhin wurde der Latex–Code an die Version Latex2_ϵ angepaßt und damit auch das Layout geringfügig geändert. Besonderer Dank hierfür gebührt Eric Hambuch, der die vorhandenen Codes entsprechend modifiziert hat.

Bonn, im Mai 2000

Werner Sandmann

Inhaltsverzeichnis

1 Einleitung

VERENA (VErteilungsREchensystem zur Numerischen Analyse) ist eine Programmbibliothek zur numerischen Analyse von Rechensystemen (z.B. Pollingsystemen, GG1-Modelle etc.). Mit Hilfe dieser Bibliothek können Verteilungen von systembeschreibenden Größen und Leistungsmaße wie Erwartungswerte und höhere Momente berechnet werden. Für den Modellierer besteht die numerische Analyse eines konkreten Rechensystems aus fünf Phasen:

1. Bildung eines Modells des realen Systems, wobei nur die für die gesuchten Leistungsmaë wesentlichen Eigenschaften des Systems Berücksichtigung finden;
2. Beschreibung des Modells mit Hilfe von Zufallsvariablen, die die Zustände des Modell-Systems und die Abläufe in diesem System strukturell charakterisieren;
3. Formulierung des Analyse-Algorithmus unter Verwendung von Operationen auf Zufallsvariablen;
4. Implementierung des Algorithmus auf einer Rechenanlage in Form eines Programms;
5. Ermittlung der gesuchten Leistungsmaße aus den mit Hilfe des Programms berechneten Zustands- und anderen Wahrscheinlichkeiten (z.B Blockierwahrscheinlichkeiten).

Der größte Teil des Zeitaufwandes der vierten Phase entfällt auf die Implementierung und Verifikation der notwendigen Operationen auf Zufallsvariablen. Trotz der starken strukturellen Unterschiede zwischen den einzelnen Varianten sind die zu implementierenden Operationen im wesentlichen gleich. Am häufigsten kommen Faltungsoperationen ($X := Y \oplus Z$), Minimumoperationen ($X := \min(Y, k)$), Maximumoperationen ($X := \max(Y, k)$) und die Anwendung des Satzes der totalen Wahrscheinlichkeit vor.

Die Programmbibliothek soll den Modellierer bei der Bewältigung der Aufgaben der vierten Phase unterstützen. Dabei wäre die Entwicklung eines Compilers, der den Analyse-Algorithmus als Eingabe akzeptiert und in ein Maschinenprogramm übersetzt, offensichtlich der Idealfall, da die gesamte Phase 4 somit entfallen würde. Überlegungen bezüglich des Aufwandes führen jedoch zu der Lösung, eine vorhandene Programmiersprache durch die Bereitstellung von Bibliotheken mit allen benötigten Datentypen und Operationen so zu erweitern, daß die Formulierung des Algorithmus in dieser Programmiersprache mit weniger Aufwand und geringen Programmierkenntnissen möglich ist. Eine spätere Erweiterung des Systems durch einen Compiler, der diese Aufgabe selbständig ausführt, wird dadurch nicht ausgeschlossen.

Aufgrund der weiten Verbreitung auf verschiedenen Rechensystemen und des Modulkonzepts wurde als Programmiersprache MODULA-2 gewählt. Sie bietet bereits alle fundamentalen sprachlichen Mittel zur Implementierung eines Analyse-Algorithmus. Die noch fehlenden Mittel werden weitgehend von der Programmbibliothek bereitgestellt.

Die Realisierung von Bibliotheken geschieht mittels Definitions- und Implementationsmodulen. In den Definitionsmodulen sind alle nach außen sichtbaren Eigenschaften der bereitgestellten (exportierten) Konstanten, Datentypen, Variablen und Prozeduren festgelegt. Die Realisierung dieser Objekte, also im wesentlichen die Anweisungsteile von Prozeduren, befinden sich in den Implementationsmodulen. Eine Einführung in die Programmierung in MODULA-2 wird beispielsweise in [Wi84] und in [Dal84] gegeben.

Im Gegensatz zu der Programmiersprache 'Pascal' existiert in MODULA-2 nur der Typ 'Procedure' und nicht der Typ 'Function'. Sollte im weiteren Text von Funktionen die Rede sein, so sind damit MODULA-2-Prozeduren gemeint, die einen Rückgabeparameter besitzen und somit Pascal-Funktionen entsprechen. Durch die Verwendung der Begriffe 'Prozedur' oder 'Funktion' ist damit sofort klar, wie das entsprechende Unterprogramm zu benutzen ist (Zuweisung oder Aufruf).

In den beiden folgenden Kapiteln werden einige Konventionen eingeführt, die zugrundeliegenden Verfahren erläutert und die Approximation von Verteilungen dargestellt. Im vierten Kapitel wird die Repräsentation von Zufallsvariablen in VERENA beschrieben. Kapitel fünf enthält die Beschreibung der Bibliotheksmodule und ihrer Komponenten. Im Anschluß daran wird die Benutzung einzelner Prozeduren anhand von Programmbeispielen vorgeführt. Am Ende des Handbuchs finden sich ein Index und eine Beschreibung der derzeitigen Installation von VERENA am Institut für Informatik der Universität Bonn.

2 Zugrundeliegende Verfahren und Konventionen

Bei der Entwicklung der Programmbibliothek fanden unter anderem zwei mathematische Verfahren Berücksichtigung – die Analyse von Markovketten mittels Vektoriteration (Power Method) und die Approximation stetiger bzw. diskret unendlicher Verteilung und durch 101-Dichten bzw. 101-Verteilungen. Sie sind in [Si91] und [Ne93] beschrieben.

Zufallsvariablen werden im folgenden grundsätzlich mit großen, Werte von Zufallsvariablen mit kleinen lateinischen Buchstaben bezeichnet. Werden mehrere Zufallsvariablen in Form eines Tupels zusammengefaßt, so wird der dann verwendete Bezeichner fett gedruckt. Die Wertmenge einer Zufallsvariablen X wird immer mit Ω_X benannt. Sind die Wertmengen $\Omega_{X^{(t)}}$ einer Folge $X^{(t)}, X^{(t+1)}, \ldots, t \in I\!N$, identisch, so werden sie einheitlich mit Ω_X bezeichnet.

Sind X und Y beliebige Zufallsvariablen mit den Wertmengen Ω_X und Ω_Y, so bezeichnet $(X|Y = y)$, $y \in \Omega_Y$, angelehnt an die Definition von bedingten Wahrscheinlichkeiten, eine Zufallsvariable mit der Wertmenge $\Omega_{(X|Y)} \subseteq \Omega_X$, deren Verteilung durch

$$P\{(X|Y = y) = x\} := P\{X = x|Y = y\}$$

für alle $x \in \Omega_X$ und alle $y \in \Omega_Y$ mit $P\{Y = y\} > 0$ definiert ist.

3 Approximation von Verteilungen

Bevor auf Datenstrukturen und Implementierungsdetails eingegangen wird, muß geklärt werden, wie Verteilungen im Rechner dargestellt werden sollen. Dies geschieht im folgenden für stetige Zufallsvariablen und für solche mit diskret unendlichem Wertebereich.

3.1 Approximation stetiger Verteilungen

Die Benutzung der Programmbibliothek für die numerische Analyse von Warteschlangensystemen, zu deren Beschreibung stetige Zufallsvariablen erforderlich sind, setzt nicht nur voraus, daß ihre Verteilungen im Rechner darstellbar sind, sondern auch, daß alle benötigten Operationen wie Faltung, Einschränkung des Ergebnisraums oder Berechnung von Momenten durch Unterprogramme realisiert werden können. Die Approximation von Verteilungen stetiger Zufallsvariablen mittels 101-Dichten, wie sie in [St88] für die Analyse von GG1-Systemen verwendet wird, erfüllt diese Voraussetzungen.

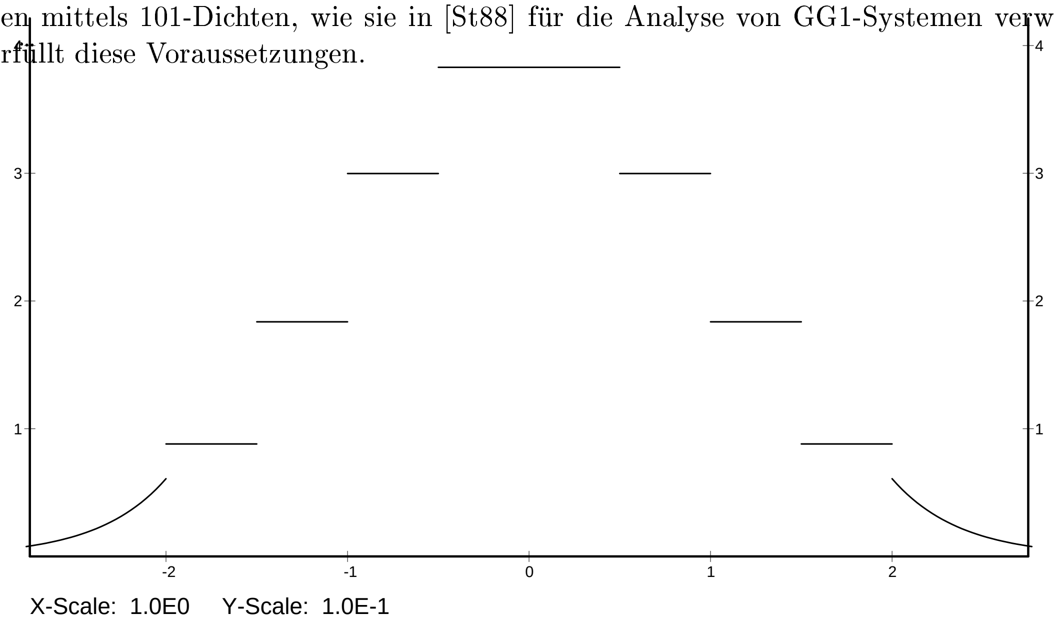

Abbildung 1: Approximation durch 101-Dichte mit 10 Intervallen

Der Definitionsbereich der betrachteten stetigen Dichtefunktionen ist $\mathbb{R}$. Die Approximation durch eine 101-Dichte beruht darauf, diesen Definitionsbereich in endlich viele Intervalle zu unterteilen und die Dichtefunktion über jedem der Intervalle durch Dichten maximaler Entropie (DME) der Ordnung 0 bzw. der Ordnung 1 anzunähern. Die Ordnung einer DME gibt an, wieviele Momente der anzunähernden Dichte für ihre Bestimmung herangezogen werden.

Die Verteilung einer Zufallsvariablen $X : \Omega \to \mathbb{R}$ sei gegeben durch die Dichtefunktion

$$f : \mathbb{R} \to \mathbb{R}_0^+.$$

Es werden die $r + 2$ Teilmengen

$$(-\infty, a \cdot \delta)\ ,\ [a \cdot \delta, (a+1) \cdot \delta)\ ,\ [(a+1) \cdot \delta, (a+2) \cdot \delta)\ ,\ \ldots\ ,\ [b \cdot \delta, (b+1) \cdot \delta)\ ,\ [(b+1) \cdot \delta, +\infty)$$

mit $r = b - a + 1$ gebildet. Die Parameter a, b und δ sind gemäß den Erfordernissen der anzunähernden Verteilung und der gewünschten Genauigkeit zu wählen.

Für die Intervalle $(-\infty, a \cdot \delta)$ und $[(b+1) \cdot \delta, +\infty)$ werden die Dichten maximaler Entropie der Ordnung 1 (DME-1) bestimmt. Die DME-1 über einem halbseitig unendlichen Intervall ist immer durch eine Exponentialfunktion gegeben. Deren Norm („nulltes Moment“) und erstes Moment über diesem Intervall sind mit den Momenten der angenäherten Dichte über dem gleichen Intervall identisch:

$$f_{links}(x) := c' e^{\lambda' x}$$

mit

$$\int_{-\infty}^{a\cdot\delta} f_{links}(x)dx = \int_{-\infty}^{a\cdot\delta} f(x)dx =: W_{links},$$

$$\int_{-\infty}^{a\cdot\delta} xf_{links}(x)dx = \int_{-\infty}^{a\cdot\delta} xf(x)dx =: M_{links},$$

sowie

$$f_{rechts}(x) := ce^{-\lambda x}$$

mit

$$\int_{(b+1)\cdot\delta}^{+\infty} f_{rechts}(x)dx = \int_{(b+1)\cdot\delta}^{+\infty} f(x)dx =: W_{rechts},$$

$$\int_{(b+1)\cdot\delta}^{+\infty} xf_{rechts}(x)dx = \int_{(b+1)\cdot\delta}^{+\infty} xf(x)dx =: M_{rechts}.$$

Für die Parameter c, c', λ und λ' gilt:

$$\lambda = \frac{1}{\frac{M_{rechts}}{W_{rechts}} - (b+1)\cdot\delta}$$

$$c = \frac{\lambda W_{rechts}}{e^{-\lambda((b+1)\cdot\delta)}}$$

$$\lambda' = -\frac{1}{\frac{M_{links}}{W_{links}} - a\cdot\delta}$$

$$c' = \frac{\lambda' W_{links}}{e^{\lambda'(a\cdot\delta)}}$$

Die Dichtefunktion der anzunähernden Verteilung wird über den r Intervallen $[a\cdot\delta, (a+1)\cdot\delta)$ bis $[b\cdot\delta, (b+1)\cdot\delta)$ mit r Dichten maximaler Entropie der Ordnung 0 (DME-0) approximiert. Die DME-0 über einem Intervall $[i\cdot\delta, (i+1)\cdot\delta)$ ist gegeben durch eine konstante Funktion:

$$f_i(x) := f_i,$$

mit

$$\int_{i\cdot\delta}^{(i+1)\cdot\delta} f_i(x)dx = \int_{i\cdot\delta}^{(i+1)\cdot\delta} f(x)dx =: W_i.$$

Dabei gilt:

$$f_i = \frac{W_i}{\delta}.$$

Die Abbildung 1 zeigt eine Approximation einer Standardnormalverteilung. Als Parameter wurden $\delta = 0.5$, $a = -4$ und $b = 3$ gewählt.

3.2 Approximation diskret unendlicher Verteilungen

Bei der Beschreibung der numerischen Analyse von Warteschlangensystemen werden neben den stetigen auch diskrete Zufallsvariablen benötigt. Diskrete Zufallsvariablen mit endlichem Wertebereich können dargestellt werden, indem jedem Punkt aus dem Wertebereich einer solchen Zufallsvariable eine Wahrscheinlichkeit zugeordnet und in einer Feldkomponente im Rechner gespeichert wird. Diese Vorgehensweise verbietet sich bei Zufallsvariablen mit diskret unendlichem Wertebereich. Es muß also eine Beschreibung von Verteilungen diskreter Zufallsvariablen gefunden werden, die im Rechner darstellbar ist und die Realisierung aller benötigten Operationen auf Verteilungen von Zufallsvariablen als Unterprogramme ermöglicht. Analog zur Approximation von Verteilungen stetiger Zufallsvariablen wurde für Zufallsvariablen mit diskret unendlichem Wertebereich die in [St88] beschriebene Approximation von Verteilungen diskreter Zufallsvariablen mittels 101-Verteilungen gewählt. Im folgenden wird die Approximation diskret unendlicher Zufallsvariablen kurz beschrieben. Ausführlicher wird dies in [Ne93] behandelt.

Der Definitionsbereich einer betrachteten Wahrscheinlichkeitsfunktion sei $\mathbb{Z}$ oder ein endliches Intervall aus $\mathbb{Z}$. Bei der Approximation durch eine 101-Verteilung wird der Definitionsbereich in endlich viele Intervalle gemäß $(-\infty, a) \cap \mathbb{Z}$, $[a, a+1) \cap \mathbb{Z}$, $[a+1, a+2) \cap \mathbb{Z}$, $\ldots$, $[b-1, b) \cap \mathbb{Z}$, $[b, b+1) \cap \mathbb{Z}$, $[b+1, \infty) \cap \mathbb{Z}$ zerlegt. In einem Bereich der Form $[a_i, a_{i+1}) \cap \mathbb{Z}$ liegt genau ein Punkt aus dem Wertebereich der Zufallsvariablen. Deshalb werden solche Bereiche öfter als „Einpunktintervalle“ bezeichnet.

Die durch die Wahrscheinlichkeitsfunktion gegebene Verteilung wird über den Intervallen, die mehr als einen Punkt enthalten, mittels einer Verteilung maximaler Entropie (VME) der Ordnung 1 approximiert. Über allen anderen Intervallen wird sie durch eine Verteilung maximaler Entropie der Ordnung 0 angenähert.

Bei der Approximation einer Wahrscheinlichkeitsfunktion durch eine VME n-ter Ordnung werden alle Momente der Wahrscheinlichkeitsfunktion von der Norm (dem „nullten Moment“) bis einschließlich des n-ten Moments exakt angenommen. Die höheren Momente der zu approximierenden Wahrscheinlichkeitsfunktion werden nicht berücksichtigt.

Sei also die Verteilung einer diskreten Zufallsvariablen gegeben durch die Wahrscheinlichkeitsfunktion $p : \mathbb{Z} \longrightarrow \mathbb{R}_0^+$. Der Definitionsbereich von p sei wie oben beschrieben in $r+2$ Intervalle $(r := b - a + 1)$ zerlegt. Die Parameter a und b können gemäß den Erfordernissen der zu approximierenden Verteilung und der gewünschten Genauigkeit frei gewählt werden. Über den

Intervallen $(-\infty, a)$ und $[b+1, \infty)$ wird p mittels Verteilungen maximaler Entropie der Ordnung 1 (VME-1) approximiert . Über einem halbseitig unendlichen Intervall ist eine VME der Ordnung 1 immer durch eine geometrische Verteilung gegeben. Da nach Voraussetzung die Norm und das 1. Moment der geometrischen Verteilung über dem Intervall $(-\infty, a)$ bzw. $[b+1, \infty)$ mit der Norm und dem 1. Moment der zu approximierenden Wahrscheinlichkeitsfunktion p über dem gleichen Intervall übereinstimmen, muß gelten:

$$P_{links}(n) \;:=\; c_1 \cdot p_1 \cdot q_1^{-n-1} \quad \text{mit } q_1 := 1 - p_1,$$

$$\sum_{n=-\infty}^{a-1} P_{links}(n) \;=\; M^0_{links},$$

$$\sum_{n=-\infty}^{a-1} n \cdot P_{links}(n) \;=\; M^1_{links}$$

bzw.

$$P_{rechts}(n) \;:=\; c_2 \cdot p_2 \cdot q_2^{n-1} \quad \text{mit } q_2 := 1 - p_2,$$

$$\sum_{n=b+1}^{\infty} P_{rechts}(n) \;=\; M^0_{rechts},$$

$$\sum_{n=b+1}^{\infty} n \cdot P_{rechts}(n) \;=\; M^1_{rechts}.$$

Dabei sind M^0_{links} und M^1_{links} bzw. M^0_{rechts} und M^1_{rechts} die Norm und das 1. Moment der zu approximierenden Wahrscheinlichkeitsfunktion p über dem Intervall $(-\infty, a)$ bzw. $[b+1, \infty)$. Für die Parameter c_1 und p_1 bzw. c_2 und p_2 der geometrischen Verteilung gilt:

$$p_1 \;=\; \frac{1}{-\frac{M^1_{links}}{M^0_{links}} + a},$$

$$c_1 \;=\; \frac{M^0_{links}}{(1 - p_1)^{-a}}$$

bzw.

$$p_2 \;=\; \frac{1}{\frac{M^1_{rechts}}{M^0_{rechts}} - b},$$

$$c_2 = \frac{M^0_{rechts}}{(1-p_2)^b}$$

Über den Intervallen $[a, a+1) \cap \mathbb{Z}$, $[a+1, a+2) \cap \mathbb{Z}$, $\ldots$, $[b-1, b) \cap \mathbb{Z}$, $[b, b+1) \cap \mathbb{Z}$ wird die Wahrscheinlichkeitsfunktion p mittels r Verteilungen maximaler Entropie der Ordnung 0 (VME-0) approximiert. Eine VME der Ordnung 0 über einem Intervall $[i, i+1)$ ist immer gegeben durch eine konstante Funktion:

$$g_i(n) = g_i \quad \text{mit } g_i = M_i^0.$$

Dabei ist M_i^0 die Norm der Wahrscheinlichkeitsfunktion p über dem Intervall $[i, i+1)$. Die Wahrscheinlichkeitsfunktion p wird bei der Approximation also exakt angenommen.

4 Repräsentation von Zufallsvariablen und Verteilungen

Für die nun folgenden Darstellungen ist es notwendig, die Bedeutung von Begriffen aus der Stochastik, wie „Zufallsvariable“, „Verteilung“ und „Wertebereich“, zu erweitern, um eine zu umständliche Notation zu vermeiden.

4.1 Begriffe

Eine Zufallsvariable ist eine Funktion, die die Elemente einer Grundmenge Ω auf die Elemente einer Wertmenge $\Omega' \subseteq \mathbb{R}$ abbildet. Der Modellierer benutzt Zufallsvariablen zur Beschreibung stochastischer Größen, wie zum Beispiel die Anzahl von Kunden in einem Warteraum oder die Dauer einer Bedienung. Der Bezeichner einer Zufallsvariablen wird stellvertretend für die stochastische Größe im Analyse-Algorithmus verwendet. Der gleiche Bezeichner wird auch in der Implementation des Analyse-Algorithmus – im Analyseprogramm – Verwendung finden. Dann bezieht er sich jedoch nicht auf die Abbildung (also die Zufallsvariable im eigentlichen Sinn), sondern ist auch der Name des Objektes, das mit den Mitteln der Programmiersprache definiert wird. Objekte, die Zufallsvariablen repräsentieren, werden im folgenden deshalb ebenfalls mit dem Begriff „Zufallsvariable“ bezeichnet. Eigenschaften und Operationen, die ursprünglich in Zusammenhang mit den Objekten definiert sind (z.B. Deklaration, Initialisierung), übertragen sich in der Folge auch auf Zufallsvariablen.

Analog bezeichnen die Begriffe „Verteilung“, „Verteilungsfunktion“ und „Dichte“ nicht nur mathematische Funktionen, sondern auch die Objekte, die für ihre Repräsentation im Analyseprogramm benutzt werden. Im Laufe eines Analyseprogramms kann es erwünscht sein, mit nicht-normierten Verteilungen zu rechnen. In diesem Zusammenhang werden in diesem Handbuch an entsprechender Stelle die Begriffe „Pseudo-Verteilung“ und „Pseudo-Wahrscheinlichkeit“ benutzt.

Im Gegensatz zur Wertmenge einer Zufallsvariablen wird im folgenden mit dem Wertebereich einer Zufallsvariablen die Teilmenge von $\mathbb{R}$ bezeichnet, die sich zusammensetzt aus

1. den Elementen, denen positive Wahrscheinlichkeiten, und aus
2. den Intervallen, denen Dichtefunktionen zugeordnet werden können,

und zwar jeweils aufgrund der zur Repräsentation der Zufallsvariablen und ihrer Verteilung **deklarierten** Objekte. Der „Wertebereich“ einer Zufallsvariablen stimmt also nicht notwendigerweise mit seiner „Wertmenge“ überein; er ist eine Obermenge der Wertmenge. Die Elemente des ersten Punktes bilden den diskreten Wertebereich, die Intervalle des zweiten den stetigen Wertebereich einer Zufallsvariablen. Diskrete Zufallsvariablen verfügen somit lediglich über einen diskreten Wertebereich, stetige Zufallsvariablen besitzen nur einen stetigen Wertebereich und gemischte Zufallsvariablen sowohl einen diskreten als auch einen stetigen Wertebereich.

Ein einfaches Beispiel für den Unterschied zwischen Wertmenge und Wertebereich einer Zufallsvariable ist ein Würfel, auf dessen Seitenfläche anstelle einer „vier“ eine weitere „fünf“ angebracht ist. Die *Wertmenge* der Zufallsvariablen, die das Ergebnis eines Wurfes des so modifizierten Würfels beschreibt, enthält die Elemente '1', '2', '3', '5' und '6'. Wird diese Zufallsvariable in einem Analyseprogramm verwendet, so ist ihr *Wertebereich* die gesamte Menge $\{1, \ldots, 6\}$, da diskrete Verteilungen mit Hilfe der MODULA-Datenstruktur 'ARRAY' dargestellt werden.

Die Verwendung von 101-Dichten erlaubt, durch die Wahl unterschiedlicher Intervallbreite und -anzahl, die Dichtefunktion einer Zufallsvariablen auf mehrere Weisen zu approximieren, obwohl der stetige Wertebereich stets $I\!R$ ist.

Mit dem Begriff „mehrdimensionale Zufallsvariable“ wird eine stochastische Variable mit Werten aus $I\!R^n$ bezeichnet (vgl. [Sp82]). Eine mehrdimensionale Zufallsvariable besitzt somit einen mehrdimensionalen Wertebereich sowie eine mehrdimensionale Verteilung.

Bei der Implementierung von Analyse-Algorithmen kann es notwendig sein, Operationen auf Zufallsvariablen in mehrere Schritte zu untergliedern und die dabei in Form von Zufallsvariablen erhaltenen Zwischenergebnisse zu speichern. Da diese Zufallsvariablen nur im Analyseprogramm, jedoch nicht im Analyse-Algorithmus auftreten, werden sie analog zum Begriff der „Hilfsvariablen“ im folgenden „Hilfszufallsvariablen“ genannt. Für die Verwendung von Hilfszufallsvariablen werden von der Programmbibliothek besondere Prozeduren angeboten, auf die an späterer Stelle eingegangen wird.

4.2 Datenstrukturen für die Darstellung von Verteilungen

Der Auswahl einer geeigneten Darstellung von Zufallsvariablen und ihren Verteilungen fällt eine erhebliche Bedeutung zu. Sie legt den Bedarf an Speicherplatz fest und hat wesentlichen Einfluß auf die Laufzeit der Analyseprogramme. Außerdem soll dem Benutzer des Programmsystems ein einziger transparenter Datentyp 'Zufallsvariable' bereitgestellt werden, mit dem stetige, diskrete und gemischte Zufallsvariablen deklariert werden, wobei die diskreten Zufallsvariablen eine eindimensionale oder mehrdimensionale Verteilung besitzen können.

Als Datenstruktur für die Speicherung der Verteilungen wird in allen Fällen die MODULA-2-Struktur 'ARRAY' benutzt. Bei dieser Wahl standen die folgenden Vorteile von Feldern gegenüber anderen Datenstrukturen im Vordergrund:

1. Mit Feldindizierung kann auf alle Elemente des Wertebereichs von Zufallsvariablen wahlfrei zugegriffen werden. Es sind keine Sortier- und Suchalgorithmen notwendig, wie z.B. bei der Verwendung von verketteten Listen.
2. Greifen Analyseprogramme sequentiell auf mehrere aufeinanderfolgende Elemente eines Wertebereichs zu, so kann durch Anwendung der linearen Adreßfortschaltung der Zugriff noch weiter beschleunigt werden. Dies wird von MODULA-2 durch die Bereitstellung der Funktion 'ADDRESS' und von arithmetischen Operationen auf Adressen ermöglicht.
3. Die für die Speicherung der Verteilungen notwendigen Felder (im folgenden auch kurz „Verteilungsfelder“ genannt) werden in der Programmumgebung des Benutzers deklariert. Der Benutzer programmiert den Zugriff auf Verteilungen mittels Feldindizierung gemäß der MODULA-2-Syntax. Die programmierten Analyse-Algorithmen können so ein hohes Maß an Lesbarkeit erhalten.

In der Programmiersprache MODULA-2 sind mehrdimensionale Felder mit offenen Indexbereichen als formale Prozedurparameter nicht zugelassen. Um dennoch Prozeduren und Funktionen bereitstellen zu können, die Operationen auf beliebigen Verteilungen realisieren, wird wie folgt vorgegangen:

- Zu einer Variable vom Typ 'ZufallsVar' deklariert der Benutzer der Programmbibliothek zusätzlich alle für die Speicherung ihrer Verteilung notwendigen Felder.
- Die Startadressen und die Größe der Felder werden den verschiedenen im Programm benutzten Zufallsvariablen mit Hilfe von Initialisierungs-Prozeduren zugeordnet.

Für diskrete, stetige und gemischte Verteilungen sind unterschiedliche Zusatzinformationen notwendig. Hierauf wird in den folgenden drei Abschnitten eingegangen. Die dort genannten Komponenten bilden zusammen mit denen, die für Zwecke der Programmsteuerung notwendig sind und später besprochen werden, den Verbundtyp 'ZVStrukt'.

Die Erstellung einer Datenstruktur zur Darstellung von Zufallsvariablen, zusammen mit der erstmaligen Bereitstellung der Zusatzinformationen durch eine Inkarnation des Typs 'ZVStrukt' wird im folgenden „Initialisierung einer *Zufallsvariablen*“ genannt. Unter dem Begriff „Initialisierung einer *Verteilung*“ wird im Gegensatz dazu die erstmalige Zuweisung von Wahrscheinlichkeiten an die Komponenten eines Verteilungsfeldes verstanden.

4.2.1 Endliche diskrete Zufallsvariablen

Betrachtet werden ein- und mehrdimensionale diskrete Zufallsvariablen, deren Wertebereiche endliche Teilmengen von $\mathbb{Z}^d$ sind. Die Dimension d des Wertebereichs wird im folgenden auch die Dimension einer Verteilung oder Dimension einer Zufallsvariable genannt. Die Elemente des Wertebereiches werden durch die Komponenten eines vom Benutzer deklarierten Feldes repräsentiert. Die Werte der Komponenten sind als Wahrscheinlichkeiten zu interpretieren. Um auf die Wahrscheinlichkeiten zugreifen zu können, werden die Speicheradresse und der Indexbereich des deklarierten Feldes benötigt. Der Indexbereich wird durch zwei Tupel $(a_1, a_2, \ldots, a_d)$ und $(b_1, b_2, \ldots, b_d)$ beschrieben; zusätzlich werden die Werte $r_i := b_i - a_i + 1$, $i \in [1 : d]$, gespeichert.

4.2.2 Unendliche diskrete Zufallsvariablen

Betrachtet werden eindimensionale diskrete Zufallsvariablen mit möglicherweise unendlichem Wertebereich. Mehrdimensionale diskrete Zufallsvariablen mit unendlichem Wertebereich können nicht dargestellt werden. Zur Darstellung eindimensionaler diskreter Zufallsvariablen wird die Approximation mittels 101-Verteilungen benutzt, wie sie in Abschnitt 3.2 beschrieben wird.

Die Speicherung der Wahrscheinlichkeiten der „Einpunktintervalle“ erfolgt in einem Feld, das mit der Indexmenge {'aDiskrDef', ..., 'bDiskrDef'} vom Benutzer deklariert wird. Der Benutzer bestimmt, welcher Teil des Definitionsbereichs einer Verteilung durch Verteilungen maximaler Entropie der Ordnung 0 erfaßt wird, d.h. er legt durch die Breite des VME-0-Bereiches die mögliche Genauigkeit der Approximation fest. Analog zur Darstellung von Verteilungen endlicher diskreter Zufallsvariablen wird der Wert 'rDiskrDef' := 'bDiskrDef' - 'aDiskrDef' + 1 zusätzlich gespeichert.

Die Approximation kann auch unter Verwendung nur eines Teilbereichs des deklarierten Feldes erfolgen. Zur Beschreibung dieses benutzten Teilbereichs werden die Werte 'aDiskr', 'bDiskr' und 'rDiskr' gespeichert. Eine derartige Abweichung von deklariertem und anfänglich benutztem Wertebereich einer Zufallsvariablen kann sinnvoll sein, wenn in einem Algorithmus dieser Zufallsvariablen später eine Verteilung mit größerem Definitionsbereich zugewiesen wird.

Für die Startadresse des Feldes wird die Komponente 'VertDiskr' benutzt. Die Wahrscheinlichkeitsmassen und 1. Momente über den beiden halbseitig unendlichen Intervallen werden in den Komponenten 'M0links', 'M1links', 'M0rechts' und 'M1rechts' des Verbundes gespeichert.

4.2.3 Stetige Zufallsvariablen

Mehrdimensionale stetige Verteilungen werden nicht berücksichtigt. Für die Darstellung eindimensionaler stetiger Verteilungen wird die approximative Methode der 101-Dichten angewendet, wie sie in Abschnitt 3.1 beschrieben wird. Die Speicherung der Wahrscheinlichkeiten der endlichen Intervalle erfolgt in einem Feld, das vom Benutzer mit der Indexmenge $\{a^{Def}, a^{Def}+1, \ldots, b^{Def}\}$ deklariert ist, d.h. mit der Deklaration legt der Benutzer fest, wieviele endliche Intervalle für die Approximation maximal Verwendung finden. Neben den Parametern a^{Def} und b^{Def} wird aus den oben genannten Gründen auch bei stetigen Zufallsvariablen der Wert $r^{Def} := b^{Def} - a^{Def} + 1$ zusätzlich festgehalten. Da die Approximation nicht unbedingt unter

Verwendung des gesamten Feldes erfolgen muß, geben die Größen a, b und r in entsprechender Weise an, welcher Teil des Feldes tatsächlich verwendet wird.

Diese Unterscheidung zwischen definiertem und benutztem Indexbereich ist z.B. dann günstig, wenn eine Zufallsvariable zu Beginn eines Analysealgorithmus exponentiell verteilt ist und erst im weiteren Verlauf der Analyse ihre Verteilung durch Dichten maximaler Entropie approximiert werden muß. Ein Beispiel hierfür ist die Berechnung der Zykluslänge bei der Analyse des GG1-Modells ([Si91]).

Die Länge der Intervalle ist durch die Komponente 'delta' festzulegen. Die Speicheradresse des Feldes wird in 'VertStet' vermerkt. Für die Speicherung der Wahrscheinlichkeitsmassen und ersten Momente in den beiden Randintervallen sind die Komponenten 'Wlinks', 'Mlinks', 'Wrechts' und 'Mrechts' vorgesehen.

4.2.4 Gemischte Zufallsvariablen

Wenn der Wertebereich einer Zufallsvariablen sowohl aus stetigen Intervallen als auch aus Wahrscheinlichkeitsmassepunkten besteht, sind zur Beschreibung der Verteilung zwei separate Felder und alle in den Abschnitten 4.2.1 und 4.2.3 genannten Komponenten des Verbunds notwendig.

4.2.5 Beschreibung der Datenstrukturen in MODULA-2-Syntax

Zur Darstellung einer Zufallsvariablen werden bei diskreten und stetigen Zufallsvariablen drei, bei gemischten vier Objekte benötigt. Das erste ist eine Variable mit dem Datentyp 'ZufallsVar'. Die Variable ist vom Benutzer zu deklarieren und wirkt während ihrer Lebensdauer als Repräsentant für die Zufallsvariable. Sie ist definiert als ein Zeiger auf ein Objekt des varianten Verbundtyps 'ZVStrukt', das alle jeweils benötigten Komponenten umfaßt. Der Grund für die Benutzung eines Zeigers ist, daß einige der im Programmpaket angebotenen Prozeduren Tupel von Zufallsvariablen als Parameter benötigen. Diese Tupel sind leichter und schneller zu handhaben, wenn Zufallsvariablen durch Zeiger repräsentiert werden. Die verbleibenden Objekte sind die Felder zur Speicherung der diskreten Verteilung bzw. der Approximation der Dichte der Zufallsvariablen. Die Abbildungen 2 und 3 verdeutlichen graphisch, auf welche Weise die einzelnen Datenstrukturen zur Speicherung von Zufallsvariablen miteinander in Beziehung stehen.

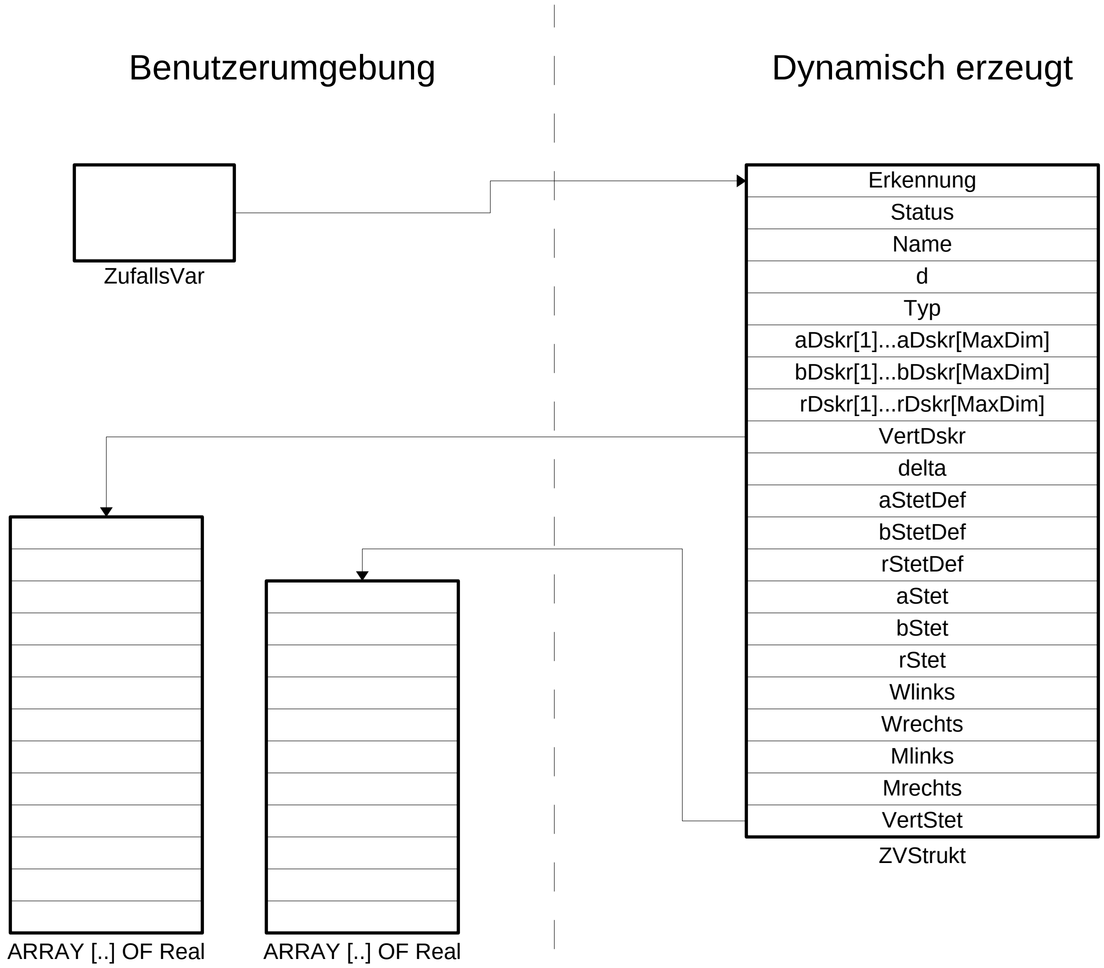

Abbildung 2: Darstellung einer eindimensionalen Zufallsvariablen mit gemischter Verteilung

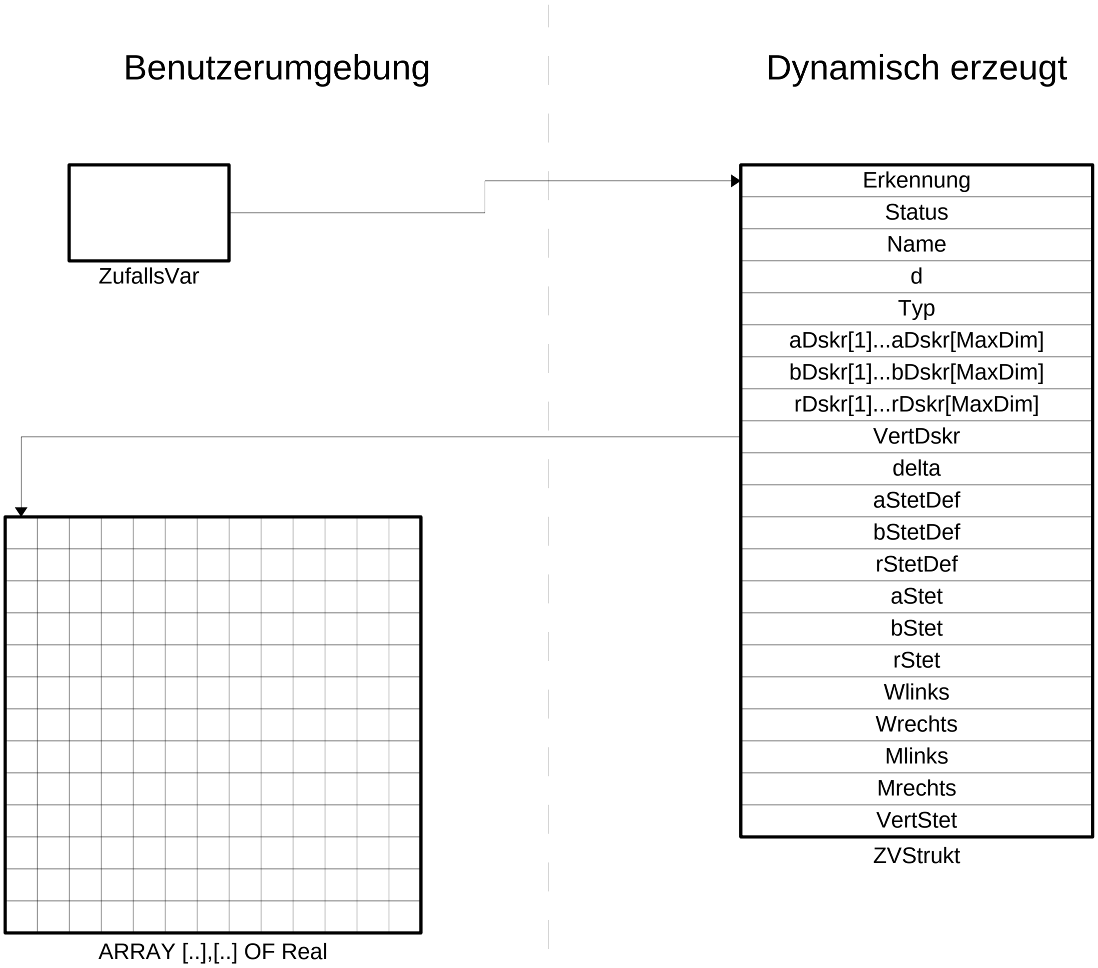

Abbildung 3: Darstellung einer zweidimensionalen Zufallsvariablen

Die Definition des Typs 'ZVStrukt' lautet in MODULA-2-Syntax:

```
TYPE ZVStrukt = RECORD

                  Erkennung: INTEGER;     (* Zur Erkennung von korrekt
                                             initialisierten Zufallsvari-
                                             ablen                        *)
                  Status   : ZVStatusSet; (* Zeigt an, ob die Zufalls-
                                             variable statisch oder
                                             dynamisch erzeugt wurde,
                                             und ob die Verteilung seit
                                             dem letzten Nullsetzen
                                             veraendert wurde             *)
                  Name     : ZVName;      (* Benutzerdefinierter Name der
                                             Zufallsvariablen             *)
                  d        : INTEGER;     (* Dimension der Verteilung     *)

             CASE Typ : ZVTyp OF


               DiskrEndl .. Gemischt :

                  (* Variablen fuer diskrete Verteilungen mit endlichem
                     Wertebereich                                         *)

                  aDskr,bDskr,
                  rDskr: ITupel;      (* Beschreibung des Wertebereichs   *)
                  VertDskr: ADDRESS;  (* Adresse des Arrays zur Speicherung
                                         der diskreten Verteilung. Falls
                                         dieser nicht vorhanden ist, so ist
                                         VertDskr = NIL                   *)

                  (* Variablen fuer stetige Verteilungen                  *)
                  delta: Real;
                  aStetDef,bStetDef,
                  rStetDef      : INTEGER;(* Beschreibung des maximalen
                                             Bereiches fuer DME der
                                             Ordnung 0                    *)
                  aStet,bStet,
                  rStet         : INTEGER;(* Benutzter Bereich            *)
                  Wlinks, Wrechts: Real;   (* Wahrscheinlichkeiten
                                             P{X < a*delta} und
                                             P{X >= (b+1)*delta}          *)
                  Mlinks, Mrechts: Real;   (* 1. Momente                  *)
                  VertStet      : ADDRESS;(* Adresse des Arrays fuer die
                                             P{i*delta <=X< (i+1)*delta} *)
```

```
              | Diskr101 :

                   (* Variablen fuer diskrete Verteilung mit unendlichem
                      Wertebereich                                      *)
                   aDiskrDef,bDiskrDef,
                   rDiskrDef        : INTEGER;(* Beschreibung des maximalen
                                                 Bereichs fuer VME der
                                                 Ordnung 0 *)
                   aDiskr,bDiskr,
                   rDiskr           : INTEGER;(* Benutzter Bereich          *)
                   M0links,M0rechts: Real;   (* Wahrscheinlichkeiten
                                                 P{X < a} und P{X > b}      *)
                   M1links,M1rechts: Real;   (* 1. Momente                  *)
                   VertDiskr        : ADDRESS;(* Adresse des Arrays fuer die
                                                 P{X_i}                     *)
          END;
END;
```

Außer den Komponenten, mit deren Hilfe Wertebereiche festgelegt oder Zugriffe auf Felder kontrolliert werden, umfaßt der Verbund weitere Komponenten.

Die Komponente 'Erkennung' dient dem Programmsystem zur Kennzeichnung korrekt angelegter Zufallsvariablen.

Die Komponente 'Status' zeigt dem Programmsystem an, ob es sich um eine dynamisch erzeugte Zufallsvariable handelt und ob die Verteilung der Zufallsvariablen seit der letzten Nullsetzung verändert wurde (vergleiche Abschnitt 4.4). Die Kennzeichnung dynamisch mit 'GenZV' und 'DelZV' erzeugter Zufallsvariablen ist nötig, da diese von einigen Prozeduren eine gesonderte Behandlung erfordern.

In 'Name' wird ein maximal 15 Zeichen langer Name gespeichert. Dieser Name wird vom Programmsystem bei der Meldung von Fehlern und der Ausgabe von Verteilungen benutzt.

In der Komponente 'Typ' wird der Typ der Zufallsvariablen bei der Initialisierung eingetragen. Die möglichen Werte, die dieser zugewiesen werden können, sind die Elemente des Aufzählungstyps 'ZVTyp', welche sind: 'DiskrEndl', 'Stet101', 'Gemischt' und 'Diskr101'. Abhängig vom Typ der Zufallsvariablen wird eine der Varianten im Verbund 'ZVStrukt' ausgewählt.

Die erste Variante enthält alle Komponenten, die zur Speicherung der Verteilung einer endlichen diskreten, einer stetigen oder gemischten Zufallsvariable benötigt werden. Die zweite Variante enthält alle Komponenten, die zur Speicherung der Verteilung einer diskreten unendlichen Zufallsvariablen benötigt werden. Die Aufteilung in zwei Varianten führt zu einer Verringerung des Speicherbedarfs für die Inkarnierung von Variablen des Typs 'ZVStrukt'. Aus programmiertechnischen Gründen wurde von einer Aufteilung von 'ZVStrukt' für diskret endliche, stetige und gemischte Zufallsvariablen auf verschiedene Varianten abgesehen, da dies bis auf die Änderung der Komponentennamen zu einer Wiederholung des Programmtexts aller Operationen auf gemischten Zufallsvariablen bei stetigen und diskret endlichen Zufallsvariablen führte. Diese Wiederholung wäre bei der Aufteilung auf verschiedene Varianten notwendig, da in MODULA-2 Komponenten unterschiedlicher Varianten eines Verbunds nicht die gleichen Namen haben dürfen.

Der Datentyp 'ITupel' (INTEGER-Tupel), der für die Deklaration der Komponenten 'aDskr', 'bDskr' und 'rDskr' benutzt wird, definiert eine Feldstruktur wie folgt:

```
TYPE ITupel   = ARRAY[0..MaxDim] OF INTEGER;
```

Die Komponente '[0]' dieses Feldes enthält die Anzahl der Elemente des Tupels, die übrigen Komponenten enthalten die Elemente selbst. Von der Programmbibliothek werden Prozeduren zur Verfügung gestellt, mit denen übersichtlich Paare, Tripel, Quadrupel usw. erzeugt werden können (Abschnitt 5.6.4).

Der Datentyp 'Real', der in der Definition von 'ZVStrukt' verwendet wird, entspricht nicht dem MODULA-2-Standardtyp 'REAL'. Stattdessen ist er innerhalb der Programmbibliothek vereinbart und wird für alle Programmvariablen benutzt, die reelle Zahlen darstellen. Auf diese Weise läßt sich die Programmbibliothek leichter durch eine andere Definition des Typs 'Real' auf unterschiedliche Genauigkeitsansprüche anpassen, ohne daß im allgemeinen am Analyseprogramm selbst Änderungen vorgenommen werden müssen (siehe auch Abschnitt 4.5).

4.2.6 Deklaration von Zufallsvariablen

Wie schon im vorhergehenden Abschnitt erläutert wurde, erfordert die Benutzung einer Zufallsvariablen mehrere Variablendeklarationen im Analyseprogramm. Durch den Aufruf einer Initialisierungsprozedur wird zusätzlich ein dynamisches Objekt vom Typ 'ZVStrukt' erzeugt und die Verbindung zwischen allen beteiligten Objekten hergestellt. Im Falle einer eindimensionalen, endlichen diskreten Zufallsvariable X mit Wertebereich $[0 : 10]$ sind die folgenden Deklarationen notwendig:

```
VAR X        : ZufallsVar;
    pX       : ARRAY [0..10] OF Real;
```

Die Namensgebung für die Variablen liegt selbstverständlich vollständig in den Händen des Benutzers und kann nach den jeweils gegebenen Erfordernissen vorgenommen werden. Es wird aber vorgeschlagen, daß der Name der Zufallsvariablen, repräsentiert durch ein Objekt vom Typ 'ZufallsVar', mit einem Großbuchstaben beginnt, dem gegebenfalls Klein- oder Großbuchstaben folgen. Der Name eines Feldes für eine diskrete Verteilung sollte aus dem Namen dieser Zufallsvariablen und einem vorangestellten 'p' bestehen; als Name für Felder zur Approximation stetiger Verteilungen sollte der Name der Zufallsvariablen mit einem vorangestellten 'f' verwendet werden.

Der Datentyp 'Real' ist als Typ für die Komponenten des Feldes für die Verteilung vorgeschrieben.

Im Anweisungsteil des Analyseprogramms muß eine Initialisierung der Zufallsvariablen X erfolgen, bevor sie als Parameter von Bibliotheksprozeduren verwendet werden kann. Für die Initialisierung von eindimensionalen, endlichen diskreten Zufallsvariablen wird die Prozedur 'InitDskrZV1D' (Initialisiere endliche diskrete Zufallsvariable, 1-dimensional) benutzt. Als Parameter verlangt die Prozedur eine Variable vom Typ 'ZufallsVar', die Adresse des zugehörigen Feldes für die Speicherung der Verteilung, sowie den ersten und letzten Indexwert dieses Feldes. Sie erzeugt ein Objekt vom Typ 'ZVStrukt', dessen Adresse dem aktuellen Parameter 'X' zugewiesen wird. Den Komponenten des Objekts werden die aus den restlichen aktuellen Parametern berechneten Werte bzw. die Werte '0', '0.0' oder 'NIL' zugewiesen. Außerdem werden alle Komponenten der Felder zur Speicherung der Verteilung mit dem Wert '0.0' initialisiert. Letzteres wird mit „Nullsetzen“ bezeichnet. Die Initialisierung zum oben stehenden Beispiel lautet:

```
InitDskrZV1D(X,'X ',ADR(pX),0,10);
```

Für die Initialisierung von Zufallsvariablen mit mehrdimensionaler, diskret unendlicher, stetiger oder gemischter Verteilung stehen die Prozeduren 'InitDskrZVMD' (Initialisiere diskrete endliche Zufallsvariable, mehrdimensional),'InitDskrUnendlZV' (Initialisiere diskrete unendliche Zufallsvariable), 'InitStetZV' (Initialisiere stetige Zufallsvariable) und 'InitGemZV' (Initialisiere gemischte Zufallsvariable) zur Verfügung, deren genaue Beschreibung in den Abschnitten 5.7.8, 5.7.9, 5.7.10 und 5.7.11 zu finden ist. Im folgenden Programmausschnitt sind die Deklaration und die Initialisierung jeweils einer mehrdimensionalen, einer diskret unendlichen, einer stetigen und einer gemischten Zufallsvariablen zusammengefaßt:

```
VAR M     : ZufallsVar;       (* mehrdimensional *)
    pM    : ARRAY[0..5],[-3..3],[-5..1] OF Real;

    ug,og: ITupel;            (* zur Festlegung der Indexgrenzen *)

    DU    : ZufallsVar;       (* diskret unendlich *)
    pDU   : ARRAY [5..20] OF Real;

    S     : ZufallsVar;       (* stetig *)
    fS    : ARRAY[-100,99] OF Real;

    G     : ZufallsVar;       (* gemischt *)
    pG    : ARRAY[0..0] OF Real;
    fG    : ARRAY[0..49] OF Real;
BEGIN
  ...
  BildeITupel3(ug, 0,-3,-5);
  BildeITupel3(og, 5, 3, 1);
  InitDskrZVMD(M,'M ',ADR(pM),ug,og);

  InitDskrUnendlZV(DU,'DU ',ADR(pDU),5,20);

  InitStetZV(S,'S ',ADR(fS),-100,99,0.1);         (* delta = 0.1 *)
                        (* entspricht dem DME-0-Bereich [-10,10) *)

  InitGemZV(G,'G ',ADR(pG),0,0,ADR(fG),0,49,0.5); (* delta = 0.5 *)
                          (* entspricht dem DME-0-Bereich [0,25) *)
  ...

END ... .
```

4.2.7 Speicherplatzbeschränkungen

Das obige Beispiel für eine eindimensionale diskrete Zufallsvariable sieht dann wie folgt aus:

```
TYPE TypVertlngX: ARRAY [0..1000] OF Real;

VAR  X : ZufallsVar;
     pX: POINTER TO TypVertlngX;

BEGIN
  ...
  ALLOCATE(VertlngX,SIZE(TypVertlngX));  (* Zuweisung von Speicherplatz *)
  InitDskrZV1D(X,'X ',pX,0,1000);        (*       Initialisierung       *)

  ...

  DEALLOCATE(pX);                    (* Freigabe des belegten Speichers *)
END ... .
```

Beschränkungen, die sich daraus ergeben, daß ein einziger deklarierter Typ einen bestimmten Bedarf an Speicherplatz nicht überschreiten darf, können nur umgangen werden, indem ein zusammenhängender Speicherbereich der gewünschten Größe mittels der 'ALLOCATE'-Prozedur reserviert wird, ohne vorher einen entsprechenden Typ zu deklarieren. Die Größe des Bereiches ist dann aus der Anzahl der Komponenten und dem Speicherplatz für eine Komponente – dieser liegt als Wert in der Variablen 'RealSize' vor – zu bestimmen. Aber auch die Reservierung von Speicherplatz durch die 'ALLOCATE'-Prozedur ist bei einigen Compilern Beschränkungen unterworfen.

Eine weitere Möglichkeit ist, Zufallsvariablen dynamisch mit der Prozedur 'DynZV' zu erzeugen (vergleiche Abschnitt 5.7.15). Wenn bereits eine Zufallsvariable mit passendem Wertebereich als „Schablone" vorliegt, so kann auch die Prozedur 'GenZV' benutzt werden (vergleiche 5.7.13).

4.3 Zugriff auf Verteilungen

Im folgenden wird beschrieben, wie man die Elemente des Verteilungsfeldes einer Zufallsvariablen durch Feldindizierung oder mit einem Objekt des Typs 'RZugriff' (s.u.) anspricht. Einzelne Elemente von Verteilungsfeldern anzusprechen ist z.B. dann notwendig, wenn man bestimmte Wahrscheinlichkeiten verändern möchte. Allerdings wird für diesen Zweck von VERENA die Prozedur 'Pgleich' zur Verfügung gestellt (Abschnitt 5.8.5). Die Benutzung dieser Prozedur ist den hier beschriebenen Möglichkeiten zur Manipulation von Verteilungen vorzuziehen, da sie komfortabler ist und den Benutzer von einigen technischen Aufgaben befreit (Überprüfung, ob Index korrekt gewählt o.ä.). Sollte jedoch ein möglichst schneller Zugriff auf das Verteilungsfeld notwendig sein, so kann man auf die in diesem Abschnitt beschriebenen Möglichkeiten für die Veränderung einzelner Wahrscheinlichkeiten zurückgreifen.

Der Benutzer der Programmbibliothek kann bei der Programmierung von Analyse-Algorithmen auf die durch die Verteilung einer Zufallsvariablen gegebenen Wahrscheinlichkeiten mittels Feldindizierung zugreifen, da die Verteilungsfelder, die die Wahrscheinlichkeiten enthalten, im Deklarationsteil des Analyseprogramms deklariert werden und somit innerhalb des Programmkörpers sichtbar sind:

```
MODULE Analyse;

... (* Importlisten *)

VAR  X  : ZufallsVar;
     pX : ARRAY [0..20] OF Real;  (* diskrete Zufallsvariable mit  *)
                                  (* Wertebereich [0..20]         *)
...

BEGIN

  InitDskrZV1D(X,'X ',ADR(pX),0,20);
  ...
  pX[5] := RK(0.2);               (* P(X=5) := 0.2                    *)
  ...                             (* Die Funktion RK() konvertiert REAL *)
                                  (* nach Real, s. Abs. 'Arithmetik'    *)
END Analyse.
```

In einer Umgebung, in der die Deklaration des zu manipulierenden Verteilungsfeldes nicht sichtbar ist, muß anders vorgegangen werden. Die Datenstruktur 'ZVStrukt' enthält die Anfangsadressen der vorhandenen Verteilungsfelder, die Start- und Endindizes. Aus diesen Informationen können die Speicheradressen beliebiger Elemente des Wertebereiches berechnet werden. Sei im folgenden 'X' eine Variable vom Typ 'ZufallsVar' und 'VertX' vom Typ 'RZugriff'. Dieser Typ wird von dem Modul 'Reals' exportiert:

```
TYPE RZugriff = RECORD
                  CASE (**):(**) BOOLEAN OF
                    TRUE:  Adr : ADDRESS;
                  | FALSE: Elem : POINTER TO Real;
                  END; (* CASE *)
                END; (* RECORD *)
```

'RZugriff' ist ein unbenannter varianter Verbund. Die Zeichenfolge '(∗∗)' ist lediglich ein Kommentar. Diese Kommentare dienen der besseren Lesbarkeit und sollen hervorheben, daß es sich bei diesem varianten Verbund um einen *unbenannten* varianten Verbund handelt. Die Deklaration von 'RZugriff' als unbenannter varianter Verbund besitzt den Vorteil, daß auf die Komponenten direkt mit Punktnotation zugegriffen werden kann.

Sei 'VertX' ein Objekt des Typs 'RZugriff'. Dann kann 'VertX.Adr' eine Adresse zugewiesen werden. Durch Dereferenzierung des Zeigers 'VertX.Elem↑' kann auf den 'Real'-Wert zugegriffen werden. Die Adresse der Speichereinheit, in der der Wert $P\{X = i\}$ abgelegt ist, wird bei eindimensionalen Wertebereichen wie folgt berechnet:

```
VertX.Adr   :=   X^.VertDskr
               + RealSize * VAL(ADDRESS,i-X^.aDskr[1]);

VertX.Elem^ :=   p;     (* P(X=i) := p *)
```

'X↑.VertDskr' ist die Anfangsadresse des Verteilungsfeldes von X, 'X↑.aDskr[1]' der Index der ersten gespeicherten Wahrscheinlichkeit und 'RealSize' der Speicherbedarf einer Variablen des Typs 'Real'.

Beispiel für den Zugriff auf die Verteilung einer diskret unendlichen Zufallsvariablen (Zufallsvariable vom Typ 'Diskr101'):

```
VertX.Adr   :=   X^.VertDiskr
               + RealSize*VAL(ADDRESS,i-X^.aDiskrDef);
```

In ähnlicher Weise wird auf stetige Wertebeiche zugegriffen:

```
VertX.Adr   :=   X^.VertStet
               + RealSize * VAL(ADDRESS,i-X^.aStetDef);

VertX.Elem^ :=   p;     (* P(i*X^.delta <= X < (i+1)*X^.delta) := p *)
```

Dabei ist 'VertStet' die Startadresse des Feldes für die stetige Verteilung, 'X↑.aStetDef' der Index des ersten Feldelements und 'X↑.delta' die Länge der Intervalle, mit denen die stetige Verteilung approximiert wird.

Bei mehrdimensionalen diskreten Wertebereichen wird ein Element des Wertebereiches durch ein Tupel **v** mit den Elementen 'v[1]',...,'v[X↑.d]' spezifiziert. Bei der Berechnung der Speicheradresse wird wie folgt vorgegangen:

```
pX.Adr := X^.VertDskr;     (* Adresse des ersten Elements            *)
Dope   := RealSize;        (* Dope ist vom Typ 'ADDRESS'             *)

FOR j := X^.d TO 1 BY -1 DO   (* Adresse des Elements v berechnen    *)
  INC(pX.Adr,Dope * VAL(ADDRESS,v[j]-X^.aDskr[j]);
  INC(Dope,VAL(ADDRESS(X^.rDskr[j]));
END;

pX.Elem^ := w       (* P(X=v) := w *)
```

Sollen nacheinander die Adressen mehrerer Elemente des Wertebereiches berechnet werden, so empfiehlt es sich, die Werte der Variablen 'Dope' in einem Feld (Dopevektor) abzuspeichern und für weitere Adreßberechnungen zu nutzen.

Beim Zugriff auf die Elemente der Verteilungsfelder muß der Benutzer darauf achten, daß er keinen Zugriff auf eine Feld-Komponente vornimmt, die außerhalb des deklarierten Feldes liegt, da dies zu Laufzeitfehlern führt!

Bei der Implementierung der Prozeduren der Programmbibliothek hat sich gezeigt, daß Operationen auf Zufallsvariablen in vielen Fällen sequentiell auf alle Komponenten eines Verteilungsfeldes zugreifen. Als einfaches Beispiel sei hier die Nullsetzung (hierzu der folgende Abschnitt) einer eindimensionalen, diskreten Verteilung angeführt. Alle Wahrscheinlichkeiten des Verteilungsfeldes der Zufallsvariablen X sollen auf den Wert '0.0' gesetzt werden:

```
pX.Adr := X^.VertDskr;   (* Zeigt auf Anfang des Verteilungsfeldes    *)
FOR i := X^.aDskr[1] TO X^.bDskr[1] DO
  pX.Elem^ := RK0;        (* P(X=i) := 0.0 *)
  INC(pX.Adr,RealSize);  (* Zeiger auf die naechste Komponente setzen *)
END;
```

Diese Technik der Adreßberechnung wird lineare Adreßfortschaltung genannt.

4.4 Nullsetzung von Verteilungen

Einige Prozeduren, die Operationen auf Zufallsvariablen ausführen, setzen voraus, daß vor Beginn der eigentlichen Berechnung des Ergebnisses die Wahrscheinlichkeiten aller Elemente des Wertebereiches der Zielverteilung auf den Wert Null gesetzt werden. Zu diesen Operationen gehören die Faltung, die Zuweisung, die Minimum- und Maximumbildung. Dieser Vorgang heißt Nullsetzung der Verteilung oder Nullsetzung der Zufallsvariablen und wird von den oben genannten Prozeduren automatisch veranlaßt. Das ist zu unterscheiden von der *Initialisierung* einer Zufallsvariablen, durch die zusätzlich eine Datenstruktur für eine neue Zufallsvariable erzeugt und initialisiert wird.

Damit Verteilungen nicht unnötig oft nullgesetzt werden und um dem Modellierer die Möglichkeit offen zu lassen, die Zielverteilung vor dem Ausführen einer Operation im nicht nullgesetzten Zustand zu belassen, besitzt die Komponente 'Status' der Datenstruktur 'ZVStrukt' das Attribut 'Vert0', das nach jedem Nullsetzen einer Verteilung gegeben und vor jedem Nullsetzen abgefragt wird.

Alle Prozeduren und Programmstücke, die eine Verteilung verändern, müssen der betroffenen Zufallsvariable dieses Attribut nehmen. Dies geschieht mittels der Prozedur 'VertVeraendert', die das Modul 'RVBasics' exportiert. Wenn in einem Analyseprogramm Elementen des Wertebereiches einer Zufallsvariablen 'X' Wahrscheinlichkeiten zugewiesen werden, ohne daß die von VERENA dafür zur Verfügung gestellten Prozeduren benutzt werden, muß nach der letzten Zuweisung die Anweisung

```
VertVeraendert(X);
```

folgen. Anderenfalls kann eine nachfolgend aufgerufene Operation mit 'X' als Ergebnis-Zufallsvariable fehlerhafte Ergebnisse liefern, da die Verteilung vor der Bearbeitung nicht nullgesetzt wird. Das Prüfen des Attributs 'Vert0' und die Nullsetzung geschieht in der Prozedur 'Setze0Vert'. Möchte der Benutzer verhindern, daß eine Verteilung nullgesetzt wird, kann er dies durch einen Aufruf der Prozedur 'NichtNullsetzen' bewirken, die ebenfalls vom Modul 'RVBasics' exportiert wird.

Bei allen Prozeduren, die im allgemeinen lediglich Pseudo-Verteilungen berechnen und die dabei erhaltenen Wahrscheinlichkeiten zu denjenigen der Ergebnis-Zufallsvariablen addieren, ist der Modellierer selbst für die Nullsetzung der Verteilung der Ziel-Zufallsvariablen verantwortlich. Eine dieser Prozeduren ist 'SummWGewichtet', die vom Modul 'RndVar0' exportiert wird. Sie wird benutzt, um aus mehreren Verteilungen unter Berücksichtigung von Gewichten die Verteilung einer Zufallsvariablen zu berechnen, indem für jedes Element des Wertebereiches der Verteilung eine gewichtete Summe von Wahrscheinlichkeiten gebildet wird. Hierzu muß die Prozedur gemäß der Anzahl der Gewichtungen mehrfach aufgerufen werden. Vor dem ersten Aufruf muß die Verteilung der Ziel-Zufallsvariablen nullgesetzt sein. Im nachfolgenden Beispiel werden die gewichteten Summen aus den Verteilungen der Zufallsvariablen 'V[1]' bis 'V[n]' und den Gewichten 'w[1]' bis 'w[n]' gebildet. Sie ergeben die Verteilung der Ergebnis-Zufallsvariablen 'Z'.

```
Setze0Vert(Z);                 (* Nullsetzung der Verteilung von Z *)
FOR i := 1 TO n DO             (* Anzahl der Summanden             *)
  SummWGewichtet(V[i],w[i],Z) (* Bildung der gewichteten Summe    *)
END; (* FOR *)
```

Der Aufruf von 'Setze0Vert' durch den Benutzer ist nicht immer notwendig. Im Beispiel etwa dann, wenn die Zufallsvariable 'Z', deren Verteilung berechnet werden soll, unmittelbar vor dem ersten Aufruf von 'SummWGewichtet' initialisiert wurde. Die Initialisierung von Zufallsvariablen umfaßt stets auch deren Nullsetzung.

Neben 'SummWGewichtet' sind weitere Prozeduren, bei deren Verwendung der Modellierer selbst für die Nullsetzung der Ergebnis-Zufallsvariablen sorgen muß, 'BesetzTWbereichKonst', 'BesetzTWbereichPMF', 'DME1LAufteilen', 'DME1RAufteilen', 'GemeinsamAbhZV' und 'GemeinsamUnZV'.

4.5 Fließkomma-Arithmetik

Das Konzept der Programmbibliothek sieht vor, daß der Datentyp, der für die Darstellung aller Fließkommawerte verwendet wird, mit wenig Aufwand zu ändern ist, um verschiedenen Genauigkeitsanforderungen gerecht zu werden.

In den Programmen der Bibliothek und in den Analyseprogrammen selbst ist für die Darstellung von Fließkommawerten der Datentyp 'Real' vorgeschrieben. Dieser Typ wird vom Modul 'Reals' exportiert.

Bei einer Änderung der verwendeten Fließkommadarstellung - z.B. von 'LONGREAL' auf 'REAL' - muß die Programmbibliothek in folgenden Punkten verändert werden:

- Änderung der Definition des Typs 'Real' im Deklarationsmodul 'Reals.def'.
- Umgestalten der Konvertierungsroutinen 'R' und 'RK' unter Berücksichtigung der Eigenheiten des verwendeten Compilers im Implementationsmodul 'Reals.mod'.
- Im Implementierungsmodul 'Math.mod' ist die Importierung anderer, auf die gewählte Darstellung angepaßter Prozeduren erforderlich. Zum Beispiel ist anstatt der Prozedur 'LongMathLib.Ln' die Prozedur 'MathLib.Ln' zu importieren.

4.6 Portabilität

Die Programmbibliothek wurde auf vier verschiedenen Rechenanlagen mit ebenfalls unterschiedlichen MODULA-2-Compilern implementiert. Dabei wurden einige Unterschiede zwischen den Compilern festgestellt, die bei der Implementierung zu berücksichtigen waren. Der größte Teil der Unterschiede betrifft die Benennung von Standard-Modulen und die von diesen exportierten Prozeduren. Durch die Verwendung von Anpassungs-Modulen wie 'IOProc' und 'Math' werden diese Unterschiede vor dem Benutzer der Programmbibliothek verdeckt, so daß Analyseprogramme zumindest in dieser Hinsicht von allen Compilern verarbeitet werden können.

Im Anpassungs-Modul 'IOProc' sind zur Zeit nur die Ausgaberoutinen berücksichtigt. Die Einbeziehung von Eingabeprozeduren, möglicherweise mit komfortablen Prozeduren für das Einlesen von Modellparameter-Dateien, kann zu einem späteren Zeitpunkt erfolgen.

Es existieren zwischen den Compilern aber auch Unterschiede in der Syntax und bei der Implementation von Low-Level-Prozeduren, die entweder nicht verdeckt werden können oder nicht verdeckt wurden, weil sie zu selten in Analyseprogrammen zum Tragen kommen. In Tabelle 1 sind diese Unterschiede der einzelnen MODULA-2-Implementationen, soweit sie das Programmieren von Analysealgorithmen betreffen könnten, gegenübergestellt.

SUN-MODULA-2 **UNIX**	**VAX-MODULA-2** **VAX-VMS**	**SPC-MODULA 2.0** **ATARI ST/TOS**	**Logitech MODULA-2** **PC-DOS/MS-DOS**
Verwendung der EXPORT QUALIFIED-Anweisung			
erforderlich	erforderlich	nein	erforderlich
Import der Systemfunktion VAL			
Import nicht notwendig	Import nicht notwendig	aus Modul 'SYSTEM'	Import nicht notwendig
Typüberlagerung in RECORD-Strukturen			
CASE : BOOLEAN OF	CASE BOOLEAN OF	CASE : BOOLEAN OF	CASE BOOLEAN OF
Parameter der Prozedur DEALLOCATE			
(Adresse,Anzahl)	(Adresse,Anzahl)	(Adresse)	(Adresse,Anzahl)
Einschränkungen bei CONST-Deklarationen			
keine	keine	LONGINT,LONGREAL *nicht* erlaubt	keine

Tabelle 1: Nicht „verdeckte“ Unterschiede der MODULA-2 Compiler

5 Die Bibliotheksmodule

Dieser Abschnitt stellt die einzelnen Module der Bibliothek vor. Im folgenden werden die von den Modulen exportierten Typen, Variablen und Prozeduren beschrieben und Beispiele für deren Verwendung angegeben. Soweit es notwendig ist, wird auch die Funktionsweise einzelner Prozeduren beschrieben. In den Modulen sind folgende Objekte implementiert:

Datenstrukturen:

- diskrete endliche Zufallsvariablen mit endlicher Wertmenge $\subset \mathbb{Z}^n$, $n \in [1:10]$ (vergleiche 5.6.1)
- diskrete unendliche Zufallsvariablen mit unendlicher Wertmenge $\subset \mathbb{Z}$
- eindimensionale stetige Zufallsvariablen
- eindimensionale gemischte Zufallsvariablen

Verteilungen:

- Einpunktverteilung (Konstante)
- diskrete Gleichverteilung
- Bernoulli-Verteilung
- Binomialverteilung
- geometrische Verteilung
- Poisson-Verteilung
- stetige Gleichverteilung
- Exponentialverteilung
- Hyperexponentialverteilung
- Erlang-K-Verteilung
- Normalverteilung

Operationen:

- Berechnung von Erwartungswerten und Momenten
- Addition stochastisch unabhängiger Zufallsvariablen (Faltung ihrer Verteilungen)
- Bildung von Minima und Maxima einer Zufallsvariablen mit einer Konstanten
- Berechnung von Verteilungen durch Anwenden des Satzes der totalen Wahrscheinlichkeit
- Normierung von Verteilungen
- Verknüpfung von Zufallsvariablen mit der Abbildung $f : I\!R \to I\!R, f(x) := -x$
- Ermittlung von Randverteilungen
- Bildung von gemeinsamen Verteilungen
- Realisierung von Zustandsübergangsfunktionen

Ausgabe:

- tabellarische Ausgabe von Verteilungen
- graphische Darstellung von Verteilungen in PostScript-Syntax zur Einbindung in LaTeX-Dokumente

Sonstige Funktionen:

- komplexe Zahlen mit den Operationen Addition und Multiplikation, Transformation aus der Gauß'schen Zahlenebene in Polarkoordinaten, sowie Rücktransformation
- schnelle Fourier-Transformation
- Werkzeuge zur Fehlersuche

Alle Objekte des Programmsystems sind gemäß ihrer Funktion auf die Module aufgeteilt. Tabelle 2 gibt einen Überblick über alle Module mit ihren Aufgabengebieten. Für die Erstellung von Analyseprogrammen sind im wesentlichen die Module 'IOProc', 'Reals', 'Tupel', 'RVBasics' sowie 'RndVar0' bis 'RndVar8' von Bedeutung.

Die formalen Parameter aller Prozeduren sind in den Deklarationen so geordnet, daß zunächst alle Eingabeparameter genannt werden und danach die Ausgabeparameter. Innerhalb dieser beiden Gruppen erscheinen formale Parameter vom Typ 'ZufallsVar' immer vor Parametern anderen Typs.

Die Module, die eine Namenserweiterung 'E' besitzen (z.B. MathE, RndVar0E etc.), enthalten die Objekte, die nur die Zufallsvariablen mit *diskret unendlichem Wertebereich* betreffen. Der Benutzer darf diese Module **nicht** importieren, auch wenn er Zufallsvariablen mit diskret unendlichem Wertebereich benutzt. Diese Module werden von den zugehörigen Modulen ohne die Namenserweiterung implizit importiert und stehen damit dem Benutzer ebenfalls zur Verfügung. Da die Einbindung der Objekte aus den Erweiterungsmodulen verdeckt geschieht, wird bei der Beschreibung eines Moduls auf sein Erweiterungsmodul nicht eingegangen; es wird zum Beispiel nur von einem Modul 'Math' oder 'RndVar0' etc. gesprochen.

Fehler FehlerE	Ausgabe von Fehlermeldungen und Bereitstellung eines Stacks zur Speicherung von Laufzeitinformationen (insbesondere Namen der aufgerufenen Prozeduren)
IOProc	Schnittstelle zu den vom verwendeten Compiler angebotenen Ein- und Ausgabeprozeduren
Math MathE	Schnittstelle zu den vom verwendeten Compiler angebotenen mathematischen Funktionen und Bereitstellung zusätzlicher Funktionen
Misc	Stringverarbeitung
Reals	Bereitstellung des Datentyps 'Real' und Konvertierungsoperationen zu diesem Typ
Tupel	Bereitstellung des Datentyps 'ITupel' (INTEGER-Tupel) mit Zuweisungs- und Vergleichsoperationen und des Typs 'RTupel'
RVBasics RVBasicE	Bereitstellung des Datentyps 'ZufallsVar' und Initialisierungsoperationen auf diesem Typ
RndVar0 RndVar0E	Allgemeine Operationen für die Festlegung von Verteilungen
RndVar1 RndVar1E	Ausgabe von Zufallsvariablen als Tabelle oder als Funktionsgraph
RndVar2 RndVar2E	Zuweisung von Verteilungen von Zufallsvariablen; Momente von Verteilungen; Minimum- und Maximum-Operationen
RndVar3 RndVar3E	Erzeugung von oft benötigten Verteilungen
RndVar4 RndVar4a RndVar4b RndVar4c RndVar4d RndVar4e RndVar4f RndVar4g RndVar4h	Faltung von Verteilungen von Zufallsvariablen
RndVar5 RndVar5E	Zuweisung von Zufallsvariablen mit unterschiedlich strukturierten Wertebereichen
RndVar6	Konvergenzprüfung
RndVar7	Berechnung von Zustandswahrscheinlichkeiten in Markovketten
RndVar8 RndVar8E	Operationen auf mehrdimensionalen Zufallsvariablen

Tabelle 2: Programm-Module und ihre Aufgabengebiete

5.1 Das Modul 'Fehler'

Das Modul 'Fehler' stellt Prozeduren zur Fehlerausgabe und zur Erleichterung der Fehlersuche zur Verfügung. Mittels eines Stackspeichers können Zeichenketten (z.B. Unterprogrammnamen) nach dem LIFO-Prinzip (Prozeduren 'PushStr' und 'PopStr') gespeichert werden. Nach Aufruf der Prozedur 'MeldeFehler' wird neben der übergebenen Fehlermeldung auch der Inhalt des Stackspeichers ausgegeben. Bei eingeschaltetem Trace-Modus (Variable 'TraceMode' = TRUE) wird jede Veränderung des Stackspeichers zusätzlich auf dem Bildschirm angezeigt.

Exportierte Konstanten:

5.1.1 Anzahl der vom Benutzer erzeugten Meldungen

Die Konstante 'MaxMeldung' legt die Anzahl der Fehlermeldungen oder Warnungen fest, die durch Aufruf der Prozeduren 'SchreibeMeldung' und 'MeldeFehler' in einem Analyseprogramm erzeugt werden dürfen. Beim Aufruf von 'SchreibeMeldung' oder 'MeldeFehler' können dem Parameter 'MeldungNr' Werte aus der Menge {0, ..., 'MaxMeldung'} übergeben werden. Der Wert von 'MaxMeldung' ist 100.

Exportierte Typen:

5.1.2 Typ für Nummern von Warnungen und Fehlermeldungen

Der Typ 'MeldungsNr' ist ein Unterbereichstyp von CARDINAL und umfaßt die Nummern von Warnungen und Fehlermeldungen, die vom System oder vom Benutzer vergeben werden dürfen.

5.1.3 Typ für Zeichenketten im Stackspeicher

Mit dem Typ 'StackStr' müssen Variablen deklariert werden, die der Prozedur 'PushStr' zur Speicherung auf dem Stack übergeben werden. Die Länge der Zeichenketten ist auf 81 Zeichen beschränkt.

Exportierte Variablen:

5.1.4 Anzahl der tolerierten Fehler

Die INTEGER-Variable 'FehlerLimit' gibt die Anzahl der Fehler an, die noch auftreten dürfen, bevor die Programmausführung abgebrochen wird. Zu Beginn wird sie mit dem Wert 10 initialisiert.

5.1.5 Tracing

Der Wert der Variablen 'TraceMode' vom Typ BOOLEAN legt fest, ob der Trace-Modus aktiv ist. Falls 'TraceMode' auf TRUE gesetzt ist, wird das Füllen und das Leeren des Stacks auf dem Bildschirm protokolliert, d.h. auf den Stack gebrachte bzw. vom Stack genommene Zeichenketten werden, gemäß der Stack-Höhe eingerückt und mit den Symbolen '=>' oder '<=' versehen, ausgegeben. Die Variable 'TraceLevel' gibt die Schachtelungstiefe von Prozeduraufrufen an, deren Namen mit 'PushStr' auf dem Stack abgelegt werden.

Exportierte Prozeduren:

5.1.6 Zeichenketten auf dem Stack ablegen

```
PROCEDURE PushStr(s: StackStr);
```

Die Zeichenkette 's' wird auf dem Trace-Stack abgelegt. Der Stack dient dazu, nach Auftreten eines Fehlers Überblick über die zuvor aufgerufenen Prozeduren in der Reihenfolge ihres Aufrufs zu erhalten. Die Anweisungsteile fast aller Prozeduren von VERENA enthalten als erste Anweisung den Aufruf 'PushStr(<Prozedurname>)' und als letzte Anweisung den Aufruf 'PopStr'. Zudem muß vor jeder 'RETURN'-Anweisung die Prozedur 'PopStr' aufgerufen werden, da der Prozedurkörper in diesem Fall nicht bis zum Ende abgearbeitet wird.

5.1.7 Zeichenketten vom Stack nehmen

```
PROCEDURE PopStr;
```

Die zuoberst auf dem Stack liegende Zeichenkette wird entfernt.

5.1.8 Meldung von Fehlern und Warnungen

Bei Aufruf von 'MeldeFehler' wird eine Fehlermeldung auf die Standardausgabedatei (normalerweise den Bildschirm) ausgegeben, die die vier Zeichenketten 'Meldung', 'ObjName1', 'ObjName2' und 'ObjName3' sowie die Nummer der Meldung enthält. Der Benutzer darf nur Nummern von 0 bis 'MaxMeldung' vergeben. Weiter erfolgt die Ausgabe des Trace-Stacks.

```
PROCEDURE MeldeFehler(Meldung  : ARRAY OF CHAR;
                      ObjName1 : ARRAY OF CHAR;
                      ObjName2 : ARRAY OF CHAR;
                      ObjName3 : ARRAY OF CHAR;
                      MeldungNr: MeldungsNr);
```

Die Anzahl der Aufrufe von 'MeldeFehler' ist durch den Wert von FehlerLimit begrenzt, da davon ausgegangen wird, daß nach einer gewissen Zahl von aufgetretenen Fehlern eine sinnvolle Fortführung des Programms nicht mehr gegeben ist (Abschnitt 5.1.4). Durch Aufrufe von 'SchreibeMeldung' hingegen ist es dem Benutzer möglich, eine beliebige Anzahl von Warnungen oder sonstigen Meldungen zu erzeugen.

```
PROCEDURE SchreibeMeldung(Meldung  : ARRAY OF CHAR;
                          ObjName1 : ARRAY OF CHAR;
                          ObjName2 : ARRAY OF CHAR;
                          ObjName3 : ARRAY OF CHAR;
                          MeldungNr: MeldungsNr);
```

Bei eingeschaltetem Trace-Modus (Variable 'TraceMode' = TRUE) wird eine Warnung auf den Bildschirm (Standardausgabedatei) ausgegeben, die die Zeichenketten 'Meldung', 'ObjName1', 'ObjName2' und 'ObjName3' sowie die Nummer der Warnung enthält. Zusätzlich wird der Trace-Stack angezeigt.

Bei Aufrufen der Prozeduren 'SchreibeMeldung' und 'MeldeFehler' sollen den Parametern 'ObjName1', 'ObjName2' und 'ObjName3' die Namen der beteiligten Zufallsvariablen übergeben werden. Falls weniger als drei Zufallsvariablen am Auftreten der Warnung beteiligt sind, sollte den nicht benötigten Parametern eine leere Zeichenkette übergeben werden. Die Zeichenketten dürfen die Länge null haben, jedoch nicht genau ein Zeichen enthalten, da sie sonst von einigen Compilern nicht als Zeichenketten, sondern als Literale vom Typ CHAR interpretiert und übersetzt werden. Der Compiler gibt dann zwar keine Fehlermeldung aus, jedoch tritt ein Laufzeitfehler auf (bei SUN-MODULA). Durch Vergabe verschiedener Nummern werden gleichlautende Meldungen unterscheidbar und können den jeweiligen Aufrufen zugeordnet werden.

5.1.9 Zusammenfassung aller Warnungen und Fehler

```
PROCEDURE FehlerZusammenfassung;
```

Die Prozedur bewirkt die Ausgabe eines Protokolls über die Häufigkeit der Ausführung der Prozeduren 'SchreibeWarnung' und 'MeldeFehler'. Sie sollte am Ende jedes Analyseprogramms aufgerufen werden.

5.2 Das Modul 'IOProc'

Das Modul 'IOProc' stellt eine einheitliche Schnittstelle zu den von den verschiedenen Compilern bereitgestellten Ein- und Ausgabeoperationen dar.

Exportierte Typen:

5.2.1 Typ für Textdateien

Mit dem Typ 'File' werden Textdateien deklariert, die später mittels 'FCreate' geöffnet und mit 'FClose' geschlossen werden können. Das Schreiben in die Textdateien erfolgt mit den Prozeduren 'FWrite', 'FWriteCard', 'FWriteString', 'FWriteInt', 'FWriteReal' und 'FWriteLn'.

Exportierte Prozeduren:

5.2.2 Umleiten der Ausgabe auf Dateien und Geräte

```
PROCEDURE OpenOutput(fname : ARRAY OF CHAR);
```

Die Standard-Ausgabe wird auf eine Datei mit dem Namen ’fname’ umgeleitet. ’fname’ kann unter einigen Betriebssystemen auch der logische Name eines Ausgabegerätes sein. Die Umlenkung bleibt solange erhalten, bis sie mit dem Aufruf der Prozedur ’CloseOutput’ aufgehoben wird.

5.2.3 Schließen einer Ausgabedatei

```
PROCEDURE CloseOutput;
```

Nach Umlenken der Standardausgabe mittels der Prozedur ’OpenOutput’ erfolgt die Ausgabe nach Aufruf dieser Prozedur wieder in die ursprüngliche Standard-Ausgabedatei (normalerweise der Bildschirm).

5.2.4 Ausgabe einzelner Zeichen

```
PROCEDURE Write(ch:CHAR);
```

Das in ’ch’ gespeicherte Zeichen wird in die Standard-Ausgabedatei geschrieben.

5.2.5 Ausgabe von ’CARDINAL’-Zahlen

```
PROCEDURE WriteCard(x,n:CARDINAL);
```

Der Wert von ’x’ vom Typ ’CARDINAL’ wird in die Standard-Ausgabedatei geschrieben. Die Länge der Ausgabe beträgt ’n’ Zeichen.

5.2.6 Ausgabe von Zeichenketten

```
PROCEDURE WriteString(s:ARRAY OF CHAR);
```

Die in der Variable ’s’ abgelegte Zeichenkette wird in die Standard-Ausgabedatei geschrieben.

5.2.7 Ausgabe von Zeichenketten mit Zeilenvorschub

```
PROCEDURE WriteStrLn(s:ARRAY OF CHAR);
```

Die in der Variablen 's' abgelegte Zeichenkette wird in die Standard-Ausgabedatei geschrieben. Zusätzlich wird ein Zeilenvorschub erzeugt.

5.2.8 Ausgabe von INTEGER-Zahlen

```
PROCEDURE WriteInt(x:INTEGER;n:CARDINAL);
```

Der Wert der Variablen 'x' vom Typ INTEGER wird in die Standard-Ausgabedatei geschrieben. Die Länge der Ausgabe beträgt 'n' Zeichen.

5.2.9 Ausgabe von 'Real'-Zahlen

```
PROCEDURE WriteReal(x:Real;n:CARDINAL);
```

Der Wert der Variablen 'x' vom Typ 'Real' wird in die Standard-Ausgabedatei geschrieben. Die Ausgabe erfolgt in Gleitpunktdarstellung mit einer Gesamtlänge von 'n' Zeichen.

5.2.10 Ausgabe von Zeilenvorschüben

```
PROCEDURE WriteLn;
```

Ein Zeilenvorschubzeichen wird in die Standard-Ausgabedatei geschrieben.

5.2.11 Erstellen von Ausgabedateien

```
PROCEDURE FCreate(VAR fdesc: Files.File;fname: ARRAY OF CHAR);
```

Es wird eine Textdatei namens 'fname' zum Schreiben geöffnet. Wenn eine Datei mit diesem Namen noch nicht existiert, so wird sie zunächst erzeugt. Mit dem Dateideskriptor 'fdesc' wird die Datei bei den nachfolgenden Ein-/Ausgabeoperationen angesprochen.

5.2.12 Schliessen von Ausgabedateien

```
PROCEDURE FClose(fdesc : Files.File);
```

Die Datei mit dem Dateideskriptor 'fdesc' wird geschlossen.

5.2.13 Einzelne Zeichen in Dateien schreiben

```
PROCEDURE FWrite(fdesc: Files.File; ch:CHAR);
```

Das in 'ch' gespeicherte Zeichen wird in die Datei mit dem Dateideskriptor 'fdesc' geschrieben.

5.2.14 'CARDINAL'-Zahlen in Dateien schreiben

```
PROCEDURE FWriteCard(fdesc: Files.File; x,n:CARDINAL);
```

Der Wert der Variablen 'x' vom Typ 'CARDINAL' wird in die Datei mit dem Dateideskriptor 'fdesc' geschrieben. Die Länge der Ausgabe beträgt 'n' Zeichen.

5.2.15 Zeichenketten in Dateien schreiben

```
PROCEDURE FWriteString(fdesc: Files.File; s:ARRAY OF CHAR);
```

Die in 's' gespeicherte Zeichenkette wird in die durch den Dateideskriptor 'fdesc' spezifizierte Datei geschrieben.

5.2.16 Zeichenketten mit Zeilenvorschub in Dateien schreiben

```
PROCEDURE FWriteStrLn(fdesc: Files.File; s:ARRAY OF CHAR);
```

Die in der Variablen 's' gespeicherte Zeichenkette wird in die Datei mit dem Dateideskriptor 'fdesc' geschrieben. Zusätzlich wird ein Zeilenvorschubzeichen erzeugt.

5.2.17 INTEGER-Zahlen in Dateien schreiben

```
PROCEDURE FWriteInt(fdesc: Files.File; x:INTEGER;n:CARDINAL);
```

Der Wert der Variablen 'x' vom Typ INTEGER wird in die Datei mit dem Dateideskriptor 'fdesc' geschrieben. Die Länge der Ausgabe beträgt 'n' Zeichen.

5.2.18 'Real'-Zahlen in Dateien schreiben

```
PROCEDURE FWriteReal(fdesc: Files.File; x:Real;n:CARDINAL);
```

Der Wert der Variablen 'x' vom Typ 'Real' wird in die Datei mit dem Dateideskriptor 'fdesc' geschrieben. Die Ausgabe erfolgt in Gleitpunktdarstellung mit einer Gesamtlänge von 'n' Zeichen.

5.2.19 Zeilenvorschübe in Dateien schreiben

```
PROCEDURE FWriteLn(fdesc: Files.File);
```

Ein Zeilenvorschubzeichen wird in die Datei mit dem Dateideskriptor 'fdesc' geschrieben.

5.3 Das Modul 'Math'

Da die Namen der Mathematik-Bibliotheken sowie der in ihnen enthaltenen mathematischen Funktionen nicht bei allen Compilern gleich sind und sich auch für unterschiedliche Definitionen des Typs 'Real' ändern, wird das Modul 'Math' als Schnittstelle zu diesen Funktionen verwendet. Einige der Prozeduren sind von Folgeänderungen betroffen, falls eine Änderung der Definition des Typs 'Real' im Modul 'Reals' erfolgt. Dabei handelt es sich grundsätzlich um alle Prozeduren, die mathematische Funktionen aus Standard-Mathematik-Bibliotheken importieren.

Exportierte Typen:

5.3.1 Typ für komplexe Zahlen

```
TYPE Komplex = RECORD
                 re,im : Real;
               END;
```

Der Typ 'Komplex' dient der Darstellung komplexer Zahlen. Die Komponente 're' gibt den Realteil an, 'im' den Imaginärteil.

5.3.2 Typ für Zugriffsobjekte für komplexe Zahlen

```
TYPE KZugriff = RECORD
                  CASE (**):(**)BOOLEAN OF
                    TRUE:  A : ADDRESS;
                  | FALSE: X : POINTER TO Komplex;
                  END; (* CASE *)
                END; (* RECORD *)
```

'KZugriff' ist ein unbenannter varianter Verbund. Auf die Bedeutung der '(**)' wurde bereits im Kapitel 4.3 bei der Beschreibung des Typs 'RZugriff' eingegangen.

Der Typ 'KZugriff' dient der Deklaration von Variablen, mit denen auf beliebig im Arbeitsspeicher plazierte Objekte vom Typ 'Komplex' zugegriffen werden kann. Hierzu wird der Komponente 'A' die Speicheradresse des Objektes zugewiesen. Mit Hilfe der Komponente 'X' erfolgt der Zugriff auf den Wert des Objektes. Dies verdeutlicht das folgende Programmstück:

```
MODULE ...
VAR c  : Komplex;
    zc : KZugriff;

BEGIN
  ...
  zc.A := Adresse;   (* Speicheradresse eines Objektes vom Typ 'Komplex' *)

  c.re := zc.X^.re;
  c.im := zc.X^.im;

  (* oder einfacher: *)

  c := zc.X^

END ... .
```

5.3.3 Typ für Zugriffsobjekte für INTEGER

```
TYPE IZugriff = RECORD                    (* Zugriff auf INTEGER-Zahlen *)
                 CASE (**):(**)BOOLEAN OF
                   TRUE:  A : ADDRESS;
                 | FALSE: X : POINTER TO INTEGER;
                 END; (* CASE *)
               END; (* RECORD *)
```

'IZugriff' ist ein unbenannter varianter Verbund. Der Typ 'IZugriff' dient zur Deklaration von Variablen, mit denen auf beliebig im Arbeitsspeicher plazierte Objekte vom Typ INTEGER zugegriffen werden kann. Hierzu wird der Komponente 'A' die Speicheradresse des Objektes zugewiesen. Mit Hilfe der Komponente 'X' erfolgt der Zugriff auf den Wert des Objektes. Auch hier zur Verdeutlichung ein Programmstück:

```
VAR i  : INTEGER;
    zi : IZugriff;

BEGIN
  ...
  zi.A := Adresse;   (* Speicheradresse eines Objektes vom Typ INTEGER *)
  i    := zi.X^;
  ...
END ... .
```

Für den Typ 'Real' sind entsprechende Zugriffsobjekte definiert (siehe 5.5.2). Der Benutzer wird die verschiedenen Zugriffsobjekte in der Regel nicht benötigen. Benutzt werden diese Objekte für den Zugriff auf dynamisch erzeugte Variablen. Dies ist beispielsweise der Fall, wenn Restriktionen für die Größe statischer Speicherbereiche umgangen werden sollen.

Exportierte Variablen:

5.3.4 Speicherbedarf für komplexe Zahlen

Die Variable 'KomplexSize' vom Typ 'ADDRESS' enthält die Anzahl der Speichereinheiten, die die Darstellung einer Variablen vom Typ 'Komplex' benötigt.

5.3.5 Speicherbedarf für INTEGER

Die Variable 'IntSize' vom Typ 'ADDRESS' enthält die Anzahl der Speichereinheiten, die die Darstellung einer Variablen vom Typ INTEGER benötigt.

5.3.6 Die Konstante 'π'

Die Konstante 'pi' ist vom Typ 'Real' und enthält einen Näherungswert für die Konstante π.

Exportierte Prozeduren:

5.3.7 Exponentialfunktion

```
exp(x: Real): Real;
```

Berechnung der Exponentialfunktion e^x.

5.3.8 Natürlicher Logarithmus

```
PROCEDURE ln(x: Real): Real;
```

Berechnung des natürlichen Logarithmus $\ln x$.

5.3.9 Sinus

```
PROCEDURE sin(x: Real): Real;
```

Berechnung des Sinus von x, wobei x in der Einheit 'rad' anzugeben ist.

5.3.10 Cosinus

```
PROCEDURE cos(x: Real): Real;
```

Berechnung des Cosinus von x. x ist in der Einheit 'rad' anzugeben.

5.3.11 Arcus Tangens

`PROCEDURE arctan(x: Real): Real;`

Berechnung des Arcus Tangens von x in der Einheit 'rad'.

5.3.12 Maximum zweier INTEGER-Werte

`PROCEDURE imax(i1,i2: INTEGER): INTEGER;`

Berechnung des Maximums der beiden INTEGER-Parameter 'i1' und 'i2'.

5.3.13 Minimum zweier INTEGER-Werte

`PROCEDURE imin(i1,i2: INTEGER): INTEGER;`

Berechnung des Minimums der beiden INTEGER-Parameter 'i1' und 'i2'.

5.3.14 Maximum zweier 'Real'-Werte

`PROCEDURE rmax(x1,x2: Real): Real;`

Berechnung des Maximums der beiden Real-Parameter 'x1' und 'x2'.

5.3.15 Minimum zweier 'Real'-Werte

`PROCEDURE rmin(x1,x2: Real): Real;`

Berechnung des Minimums der beiden Real-Parameter 'x1' und 'x2'.

5.3.16 Runden auf das Vielfache einer Zahl

`PROCEDURE Snap(x,d: Real): Real;`

Die Prozedur berechnet das zu x nächstgelegene Vielfache von d:

$$f(x) := d\lfloor\frac{x}{d} + \frac{1}{2}\rfloor$$

5.3.17 Aufrunden auf das Vielfache einer Zahl

```
PROCEDURE SnapUp(x,d: Real): Real;
```

'SnapUp' ermittelt das kleinste Vielfache von d, das größer als x ist:

$$f(x) := d\lceil \frac{x}{d} \rceil$$

5.3.18 Abrunden auf das Vielfache einer Zahl

```
PROCEDURE SnapDown(x,d: Real): Real;
```

Es wird das größte Vielfache von d errechnet, das kleiner als x ist:

$$f(x) := d\lfloor \frac{x}{d} \rfloor$$

5.3.19 Umwandlung von 'Real' nach INTEGER durch Runden

```
PROCEDURE Int(x: Real): INTEGER;
```

Der Wert des Real-Parameters 'x' wird durch Runden in eine INTEGER-Zahl umgewandelt:

$$\mathrm{Int}(x) := \lfloor x + \frac{1}{2} \rfloor$$

5.3.20 Logarithmus zur Basis zwei für INTEGER

```
PROCEDURE LogZwei(i: INTEGER): INTEGER;
```

Die Funktion berechnet die Anzahl der Binärstellen, die zur Darstellung der Zahl i benötigt werden.

5.3.21 Potenzen zur Basis 2

```
PROCEDURE ZweiHoch(i: INTEGER): INTEGER;
```

Die Funktion berechnet 2^i.

5.3.22 Berechnung von DME-1 aus Momenten

Bei dieser und der folgenden Prozedur bezeichnet „M“ die Darstellung mit Momenten, „Exp“ die Repräsentation durch Parameter der Dichte einer Exponentialverteilung.

```
PROCEDURE KonvertMExp(M0,M1,a,sign: Real; VAR c,lambda: Real);
```

Sei $f_A(x) = ce^{-sign\lambda x}$ das Teilstück einer Dichtefunktion über dem Intervall A mit

$$A = \begin{cases} (-\infty, a) & \text{, wenn sign} = -1 \\ (a, \infty) & \text{, wenn sign} = 1 \end{cases}$$

Die Prozedur 'KonvertMExp' berechnet aus den gegebenen Momenten 'M0' := $\int_A f(x)dx$ und 'M1' := $\int_A xf(x)dx$ die Werte von c und λ (= 'lambda'). Benutzt wird diese Prozedur – und auch die Prozedur 'KonvertMExp' – immer dort, wo mit Dichten maximaler Entropie der Ordnung 1 (DME-1) operiert wird (z.B. bei der Addition von Zufallsvariablen durch Faltung ihrer Verteilungsfunktionen). Die DME-1-Bereiche einer Dichtefunktion, die mit 101-Dichten approximiert wurde, werden intern durch die Werte von 'Wlinks' und 'Mlinks' bzw. 'Wrechts' und 'Mrechts' repräsentiert. 'Wlinks', 'Mlinks', 'Wrechts' und ‘Mrechts' sind Komponenten des Verbundes 'ZVStrukt', welcher eine Zufallsvariable beschreibt. Bei einer Faltung von Verteilungen von Zufallsvariablen werden aus diesen Größen die Parameter 'c' und 'λ' der DME-1 durch 'KonvertMExp' berechnet. Die Bestimmung der Parameter 'c' ind 'λ' ist erforderlich, da die DME-1-Bereiche bei der Faltung gegebenenfalls aufgeteilt werden müssen. Nach Ausführung der Faltung, müssen aus den Parametern 'c' und 'λ' der DME-1-Anteile die Werte 'Wlinks' und 'Mlinks' bzw. 'Wrechts' und 'Mrechts' für die interne Darstellung der Verteilung berechnet werden; dies geschieht mit der Prozedur 'KonvertExpM'.

Bei der Faltung von Verteilungsfunktionen werden die Konversionen vom System übernommen. Wenn der Benutzer explizit mit 101-Dichten arbeitet, so muß er diese beiden Prozeduren benutzen, um von der internen Darstellung in 101-Dichten und umgekehrt zu konvertieren.

5.3.23 Berechnung von Momenten aus DME-1

```
PROCEDURE KonvertExpM(c,lambda,a,d,sign: Real; VAR M0,M1: Real);
```

Sei $f_A(x) = ce^{-sign\lambda x}$ das Teilstück einer Dichtefunktion über dem Intervall A, wobei A wie in Abschnitt 5.3.22 definiert sei. Die Prozedur 'KonvertExpM' berechnet aus gegebenem c und λ die Momente 'M0' := $\int_A f(x)dx$ und 'M1' := $\int_A xf(x)dx$ (siehe auch Abschnitt 5.3.22).

5.3.24 Potenzen von Real-Zahlen

```
PROCEDURE PHochN(p: Real; n: INTEGER): Real;
```

Berechnung des Werts p^n.

5.3.25 Berechnung von VME-1 aus Momenten

Die Aufgaben der in diesem und dem nächsten Abschnitt beschriebenen Operationen auf diskreten 101-Verteilungen entsprechen denen von 'KonvertMExp' und 'KonvertExpM'.

```
PROCEDURE KonvertMGeometr(M0,M1:Real; a,sign:INTEGER; VAR c,p:Real);
```

Sei $p_A(n) = c \cdot p \cdot q^{sign \cdot n - 1}$, $q = 1 - p$, das Teilstück einer Wahrscheinlichkeitsfunktion über dem Intervall A mit

$$A = \begin{cases} (-\infty, a) & \text{, wenn sign} = -1 \\ [a+1, \infty) & \text{, wenn sign} = 1 \end{cases}$$

Die Prozedur 'KonvertMGeometr' berechnet aus den gegebenen Momenten 'M0' und 'M1' mit

$$\begin{aligned} M^0 &:= \sum_{n \in A} p_A(n) \\ M^1 &:= \sum_{n \in A} n \cdot p_A(n) \end{aligned}$$

die Werte von c und p.

Die mit einer VME-1 approximierten Bereiche einer Wahrscheinlichkeitsfunktion werden intern durch die Werte von 'M0links' und 'M1links' bzw. 'M0rechts' und 'M1rechts' repräsentiert. Diese sind Komponenten des varianten Verbundes 'ZVStrukt' und sind nur für diskret unendliche Zufallsvariablen sichtbar. Beim Rechnen mit den Wahrscheinlichkeiten für diese Bereiche, also z.B. bei der Berechnung der Wahrscheinlichkeitsmasse für einen Punkt oder über einem Intervall, müssen die Werte der Parameter c und p bekannt sein. Diese Parameter lassen sich mit der Prozedur 'KonvertMGeometr' ermitteln. Umgekehrt ist nach einer Änderung der Parameter c und p die Berechnung von 'M0links' und 'M1links' bzw. 'M0rechts' und 'M1rechts' notwendig, da nur diese Werte bei der internen Speicherung der VME-1-Abschnitte von Zufallsvariablen benutzt werden. Dieser Berechnung dient die Prozedur 'KonvertGeometrM'.

5.3.26 Berechnung von Momenten aus VME-1

```
PROCEDURE KonvertGeometrM(c,p: Real; a,sign: INTEGER; VAR M0,M1: Real);
```

Sei $p_A(n) = c \cdot p \cdot q^{sign \cdot n - 1}$, $q = 1 - p$, das Teilstück einer Wahrscheinlichkeitsfunktion über dem Intervall A, wobei A wie oben definiert sei. Die Prozedur 'KonvertGeometrM' berechnet aus c und p die Momente 'M0' und 'M1':

$$\begin{aligned} M^0 &:= \sum_{n \in A} p_A(n) \\ M^1 &:= \sum_{n \in A} n \cdot p_A(n) \end{aligned}$$

(siehe auch Abschnitt 5.3.25)

5.3.27 Addition komplexer Zahlen

```
PROCEDURE KAddition(c1,c2: Komplex; VAR c3: Komplex);
```

Berechnung der Summe zweier komplexer Zahlen.

5.3.28 Multiplikation komplexer Zahlen

```
PROCEDURE KMultipl(c1,c2: Komplex; VAR c3: Komplex);
```

Berechnung des Produktes zweier komplexer Zahlen.

5.3.29 Transformation kartesischer Koordinaten in Polarkoordinaten

```
PROCEDURE KPolar(re,im: Real; VAR r,gamma: Real);
```

Aus gegebenem Realteil 're' und Imaginärteil 'im' einer komplexen Zahl c werden deren Polarkoordinaten 'r' und 'gamma' berechnet mit $c = r(\cos\gamma + i\sin\gamma)$.

5.3.30 Transformation von Polar- in kartesische Koordinaten

```
PROCEDURE KKartesisch(r,gamma: Real; VAR re,im: Real);
```

Die Funktion berechnet aus der mit 'r' und 'gamma' gegebenen Polardarstellung einer komplexen Zahl deren Realteil 're' und Imaginärteil 'im'.

5.3.31 Potenzierung komplexer Zahlen

```
PROCEDURE KPotenz(c1: Komplex; j: INTEGER; VAR c2: Komplex);
```

'c2' wird berechnet als 'j'-te Potenz der komplexen Zahl 'c1'.

5.4 Das Modul 'Misc'

Das Modul 'Misc' ist für Prozeduren vorgesehen, die sich anhand ihrer Aufgabenstellung nicht in eines der übrigen Module einordnen lassen (Miscellaneous).

Exportierte Prozeduren:

5.4.1 Zusammenfügen von Zeichenketten

```
PROCEDURE Konkat(S1,S2 : ARRAY OF CHAR; VAR S3 : ARRAY OF CHAR);
```

Die Zeichenkette 'S3' entsteht durch Anhängen von 'S2' an 'S1'.

5.5 Das Modul 'Reals'

Vom Modul 'Reals' wird der Typ 'Real' bereitgestellt. Mit der Definition von 'Real' wird die verwendete Fließkomma-Darstellung festgelegt (vgl. Abschnitt 4.5). Von einer Veränderung der Definition sind auch die Konvertierungs-Prozeduren 'R' und 'RK', sowie das Implementationsmodul von 'Math' betroffen. Hierauf wird in den jeweiligen Abschnitten eingegangen.

Exportierte Typen:

5.5.1 Typ 'Real' für reelle Zahlen

Der Typ 'Real' wird im gesamten Programmsystem generell als Datentyp für die Darstellung von reellen Zahlen benutzt. Er ist, je nach Genauigkeitsanforderungen und den Möglichkeiten, die vom Compiler angeboten werden, als einfach- oder doppelt-genaue REAL-Zahl definiert.

5.5.2 Typ für Zugriffsobjekte für 'Real'-Zahlen

```
TYPE RZugriff = RECORD
                  CASE (**):(**)BOOLEAN OF
                    TRUE:  Adr : ADDRESS;
                  | FALSE: Elem : POINTER TO Real;
                  END; (* CASE *)
                END; (* RECORD *)
```

Der Typ 'RZugriff' dient zur Deklaration von Variablen, mit denen auf beliebig im Arbeitsspeicher plazierte Objekte vom Typ 'Real' zugegriffen werden kann. Hierzu wird der Komponente 'Adr' die Speicheradresse des Objektes zugewiesen. Mit Hilfe der Komponente 'Elem' erfolgt der Zugriff auf den Wert des Objektes:

```
VAR r  : Real;
    zr : RZugriff;

BEGIN
  ...
  zr.Adr := Adresse;   (* Speicheradresse eines Objektes vom Typ 'Real' *)

  r := zr.Elem^;

END ... .
```

Exportierte Variablen:

5.5.3 Speicherbedarf für 'Real'-Zahlen

Die Variable 'RealSize' vom Typ 'ADDRESS' gibt die Anzahl der Speichereinheiten an, die Variablen vom Typ 'Real' belegen. Sie wird zur Berechnung der Größe von Speicherbereichen herangezogen, die beim Betriebssystem für die Erzeugung dynamischer Datenstrukturen angefordert werden müssen.

5.5.4 Häufig verwendete 'Real'-Konstanten

'RK0', 'RK1', 'RK2' und 'RK10' sind Variablen vom Typ 'Real', deren Werte die häufig benötigten Konstanten 0.0, 1.0, 2.0 und 10.0 sind. Sie können an Stelle der Ausdrücke 'RK(0.0)', 'RK(1.0)', usw. benutzt werden. Zur Verwendung der Funktion 'RK' siehe auch Abschnitt 5.5.6.

Exportierte Prozeduren:

5.5.5 Umwandlung von 'Real' in REAL

```
PROCEDURE R(r: Real): REAL;
```

Mit dieser Prozedur kann der Wert einer Variablen vom Typ 'Real' einer Variablen vom Typ REAL zugewiesen werden.

5.5.6 Umwandlung von REAL in 'Real'

```
PROCEDURE RK(r: REAL): Real;
```

oder

```
PROCEDURE RK(r: Real): Real;
```

Mit Hilfe dieser Prozedur kann der Wert einer Variablen oder einer Konstanten vom Typ REAL einer Variablen vom Typ 'Real' zugewiesen werden. Die beiden Varianten der Prozedurdeklaration sind im Definitionsmodul 'Reals.def' erläutert. Werden in einem Analyseprogramm Fließkommazahlen der Typen REAL und 'Real' benutzt, kann es je nach benutztem Compiler Probleme bei der Konvertierung geben, wie das nachfolgende Beispiel zeigt. Generell wird daher geraten, nur den Typ 'Real' zu benutzen.

Das Programmstück

```
CONST c = 0.21;
VAR   lr1,lr2 : LONGREAL;
      r : REAL;
BEGIN
  r := c;
  lr1 := r;
  lr2 := c;
  WriteLongReal(lr1,20);
  WriteLn;
  WriteLongReal(lr2,20);
  WriteLn;
END.
```

erzeugt (bei Verwendung des SUN-MODULA-Compilers) die Ausgabe:

```
2.099999934434891E-1
2.100000000000000E-1
```

Da einige Compiler Real-Konstanten wie z. B. „2.0“ als vom Typ REAL betrachten, sollten diese Konstanten grundsätzlich mit der Prozedur 'RK' konvertiert werden.

5.6 Das Modul 'Tupel'

Bei der Initialisierung von Zufallsvariablen mit endlichem, mehrdimensionalem diskreten Wertebereich werden Tupel von INTEGER– und 'Real'-Zahlen benötigt, die von dem Modul 'Tupel' neben den dazugehörigen Operationen bereitgestellt werden.

Exportierte Konstanten:

5.6.1 Maximale Anzahl von Elementen eines Tupels

Mit der Konstanten 'MaxDim' ist die maximale Anzahl von Elementen eines Tupels festgelegt. Somit ist auch der maximale Wert für die Dimension einer Zufallsvariablen festgelegt, da der Datentyp 'Tupel' für die Darstellung von Elementen aus dem Wertebereich mehrdimensionaler Zufallsvariablen verwendet wird. Die Konstante 'MaxDim' ist in dem Definitionsmodul 'Tupel.def' definiert und hat den Wert '10'.

Exportierte Typen:

5.6.2 Typ für INTEGER-Tupel

Der Typ 'ITupel' dient der Darstellung von Tupeln mit Elementen vom Typ INTEGER. Sei die Variable 't' vom Typ 'ITupel'. Dann gibt 't[0]' die aktuelle Anzahl der Elemente des Tupels, und 't[i]' den Wert des i-ten Elementes an.

5.6.3 Typ für 'Real'-Tupel

Objekte vom Typ 'RTupel' werden bei der Zuweisung von Werten an mehrdimensionale diskrete Verteilungen und als Rückgabeparameter der Prozedur 'Erwartung' zur Berechnung ihrer Erwartungswerte benötigt.

```
TYPE RTupel   = RECORD
  dim: INTEGER;
  val: ARRAY[1..MaxDim] OF Real;
                END;
```

Beim Typ 'RTupel' handelt es sich um einen Verbundtyp. Seine Komponente 'dim' gibt die aktuelle Anzahl von Elementen des Tupels an, im Feld 'val' werden die Werte der Elemente gespeichert. Die Konstante 'MaxDim' ist im Definitionsmodul 'Tupel.def' definiert und hat den Wert '10'.

5.6.4 Erzeugung von INTEGER-Tupeln

Die Prozedur 'BildeITupel' ist 'MaxDim'-fach implementiert, da die Anzahl der formalen Parameter der Prozeduren für verschiedene n unterschiedlich ist.

Beispiel: Prozedur-Deklaration für n = 5:

```
PROCEDURE BildeITupel5  (VAR IT: ITupel; I1,I2,I3,I4,I5: INTEGER);
```

Mit den Prozeduren 'BildeITupel$n$' werden INTEGER-Tupel mit n Elementen initialisiert. Die Werte der Elemente werden mit I1, I2, ..., In festgelegt. Das Tupel wird in der Variablen IT zurückgegeben.

5.6.5 Vergleichsoperation für INTEGER-Tupel

```
PROCEDURE VglITupel(IT1, IT2: ITupel): INTEGER;
```

Die Tupel 'IT1' und 'IT2' werden elementweise miteinander verglichen. Der Rückgabewert ist wie folgt festgelegt:

$$\text{VglITupel[IT1,IT2]} := \begin{cases} -1 & \text{wenn IT1[i]} \leq \text{IT2[i]} \quad \text{für alle i} \\ & \text{und IT1[i]} < \text{IT2[i]} \quad \text{für wenigstens ein i} \\ 0 & \text{wenn IT1[i]} = \text{IT2[i]} \quad \text{für alle i} \\ 1 & \text{wenn IT1[i]} \geq \text{IT2[i]} \quad \text{für alle i} \\ & \text{und IT1[i]} > \text{IT2[i]} \quad \text{für wenigstens ein i} \\ 2 & \text{wenn die Anzahl der Elemente von IT1 und IT2} \\ & \text{nicht übereinstimmt.} \\ 3 & \text{sonst} \end{cases}$$

5.6.6 Vergleichsoperation „größer oder gleich“ für INTEGER-Tupel

```
PROCEDURE ITupelGrGl(IT1, IT2: ITupel): BOOLEAN;
```

Die Funktion 'ITupelGrGl' ergibt genau dann den Wert TRUE, wenn die Funktion 'VglITupel' (Abschnitt 5.6.5), aufgerufen mit den Parametern 'IT1' und 'IT2', den Wert '0' oder den Wert '1' zurückgibt.

5.6.7 Vergleichsoperation „größer als“ für INTEGER-Tupel

```
PROCEDURE ITupelGr(IT1, IT2: ITupel): BOOLEAN;
```

Die Funktion 'ITupelGr' ergibt genau dann den Wert TRUE, wenn die Funktion 'VglITupel' (Abschnitt 5.6.5), aufgerufen mit den Parametern 'IT1' und 'IT2', den Wert '1' zurückgibt.

5.6.8 Vergleichsoperation „kleiner oder gleich“ für INTEGER-Tupel

```
PROCEDURE ITupelKlGl(IT1, IT2: ITupel): BOOLEAN;
```

Die Funktion ITupelKlGl' liefert genau dann den Wert TRUE, wenn die Funktion 'VglITupel' (Abschnitt 5.6.5), aufgerufen mit den Parametern 'IT1' und 'IT2', den Wert '-1' oder den Wert '0' zurückgibt.

5.6.9 Vergleichsoperation „kleiner als“ für INTEGER-Tupel

```
PROCEDURE ITupelKl(IT1, IT2: ITupel): BOOLEAN;
```

Die Funktion 'ITupelKl' ergibt genau dann den Wert TRUE, wenn die Funktion 'VglITupel' (Abschnitt 5.6.5), aufgerufen mit den Parametern 'IT1' und 'IT2', den Wert '-1' zurückgibt.

5.6.10 Vergleichsoperation „gleich“ für INTEGER-Tupel

```
PROCEDURE ITupelGl(IT1, IT2: ITupel): BOOLEAN;
```

Die Funktion 'ITupelGl' ergibt genau dann den Wert TRUE, wenn die Funktion 'VglITupel' (Abschnitt 5.6.5), aufgerufen mit den Parametern 'IT1' und 'IT2', den Wert '0' zurückgibt.

5.6.11 Addition von INTEGER-Tupeln

```
PROCEDURE AddITupel(IT1, IT2: ITupel; VAR RT: ITupel);
```

Die Prozedur 'AddITupel' betrachtet 'IT1', 'IT2' und 'RT' als Vektoren. 'RT' wird gemäß der Vektoraddition aus 'IT1' und 'IT2' berechnet.

5.6.12 Subtraktion von INTEGER-Tupeln

```
PROCEDURE SubITupel(IT1, IT2: ITupel; VAR RT: ITupel);
```

Die Prozedur 'SubITupel' betrachtet 'IT1', 'IT2' und 'RT' als Vektoren. 'RT' wird gemäß der Vektorsubtraktion aus 'IT1' und 'IT2' berechnet.

5.7 Das Modul 'RVBasics': Darstellung und Initialisierung von Zufallsvariablen

Das Modul 'RVBasics' exportiert die Datentypen 'ZufallsVar' und 'ZVStrukt' für die Darstellung von Zufallsvariablen, und es werden Prozeduren für die Initialisierung von Zufallsvariablen bereitgestellt. Auf den Aufbau der Datenstrukturen für Zufallsvariablen wird in Abschnitt 4.2 genauer eingegangen. Weiterhin werden Prozeduren für die dynamische Erzeugung von Zufallsvariablen angeboten.

Exportierte Typen:

5.7.1 Typ zur Auswahl einer Variante der Beschreibungsstruktur von Zufallsvariablen

```
TYPE ZVTyp = (DiskrEndl, Stet101, Gemischt, Diskr101);
```

Die Elemente des Aufzählungstyps 'ZVTyp' bezeichnen die unterschiedlichen Möglichkeiten, die in der Programmbibliothek zur Approximation einer Verteilung mittels Verteilungen maximaler Entropie eingesetzt werden. Ein Objekt vom Typ 'ZVTyp' wird benötigt, um die Variante eines Verbundes vom Typ 'ZVStrukt' festzulegen.

5.7.2 Typ für Beschreibungsstrukturen von Zufallsvariablen

```
TYPE ZVStrukt = RECORD

                  Erkennung: INTEGER;     (* Zur Erkennung von korrekt
                                             initialisierten Zufallsvari-
                                             ablen                       *)
                  Status   : ZVStatusSet; (* Zeigt an, ob die Zufalls-
                                             variable statisch oder
                                             dynamisch erzeugt wurde,
                                             und ob die Verteilung seit
                                             dem letzten Nullsetzen
                                             veraendert wurde            *)
                  Name     : ZVName;      (* Benutzerdefinierter Name der
                                             Zufallsvariablen            *)
                  d        : INTEGER;     (* Dimension der Verteilung    *)

             CASE Typ : ZVTyp OF


               DiskrEndl .. Gemischt :

                  (* Variablen fuer diskrete Verteilungen mit endlichem
                     Wertebereich                                        *)

                  aDskr,bDskr,
                  rDskr: ITupel;      (* Beschreibung des Wertebereichs  *)
                  VertDskr: ADDRESS;  (* Adresse des Arrays zur Speicherung
                                         der diskreten Verteilung. Falls
                                         dieser nicht vorhanden ist, so ist
                                         VertDskr = NIL                  *)

                  (* Variablen fuer stetige Verteilungen                 *)
                  delta: Real;
                  aStetDef,bStetDef,
                  rStetDef       : INTEGER;(* Beschreibung des maximalen
                                              Bereiches fuer DME der
                                              Ordnung 0                  *)
                  aStet,bStet,
                  rStet          : INTEGER;(* Benutzter Bereich          *)
                  Wlinks, Wrechts: Real;   (* Wahrscheinlichkeiten
                                              P{X < a*delta} und
                                              P{X >= (b+1)*delta}        *)
                  Mlinks, Mrechts: Real;   (* 1. Momente                 *)
                  VertStet       : ADDRESS;(* Adresse des Arrays fuer die
                                              P{i*delta <=X< (i+1)*delta} *)


             | Diskr101 :

                  (* Variablen fuer diskrete Verteilung mit unendlichem
                     Wertebereich                                        *)
                  aDiskrDef,bDiskrDef,
                  rDiskrDef       : INTEGER;(* Beschreibung des maximalen
                                               Bereichs fuer VME der
                                               Ordnung 0 *)
```

5.7.3 Typ für Zufallsvariablen

Der Typ 'ZufallsVar' ist als Zeiger auf ein Objekt vom Typ 'ZVStrukt' definiert. Er wird für die Deklaration aller Variablen benutzt, die Zufallsvariablen repräsentieren.

Exportierte Prozeduren:

5.7.4 Überprüfung der korrekten Initialisierung

```
PROCEDURE Existent(X : ZufallsVar): BOOLEAN;
```

Die Funktion gibt TRUE als Resultat zurück, wenn **jede** der folgenden Bedingungen erfüllt ist:

1. Die Variable 'X' vom Zeigertyp 'ZufallsVar' besitzt einen Wert ungleich 'NIL'
2. Enthält 'X↑.Erkennung' den Wert 'Kennung', so ist 'x' initialisiert und darf benutzt werden.
3. Wenigstens eine der Verbundtypkomponenten 'X↑.VertDskr' und 'X↑.VertStet' ist von 'NIL' verschieden, falls 'X↑.Typ den Wert 'DiskrEndl', 'Stet101' oder 'Gemischt' besitzt. Falls 'X↑.Typ den Wert 'Diskr101' hat, ist 'NIL' für die Verbundtypkomponente 'X↑.VertDiskr' zulässig, wenn 'X↑.aDiskrDef' = 'X↑.aDiskr' = 'X↑.bDiskrDef' + 1 = 'X↑.bDiskr' + 1. Dies ist zum Beispiel sinnvoll, wenn eine geometrische Verteilung dargestellt werden soll.

Während der Laufzeit sollen durch Prüfung der drei Bedingungen fehlerhaft initialisierte oder nicht initialisierte Zufallsvariablen entdeckt werden.

5.7.5 Nullsetzen von Verteilungen

```
PROCEDURE Setze0Vert(X : ZufallsVar);
```

Alle Wahrscheinlichkeiten der Verteilung der Zufallsvariable X werden auf den Wert '0.0' gesetzt. Bei Zufallsvariablen mit stetigem Wertebereich werden außerdem die Komponenten 'Mrechts' und 'Mlinks', bei Zufallsvariablen mit diskret unendlichem Wertebereich die Komponenten 'M0rechts' und 'M0links', auf den Wert '0.0' gesetzt.

Eine explizite Nullsetzung von Verteilungen ist nur notwendig vor dem Aufruf von Prozeduren, die diese nicht selbständig vornehmen. Es sind dies die Prozeduren 'BesetzTWbereichKonst', 'BesetzTWbereichPMF', 'DME1LAufteilen', 'DME1RAufteilen' und 'SummWGewichtet' aus dem Modul 'RndVar0'. Diesen Prozeduren ist gemeinsam, daß sie sowohl mit nullgesetzten als auch mit nicht nullgesetzten Zielverteilungen sinnvoll angewendet werden können. Ob die Nullsetzung der Zielverteilung erfolgen muß, ergibt sich aus dem jeweiligen Programmkontext.

Wenn die Zufallsvariable sofort nach der Nullsetzung ihrer Verteilung einer Prozedur als Parameter übergeben wird, so kann statt der Prozedur 'Setze0Vert' auch die folgende Funktion 'S0V' verwendet werden.

5.7.6 Nullsetzen von Verteilungen innerhalb von Ausdrücken

```
PROCEDURE S0V(X: ZufallsVar): ZufallsVar;
```

Die Funktion 'S0V' hat dieselbe Wirkung wie die Prozedur 'Setze0Vert', d.h. die Wahrscheinlichkeiten und Momente der Zufallsvariablen X werden mit dem Wert '0.0' initialisiert. Als Resultat der Funktion wird der Wert der Variablen 'X', also eine Adresse, zurückgegeben. Diese Funktion wurde in die Programmbibliothek aufgenommen, um alle Komponenten der Verteilung einer Zufallsvariablen, die als aktueller Parameter einer Prozedur auftritt, mit dem Wert '0.0' zu belegen. Auf diese Weise tritt dieser rein technische Vorgang nicht als eigenständige Anweisung auf. Dies trägt zu einer größeren textuellen Entsprechung von Analyseprogramm und -algorithmus bei.

Beispiel:

Sei 'OperationAufX' eine Prozedur, die die Verteilung einer als Parameter übergebenen Zufallsvariable 'X' in irgendeiner Weise verändert, dabei aber davon ausgeht, daß die Verteilung bereits zuvor nullgesetzt wurde.

Dann ist das Programmstück:

```
Setze0Vert(X);
OperationAufX(X);
```

äquivalent zu der Programmzeile:

```
OperationAufX(S0V(X));
```

5.7.7 Initialisierung diskreter, eindimensionaler Zufallsvariablen mit endlichem Wertebereich

```
PROCEDURE InitDskrZV1D(VAR X: ZufallsVar; Bezeichnung: ZVName;
                       Vert: ADDRESS; a,b: INTEGER);
```

Die Prozedur initialisiert die Zufallsvariable X mit eindimensionalem diskreten Wertebereich. Dabei wird die Verteilung von X mittels der Prozedur 'Setze0Vert' nullgesetzt. Eine einmalige Initialisierung ist für alle statisch erzeugten Zufallsvariablen notwendig, bevor Operationen auf ihnen ausgeführt werden können. Bei mit 'GenZV' oder 'DynZV' dynamisch angelegten Zufallsvariablen ist kein Aufruf einer der 'InitxxZVxx'-Prozeduren nötig.

Die Variable 'X' repräsentiert die Zufallsvariable im weiteren Programmverlauf. Über den Parameter 'Bezeichnung' kann der Benutzer der Zufallsvariablen einen maximal 15 Zeichen langen Namen geben. Der aktuelle Parameter darf allerdings kein Zeichenkettenliteral aus nur einem Zeichen sein, da diese z.B. vom Sun-MODULA-Compiler als Literal vom Typ CHAR interpretiert werden. Soll der Name trotzdem nur ein Zeichen lang sein, so ist an dieses Zeichen mindestens noch ein Leerzeichen anzuhängen.

Der Parameter 'Vert' erhält die Speicheradresse eines Feldes (oder auch einer Real-Zahl, wenn die Mächtigkeit des Wertebereichs 1 beträgt), das die Verteilung der Zufallsvariable aufnehmen soll. Diese Adresse wird durch den Benutzer mit der Funktion 'ADR()' ermittelt.

Die Parameter 'a' und 'b' legen gemäß den Ausführungen in Abschnitt 4.2 den Wertebereich auf die Menge $[a : b] \subset \mathbb{Z}$ fest. Die Intervallgrenzen a und b sollten in der Regel mit den Grenzen des Unterbereichs übereinstimmen, mit dem das Verteilungsfeld deklariert wurde. Zu den Fällen, in denen Abweichungen sinnvoll sind, siehe Abschnitt 4.2.2. Falls jedoch auf Einzelwahrscheinlichkeiten von Verteilungen auch mittels Feldindizierung zugegriffen werden soll, so darf es keine Abweichung bei der Untergrenze 'a' geben, da sonst Verschiebungen des Indexbereiches auftreten.

Von 'InitDskrZV1D' werden im einzelnen die folgenden Arbeitsschritte ausgeführt:

- Anforderung von Arbeitsspeicher beim Laufzeitsystem zum Anlegen einer Datenstruktur vom Typ 'ZVStrukt'.
- Initialisierung der Komponenten dieser Datenstruktur. Insbesondere wird der Komponente 'ZVStrukt.VertDskr' der Wert des Parameters 'Vert' und der Komponente 'ZVStrukt.VertStet' der Wert 'NIL' zugewiesen. Der Diskriminante 'Typ' des varianten Verbundes 'ZVStrukt' wird der Wert 'DiskrEndl' zugewiesen; dadurch wird die erste Variante des Verbundes ausgewählt.
- Das durch die Parameter 'Vert', 'a' und 'b' festgelegte Verteilungsfeld wird nullgesetzt, d.h. alle Elemente des Feldes werden mit dem Wert '0.0' vorbesetzt.
- Der Komponente 'Erkennung' wird der Wert 'Kennung' zugewiesen.
- Wenn alle vorhergehenden Schritte erfolgreich durchgeführt werden konnten, wird dem Parameter 'X' die Adresse der erzeugten Datenstruktur zugewiesen.

5.7.8 Initialisierung diskreter, mehrdimensionaler Zufallsvariablen mit endlichem Wertebereich

```
PROCEDURE InitDskrZVMD(VAR X: ZufallsVar; Bezeichnung: ZVName;
                       Vert: ADDRESS; a,b: ITupel);
```

Die Prozedur 'InitDskrZVMD' vollzieht die Initialisierung von diskreten, mehrdimensionalen Zufallsvariablen. Die Bedeutung der Parameter 'X', 'Bezeichnung' und 'Vert' überträgt sich sinngemäß aus der Beschreibung zur Prozedur 'InitDskrZV1D', Abschnitt 5.7.7 . Der Wertebereich der Verteilung der Zufallsvariablen X wird durch die Parameter 'a' (Untergrenze in allen Dimensionen) und 'b' (Obergrenze in allen Dimensionen) festgelegt (siehe auch Kapitel 4.2). Sie sind vom Typ 'ITupel' und müssen deshalb mit Hilfe der Prozeduren 'BildeITupel$n$' (Abschnitt 5.6.4) erzeugt werden. Wenn auf die Wahrscheinlichkeit von Massepunkten der Verteilung von X auch mit Feldindizierung zugegriffen werden soll, so müssen die mit 'a' und 'b' festgelegten Intervallgrenzen mit den Bereichsgrenzen des Verteilungsfeldes übereinstimmen.

Die von der Prozedur 'InitDskrZVMD' ausgeführten Arbeitsschritte stimmen mit denen der Prozedur 'InitDskrZV1D' überein.

5.7.9 Initialisierung diskreter Zufallsvariablen mit unendlichem Wertebereich

```
PROCEDURE InitDskrUnendlZV(VAR X: ZufallsVar; Bezeichnung: ZVName;
   Vert: ADDRESS; a,b: INTEGER);
```

Durch die Prozedur 'InitDskrUnendlZV' wird eine Zufallsvariable X mit einem eindimensionalen, diskret unendlichen Wertebereich initialisiert. Die Parameter 'X', 'Bezeichnung' und 'Vert' werden wie in der Prozedur 'InitDskrZV1D' benutzt (Abschnitt 5.7.7). Die Parameter 'a' und 'b' legen die Grenzen des Intervalls fest, auf dem die Verteilung von X durch eine VME-0 approximiert wird (vergleiche Abschnitt 4.2.2). Die Prozedur führt die folgenden Arbeitsschritte aus:

- Anforderung von Arbeitsspeicher beim Laufzeitsystem zum Anlegen einer Datenstruktur vom Typ 'ZVStrukt'
- Die Initialisierung der Komponenten dieser Datenstruktur wird vorgenommen. Die Diskriminante ('Typ') des varianten Verbundes erhält den Wert 'Diskr101'; damit wird die zweite Variante des Verbundes ausgewählt. Anschließend wird der Komponente 'aDiskrDef' der Wert von 'a', der Komponente 'bDiskrDef' der Wert von 'b' zugewiesen.
- Alle Feldkomponenten des Verteilungsfeldes werden mit dem Wert '0.0' vorbesetzt, ebenso wie die Komponenten 'ZVStrukt.M0links', 'ZVStrukt.M1links', 'ZVStrukt.M0rechts' und 'ZVStrukt.M1rechts'.
- Der Komponente 'Erkennung' wird der Wert 'Kennung' zugewiesen.
- Wenn alle vorhergehenden Schritte erfolgreich durchgeführt werden konnten, wird dem Parameter 'X' die Adresse der erzeugten Datenstruktur zugewiesen.

5.7.10 Initialisierung stetiger Zufallsvariablen

```
PROCEDURE InitStetZV(VAR X: ZufallsVar; Bezeichnung: ZVName;
                     Vert: ADDRESS; a,b: INTEGER; delt: Real);
```

Durch die Prozedur 'InitStetZV' wird eine Zufallsvariable X mit eindimensionalem, stetigem Wertebereich initialisiert. Die Bedeutung der Parameter 'X', 'Bezeichnung' und 'Vert' entspricht denen der Prozedur 'InitDskrZV1D', Abschnitt 5.7.7. Der Parameter 'delt' gibt die Länge der Intervalle an, mit denen die Dichtefunktion von X approximiert wird. Das Intervall, auf dem die Dichtefunktion mit Dichten maximaler Entropie der Ordnung 0 approximiert wird, besitzt die Untergrenze $a \cdot \delta$ und die Obergrenze $(b+1) \cdot \delta$. Diese ergeben sich aus den Parametern 'a', 'b' und 'delt' (Genaueres hierzu in Abschnitt 4.2).

Die Prozedur führt folgende Arbeitsschritte aus:

- Anforderung von Arbeitsspeicher beim Laufzeitsystem zum Anlegen einer Datenstruktur vom Typ 'ZVStrukt'.
- Initialisierung der Komponenten dieser Datenstruktur. Insbesondere wird der Komponente 'ZVStrukt.VertStet' der Wert des Parameters 'Vert' und der Komponente 'ZVStrukt.VertDskr' der Wert 'NIL' zugewiesen. Der Diskriminante 'Typ' des varianten Verbundes 'ZVStrukt' wird der Wert 'Stet101' zugewiesen; dadurch wird die erste Variante des Verbundes ausgewählt.
- Das durch die Parameter 'Vert', 'a' und 'b' festgelegte Verteilungsfeld wird nullgesetzt, d.h. alle Einträge werden mit dem Wert '0.0' vorbesetzt. Außerdem werden die Komponenten 'Wlinks', 'Wrechts', 'Mlinks'und 'Mrechts' auf den Wert '0.0' gesetzt.
- Der Komponente 'Erkennung' wird der Wert 'Kennung' zugewiesen.
- Wenn alle vorhergehenden Schritte erfolgreich durchgeführt werden konnten, wird dem Parameter 'X' die Adresse der erzeugten Datenstruktur zugewiesen.

5.7.11 Initialisierung von gemischten Zufallsvariablen

```
PROCEDURE InitGemZV(VAR X: ZufallsVar; Bezeichnung: ZVName;
                    DVert: ADDRESS; a,b: INTEGER;
                    SVert: ADDRESS; as,bs: INTEGER; delt: Real);
```

Die Aufgabe der Prozedur 'InitGemZV' ist die Initialisierung von Zufallsvariablen, deren Wertebereich zum Teil diskret, zum Teil stetig ist . Die Parameter 'DVert', 'a' und 'b' entsprechen den Parametern 'Vert', 'a' und 'b' der Prozedur 'InitDskrZV1D', die Parameter 'SVert', 'as', 'bs' und 'delt' den Parametern 'Vert', 'a', 'b' und 'delt' der Prozedur 'InitStetZV'. Der Diskriminante 'Typ' des varianten Verbundes 'ZVStrukt' wird der Wert 'Gemischt' zugewiesen.

5.7.12 Auflösen der Beschreibungsstrukturen von Zufallsvariablen

```
PROCEDURE UnInitZV(VAR X: ZufallsVar);
```

Wie oben beschrieben, wird von den Prozeduren 'InitDskrZV1D', 'InitDskrUnendl', 'InitDskrZVMD', 'InitStetZV' und 'InitGemZV' dynamisch Speicher angefordert, um die Datenstruktur 'ZVStrukt' zu generieren. Die Prozedur 'UnInitZV' löst die zur Zufallsvariablen X gehörende Datenstruktur 'ZVStrukt' auf und gibt deren Speicherplatz frei. Die Variable 'X' wird auf den Wert 'NIL' gesetzt. Werden in einem Unterprogramm eines Analyseprogramms Zufallsvariablen durch 'InitxxZVxx'-Prozeduren erzeugt, so müssen am Ende des Unterprogramms diese Zufallsvariablen durch 'UnInitZV' aufgelöst werden. Anderenfalls können Speicherüberläufe oder langwierige Garbage-Collections auftreten.

5.7.13 Erzeugen von Hilfszufallsvariablen

```
PROCEDURE GenZV(VAR hZV: ZufallsVar; Bezeichnung: ZVName; X: ZufallsVar);
```

Die Prozedur dient der Erzeugung von Hilfszufallsvariablen, die bei der Implementierung von Analyse-Algorithmen notwendig sein können, um Zwischenergebnisse zu speichern. Der Unterschied zwischen diesen Hilfszufallsvariablen und den sonst verwendeten Zufallsvariablen besteht darin, daß die Feldvariablen, die die Wahrscheinlichkeiten der Verteilung aufnehmen, nicht vom Benutzer deklariert werden müssen, sondern dynamisch von VERENA erzeugt werden.

Durch den Parameter 'Bezeichnung' wird der erzeugten Hilfszufallsvariablen, wie bei anderen Zufallsvariablen auch, ein Name zugewiesen.

Der Parameter 'X' legt den Wertebereich der zu erzeugenden Zufallsvariablen fest. Es wird eine Zufallsvariable erzeugt, deren Wertebereich mit dem der Zufallsvariablen X identisch ist. X dient also als Muster. Die Verteilung wird dabei nicht kopiert, sondern es werden alle Wahrscheinlichkeiten und Momente mit dem Wert 0.0 initialisiert.

Wird die dynamisch erzeugte Zufallsvariable nicht mehr benötigt, so muß sie mit der Prozedur 'DelZV' (Abschnitt 5.7.14) wieder gelöscht werden, damit nicht unnötig Speicherplatz belegt wird.

5.7.14 Auflösen von Hilfszufallsvariablen

```
PROCEDURE DelZV(VAR hZV: ZufallsVar);
```

Mit dieser Prozedur wird der durch die Hilfszufallsvariable 'hZV' belegte Arbeitsspeicher freigegeben und 'hZV' der Wert 'NIL' zugewiesen. Die Prozedur darf nur auf dynamisch angelegte Zufallsvariablen angewendet werden, die mit der Prozedur 'GenZV' oder 'EinpunktVert' (vgl. Abschnitt 5.7.17) erzeugt wurden. Die Anwendung auf anders erzeugte Zufallsvariablen kann zu einem Adressierungsfehler führen.

5.7.15 Dynamisch erzeugte Zufallsvariablen

Die Prozedur 'DynZV' dient – wie die bereits beschriebenen Prozeduren zur Initialisierung von Verteilungen von Zufallsvariablen – der Erzeugung aller Datenstrukturen, die zu einer Zufallsvariable gehören. Der einzige Unterschied ist, daß, anders als bei den 'InitxxZVxx'-Prozeduren, die Verteilungsfelder mit 'DynZV' dynamisch erzeugt werden. 'DynZV' unterscheidet sich auch von der Prozedur 'GenZV'. Der Benutzer kann mit 'DynZV' zur Laufzeit Zufallsvariablen erzeugen, ohne daß wie mit 'GenZV' eine zuvor erzeugte Zufallsvariable „kopiert“ werden muß. Da die Prozedur 'DynZV' weniger Parameter besitzt und flexibler einzusetzen ist als die 'InitxxZVxx'-Prozeduren, sollte der Benutzer die Prozedur 'DynZV' verwenden, wenn die Wertebereichsgrenzen nicht in parametrisierter Form vorliegen (vgl. Def. von 'Bereich').

```
PROCEDURE DynZV(VAR ZV: ZufallsVar; ZName: ZVName; ZTyp: ZVTyp;
                Bereich: ARRAY OF CHAR);
```

Die Prozedur 'DynZV' legt dynamisch alle Strukturen an, die man für die Festlegung einer Verteilung benötigt. 'ZName' gibt den Namen an, 'ZTyp' legt den Typ der Verteilung der Zufallsvariablen 'ZV' fest. Um den Typ der Zufallsvariablen angeben zu können, muß der Benutzer den Datentyp 'ZVTyp' aus dem Modul 'RVBasics' importieren.

Mit dem Parameter 'Bereich' werden die Grenzen des DME-0-Bereichs festgelegt. Der aktuelle Parameter 'Bereich' muß eine Zeichenkettenkonstante oder eine Zeichenkettenvariable sein. 'Bereich' muß wie folgt aufgebaut sein, falls 'ZV' diskret ist:

$$\begin{array}{lcl}
< \mathit{Bereich} > & ::= & \text{“}''\text{[“} < \mathit{Grenzen} > \text{“]“} \quad \{ \text{“,[“} < \mathit{Grenzen} > \text{“]“} \} \text{ “}''\text{“} \\
< \mathit{Grenzen} > & ::= & < a_i > \text{“..“} < b_i > \\
< a_i > & ::= & < \mathit{Integer} > \\
< b_i > & ::= & < \mathit{Integer} > \\
< \mathit{Integer} > & ::= & [\text{“} - \text{“}] < \mathit{Digit} > \{< \mathit{Digit} >\} \\
< \mathit{Digit} > & ::= & \text{“0“} | \ldots | \text{“9“}
\end{array}$$

DME-0- und VME-0-Bereiche von Verteilungen werden syntaktisch genauso festgelegt wie die Grenzen eines MODULA-2-Feldes. Da der Parameter 'Bereich' eine Zeichenkette ist, wird er durch Begrenzungszeichen (Anführungszeichen) eingeschlossen. Statt der hier verwendeten doppelten Anführungszeichen können auch die einfachen Anführungszeichen benutzt werden. Jedoch darf man einfache und doppelte Anführungszeichen nicht gemischt als Zeichenkettenbegrenzer einsetzen.

Ist die Zufallsvariable 'ZV' vom Typ 'DiskrEndl' (diskret endlich), so kann die Dimension der Verteilung größer als eins sein. Die Dimension einer endlichen diskreten Zufallsvariablen ergibt sich dann aus der Anzahl der angegebenen Abschnitte, die durch eckige Klammern eingeschlossen sind. Die Werte von a_i und b_i geben die Unter- und Obergrenzen der VME-0-Bereiche in den verschiedenen Dimensionen an.

Ist der Typ der Zufallsvariable 'ZV' jedoch 'Diskr101' (diskret, approximiert mit 101-Dichten), so muß die Dimension gleich eins sein. Anderenfalls wird eine Fehlermeldung ausgegeben.

Falls 'ZV' stetig ist, besitzt 'Bereich' folgende Gestalt:

$$\begin{array}{lcl}
< \mathit{Bereich} > & ::= & \text{“}''\text{[“} < a_1 > \text{“..“} < b_1 > \text{“],“} < \mathit{PosReal} > \text{“}''\text{“} \\
< a_1 > & ::= & < \mathit{Integer} > \\
< b_1 > & ::= & < \mathit{Integer} > \\
< \mathit{Integer} > & ::= & [\text{“} - \text{“}] < \mathit{Digit} > \{< \mathit{Digit} >\} \\
< \mathit{PosReal} > & ::= & < \mathit{Digit} > \{< \mathit{Digit} >\}\text{“.“}\{< \mathit{Digit} >\}[< \mathit{Exponent} >] \\
< \mathit{Exponent} > & ::= & \text{“}E\text{“}[\text{“} + \text{“}|\text{“} - \text{“}] < \mathit{Digit} > \{< \mathit{Digit} >\} \\
< \mathit{Digit} > & ::= & \text{“0“} | \ldots | \text{“9“}
\end{array}$$

Stetige Zufallsvariablen müssen eindimensional sein. 'PosReal' repräsentiert die Intervallbreite δ als REAL- oder LONGREAL-Konstante und ist durch ein Komma von der Angabe der Grenzen getrennt.

Ist 'ZV' gemischt, so sieht 'Bereich' wie folgt aus:

$$\begin{array}{lcl}
< Bereich > & ::= & \text{“}''\text{[“} < a_{disk} > \text{“..“} < b_{disk} > \text{“],[“} < a_{stet} > \text{“..“} < b_{stet} > \text{“],“} \\
 & & < PosReal > \text{“}''\text{“} \\
< a_{ds} > & ::= & < Integer > \\
< b_{ds} > & ::= & < Integer > \\
< a_{st} > & ::= & < Integer > \\
< b_{st} > & ::= & < Integer > \\
< Integer > & ::= & [\text{“} - \text{“}] < Digit > \{< Digit >\} \\
< PosReal > & ::= & < Digit > \{< Digit >\}\text{“.“}\{< Digit >\}[< Exponent >] \\
< Exponent > & ::= & \text{“}E\text{“}[\text{“} + \text{“}|\text{“} - \text{“}] < Digit > \{< Digit >\} \\
< Digit > & ::= & \text{“0“}|\ldots|\text{“9“}
\end{array}$$

In diesem Fall werden die ersten Grenzangaben als Grenzen des (diskreten) VME-0-Bereichs, die zweiten Grenzangaben als Grenzen des (stetigen) DME-0-Bereichs interpretiert.

Wenn eine Zufallsvariable und die dazugehörenden Datenstrukturen nicht angelegt werden können, so wird eine Fehlermeldung ausgegeben. Dies kann vorkommen, wenn der Benutzer bei der Festlegung von 'Bereich' die beschriebene Syntax mißachtet oder die von der Prozedur 'DynZV' ermittelte Dimension nicht zum Typ der Zufallsvariablen 'ZV' paßt.

5.7.16 Löschen dynamisch erzeugter Zufallsvariablen

```
PROCEDURE DelDynZV(ZV:ZufallsVar);
```

Um den Speicher des Rechners nicht unnötigt zu belasten, sollte der Benutzer die mit 'DynZV' erzeugten Zufallsvariablen löschen, wenn er diese Zufallsvariablen nicht mehr benötigt. Bei Aufruf der Prozedur 'DelDynZV' werden alle Strukturen, die zu der Zufallsvariablen 'ZV' gehören, aufgelöst, und der freigewordene Heap-Speicher kann wieder für die Erzeugung anderer dynamischer Objekte benutzt werden.

Die Zufallsvariable 'ZV' muß mit der Prozedur 'DynZV' erzeugt worden sein, da sonst eine Fehlermeldung ausgegeben wird.

Beispiel:

```
MODULE Beispiel;
FROM RVBasics IMPORT DynZV,DelDynZV,ZVTyp;
(* Weitere Import-Anweisungen *)
     :
     :
VAR X1,X2,X3,X4:ZufallsVar;
(* Weitere Deklarationen *)
     :
     :
BEGIN
   (* Erzeugen einer diskreten, endlichen Verteilung *)
   DynZV(X1,'X1 ',DiskrEndl,"[-5..5],[0..10]");

   (* Erzeugen einer diskreten 101-Verteilung *)
   DynZV(X2,'X2 ',Diskr101,"[-5..5]");

   (* Erzeugen einer stetigen Verteilung *)
   DynZV(X3,'X3 ',stet101,"[-5..5],0.1");

   (* Erzeugen einer gemischten Verteilung *)
   DynZV(X4,'X4 ',Gemischt,"[0..10],[-5..5],0.1");

   (* Weitere Anweisungen *)
       :
       :
   (* L"oschen der Strukturen *)
   DelDynZV(X1);
   DelDynZV(X2);
   DelDynZV(X3);
   DelDynZV(X4);
END Beispiel.
```

5.7.17 Erzeugen konstanter Zufallsvariablen

```
PROCEDURE EinpunktVert(VAR X: ZufallsVar; Bezeichnung : ZVName; k: INTEGER);
```

Diese Prozedur gibt in 'X' eine dynamisch erzeugte konstante („einpunktverteilte“) Zufallsvariable zurück. Ein Aufruf einer der 'InitxxZVxx'-Prozeduren ist für X nicht erforderlich, weil das Verteilungsfeld, das in diesem Falle nur aus einer Komponente besteht, dynamisch erzeugt wird. Wenn die Zufallsvariable nicht mehr benötigt wird, sollte sie mit der Prozedur 'DelZV' entfernt werden, um nicht unnötig Speicherplatz zu belegen. Der Name der Zufallsvariablen wird mit dem Parameter 'Bezeichnung' festgelegt.

5.7.18 Vertauschen der Verteilungen zweier Hilfszufallsvariablen

```
PROCEDURE TauscheZV(VAR X,Y: ZufallsVar);
```

Werden Verteilungen iterativ berechnet, so ist es zu Beginn eines Iterationsschrittes häufig notwendig, einer Zufallsvariablen eine Verteilung zuzuweisen, die in einem vorangegangenen Iterationsschritt berechnet wurde.

Benutzt man in diesem Fall dynamisch erzeugte Zufallsvariablen, so kann für eine solche Zuweisung die Prozedur 'TauscheZV' benutzt werden. Sie vertauscht lediglich die Werte der (Zeiger-) Variablen 'X' und 'Y', anstatt die gesamten Verteilungen zu übertragen. Wurden die Zufallsvariablen statisch erzeugt, darf 'TauscheZV' *nicht* benutzt werden. In diesem Fall sind die Verteilungsfelder (durch den Linker) an feste Speicheradressen gebunden. Diese Bindung kann von der Prozedur nicht verändert werden.

Zum Einsatz von Hilfszufallsvariablen und zur Verwendung der Prozedur 'TauscheZV' das folgende Beispiel:

```
GenZV(HZV1,'HZV1',X);    (* X ist eine beliebige Zufallsvariable, HZV1 *)
GenZV(HZV2,'HZV2',X);    (* und HZV2 sind Hilfs-Zufallsvariablen       *)
ZuweisenZV(X,HZV1);

REPEAT
  BerechneHZV2ausHZV1(HZV1,HZV2);
  TauscheZV(HZV1,HZV2);
UNTIL Bedingung;

ZuweisenZV(HZV1,X);
DelZV(HZV1);
DelZV(HZV2);
```

5.7.19 Kennzeichnung einer Verteilung als verändert

```
PROCEDURE VertVeraendert(X: ZufallsVar);
```

Mit dieser Prozedur wird dem Programmsystem angezeigt, daß die Verteilung einer Zufallsvariablen verändert wurde. Alle Prozeduren von VERENA, die die Veränderung von Verteilungen bewirken, rufen diese Prozedur selbständig auf. Verändert der Benutzer innerhalb seines Analyseprogramms Verteilungen durch einen direkten Zugriff, so muß er dies dem Programmsystem durch Aufruf dieser Prozedur anzeigen, bevor eine andere Prozedur von VERENA aufgerufen wird.

5.7.20 Verhinderung der Nullsetzung von Verteilungen

```
PROCEDURE NichtNullsetzen(X: ZufallsVar);
```

Bei zahlreichen Prozeduren des Programmsystems werden die Verteilungen der Zufallsvariablen grundsätzlich nullgesetzt, bevor sie das Resultat einer Operation aufnehmen. Es kann Ausnahmefälle geben, in denen diese Nullsetzung von Verteilungen nicht gewünscht wird.

Erreicht wird das mit der Verwendung der Prozedur 'NichtNullsetzen' vor dem Aufruf der gewünschten Prozedur. Die Prozedur setzt die Komponente 'Status' der Zufallsvariablen X auf einen Wert, der der Prozedur 'Setze0Vert' anzeigt, daß eine (nochmalige) Nullsetzung nicht notwendig ist.

Anwendungsbeispiel:

```
...
BinomialVert(X,4,RK(0.3));
NichtNullsetzen(X);
GeometrVert(X,p);         (* ACHTUNG! Die Verteilung hat nach
                             dieser Prozedur die Norm 2.0 !!! *)
NormiereZV(X);
...
```

Die Prozedur sollte allerdings nur angewendet werden, wenn aus Gründen der Programmlaufzeit eine Verwendung der Prozedur 'SummWGewichtet' (Abschn. 5.8.10) nicht in Frage kommt. Von der Anwendung auf Zufallsvariablen mit stetigem Wertebereich wird abgeraten, da hierbei aufgrund der dynamischen Festlegung der Intervallgrenzen 'aStet' und 'bStet' schwer vorhersehbare Ergebnisse entstehen können.

Das obige Beispiel kann unter Umgehung der Prozedur 'NichtNullsetzen' wie folgt programmiert werden:

```
...
GenZV(Binom,'Binom',X);
GenZV(Geometr,'Geometr',X);

BinomialVert(Binom,4,RK(0.3));
GeometrVert(Geometr,p);

SummWGewichtet(Binom,RK(0.5),SOV(X));
SummWGewichtet(Geometr,RK(0.5),X);

DelZV(Binom);
DelZV(Geometr);
...
```

5.8 Das Modul 'RndVar0'

In dem Modul 'RndVar0' sind Prozeduren zusammengefaßt, die der Festlegung spezieller benutzerdefinierter Verteilungen, d.h. der Zuordnung von Wahrscheinlichkeiten bzw. Momenten zu Teilmengen des Wertebereichs von Zufallsvariablen dienen. Die den Prozeduren übergebenen Zufallsvariablen besitzen im allgemeinen keine normierten Verteilungen. Derartige Gebilde werden im folgenden als *Pseudo-Verteilungen* bezeichnet. Durch (mehrfache) Anwendung der hier definierten Prozeduren entstehen nach und nach korrekte Verteilungen.

Exportierte Typen:

5.8.1 Diskrete Verteilungsfunktionen

```
TYPE PMF  = PROCEDURE(INTEGER): Real;
```

Der Typ 'PMF' dient der Deklaration von Prozeduren, die mit der Bedeutung von diskreten Verteilungsfunktionen (Probability-Mass-Functions) der Prozedur 'BesetzTWbereichPMF' (Abschnitt 5.8.7) als Parameter übergeben werden können. So ist es möglich, mit diesen Prozeduren willkürlich diskrete Verteilungen festzulegen.

5.8.2 Typ für die Darstellung von DME-1

```
TYPE DME1 = RECORD
              Grenze : INTEGER;
              W,M    : Real;
            END;
```

In einer Variable vom Typ 'DME1' sind alle Daten gespeichert, die bei Vorgabe einer Intervallbreite δ zur Bestimmung einer Dichte maximaler Entropie der Ordnung 1 (DME-1) hinreichend sind, deren Wertebereich A entweder durch das Intervall $(-\infty, \delta \cdot \text{Grenze})$ oder das Intervall $[\delta \cdot \text{Grenze}, \infty)$ gegeben ist. Die Verbundkomponenten 'W' (Wahrscheinlichkeit) und 'M' (1. Moment) sind wie folgt festgelegt:

$$\mathrm{W} := \int_A f(x)dx$$

und

$$\mathrm{M} := \int_A xf(x)dx,$$

wobei $f(x)$ die anzunähernde Dichtefunktion ist.

Exportierte Prozeduren:

5.8.3 Zugriff auf einzelne Wahrscheinlichkeiten

```
PROCEDURE Pvon(X: ZufallsVar; i: INTEGER): Real;
```

Wenn X eine eindimensionale diskrete oder gemischte Zufallsvariable ist, kann mit dieser Funktion die Wahrscheinlichkeit P{X=i} ermittelt werden.

Die Funktion wird dann benötigt, wenn das zu 'X' gehörende Verteilungsfeld nicht mit Feldindizierung angesprochen werden kann, beispielsweise bei Zufallsvariablen, die durch 'GenZV' oder 'DynZV' erzeugt wurden.

Wenn die Dimension einer diskreten endlichen Zufallsvariablen größer als eins ist, darf die Funktion 'Pvon' nicht verwendet werden. Die Benutzung dieser Funktion bei mehrdimensionalen Zufallsvariablen führt zu einem Fehler. Für Zufallsvariablen mit mehrdimensionaler Verteilung steht die Funktion 'PvonMD' zur Verfügung:

```
PROCEDURE PvonMD(X : ZufallsVar; Tupel : ARRAY OF CHAR):Real;
```

Diese Funktion liefert die Wahrscheinlichkeit $P\{X = (x_1, \ldots, x_d)\}$ der d-dimensionalen Zufallsvariablen X. Dem Parameter 'Tupel' ist als aktueller Parameter eine Zeichenkette zu übergeben, die den Punkt $(x_1, \ldots, x_d)$ beschreibt. Der aktuelle Parameter muß folgende Form haben:

$$\text{“}\underbrace{[< Integer >, \ldots, < Integer >]}_{d-\text{mal}}\text{“}$$

bzw.

$$'\underbrace{[< Integer >, \ldots, < Integer >]}_{d-\text{mal}}'$$

Beispiel:

Von einer dreidimensionalen Zufallsvariablen X soll die Wahrscheinlichkeit des Wertes $(2, 5, -5)$ ermittelt und einer Variablen p zugewiesen werden. Folgende Anweisungen sind im Analyseprogramm zu benutzen:

```
MODULE ...;
   :
 (* IMPORT-Anweisungen und Typ-Deklarationen *)
   :
VAR p : Real;
    X : ZufallsVar;

 (* Weitere Deklarationen *)

BEGIN
   :
   :
   p:=PvonMD(X,"[2,5,-5]");
   :
   :
END ...  .
```

5.8.4 Kumulative Verteilungsfunktionen

```
PROCEDURE Fvon(X: ZufallsVar; y: Real): Real;
```

Die Prozedur ermittelt den Wert der kumulativen Verteilungsfunktion der Zufallsvariablen X an der Stelle y, also die Wahrscheinlichkeit P$\{-\infty < X \leq y\}$.

5.8.5 Veränderung einzelner Wahrscheinlichkeiten

```
PROCEDURE Pgleich(X: ZufallsVar; i: INTEGER; p: Real);
```

Wenn X eine eindimensionale diskrete oder gemischte Zufallsvariable ist, kann mit dieser Funktion die Wahrscheinlichkeit 'p' = P{X=i} festgelegt werden.

Die Funktion bildet somit das Gegenstück zur Funktion 'Pvon' und wird ebenfalls dann benötigt, wenn das zu X gehörende Verteilungsfeld nicht über Feldindizierung angesprochen werden kann.

Wenn der Benutzer einzelne Wahrscheinlichkeiten verändern möchte, so sollte er diese Prozedur benutzen und auf die Programmierung eigener Feldzugriffsoperationen verzichten. Dies erspart dem Benutzer den Aufruf 'VertVeraendert', da die Prozedur 'Pgleich' die Prozedur 'VertVeraendert' aufruft. Es soll allerdings nicht verschwiegen werden, daß die in Abschnitt 5.5.2 beschriebene Methode Geschwindigkeitsvorteile gegenüber der Prozedur 'Pgleich' besitzt.

5.8.6 Mehrere Wahrscheinlichkeiten auf einen Wert setzen

```
PROCEDURE BesetzTWbereichKonst(X          : ZufallsVar;
                               Start, Ende: INTEGER;
                               Prob       : Real);
```

Sei X eine eindimensionale Zufallsvariable, deren diskreter Wertebereich die Teilmenge [Start : Ende] enthalte. Die Prozedur 'BesetzTWbereichKonst' (besetze Teilmenge eines Wertebereichs mit konstanter Wahrscheinlichkeit) bewirkt für alle $i \in$ [Start : Ende] die Zuweisung

$$P\{X = i\} := \text{Prob} \ .$$

5.8.7 Verteilung durch diskrete Verteilungsfunktion festlegen

```
PROCEDURE BesetzTWbereichPMF(X          : ZufallsVar;
                             Start, Ende: INTEGER;
                             Prob       : PMF);
```

Sei X eine eindimensionale Zufallsvariable, deren diskreter Wertebereich die Teilmenge [Start : Ende] enthalte. Ferner sei $g : \mathbb{Z} \rightarrow \mathbb{R}$ eine durch die Prozedur 'Prob' definierte Funktion (Abschnitt 5.8.1). Dann führt die Prozedur 'BesetzTWbereichPMF' (besetze Teilmenge eines Wertebereichs gemäß einer Verteilungsfunktion (PMF)) die Zuweisung

$$P\{X = i\} := g(i)$$

für alle $i \in$ [Start : Ende] aus.

5.8.8 Addition von Dichten über linksseitig unbegrenzten Intervallen

```
PROCEDURE DME1LAufteilen(X: ZufallsVar; DME1L: DME1);
```

Diese Prozedur addiert zwei Dichten über dem Intervall $(-\infty, \delta \cdot (\text{DME1L.Grenze}))$. Dies ist eine Operation, die bei der Realisierung der Faltung im Programmsystem häufig benutzt wird.

Zunächst muß dazu die Darstellung der durch 'DME1L' approximierten Dichte g an die von X über dem Intervall $(-\infty, \delta \cdot (\text{DME1L.Grenze}))$ angepaßt werden. Danach werden die beiden Dichten addiert, so daß als Ergebnis der Operation eine Pseudo-Verteilung von X vorliegt.

Die Pseudo-Verteilung von X sei vor dem Prozeduraufruf gegeben durch die 101-Approximation der Dichte $f(x)$ mit den Wahrscheinlichkeiten

$$p_i = \int_{\delta i}^{\delta(i+1)} f(x)dx, \quad i \in [a : b],$$

$$W_{\text{links}} = \int_{-\infty}^{\delta a} f(x)dx$$

$$W_{\text{rechts}} = \int_{\delta(b+1)}^{\infty} f(x)dx$$

und den beiden ersten Momenten

$$M_{\text{links}} = \int_{-\infty}^{\delta a} xf(x)dx$$

$$M_{\text{rechts}} = \int_{\delta(b+1)}^{\infty} xf(x)dx \ .$$

Für 'DME1L' gelte

$$\text{DME1L.W} := \int_{-\infty}^{\delta(\text{DME1L.Grenze})} g(x)dx$$

und

$$\text{DME1L.M} := \int_{-\infty}^{\delta(\text{DME1L.Grenze})} xg(x)dx\ .$$

Über dem Intervall $[\delta \cdot (\text{DME1L.Grenze}), +\infty)$ sei $g(x) := 0$ definiert. Dann gilt für die Pseudo-Verteilung von X nach Ausführung von 'DME1LAufteilen':

$$p_i' = p_i + \int_{\delta i}^{\delta(i+1)} g(x)dx, \quad i \in [a : b],$$

$$W_{\text{links}}' = W_{\text{links}} + \int_{-\infty}^{\delta a} g(x)dx$$

$$W_{\text{rechts}}' = W_{\text{rechts}} + \int_{\delta(b+1)}^{\infty} g(x)dx$$

und für die beiden ersten Momente

$$M_{\text{links}}' = M_{\text{links}} + \int_{-\infty}^{\delta a} xg(x)dx$$

$$M_{\text{rechts}}' = M_{\text{rechts}} + \int_{\delta(b+1)}^{\infty} xg(x)dx\ .$$

5.8.9 Addition von Dichten über rechtsseitig unbegrenzten Intervallen

```
PROCEDURE DME1RAufteilen(X: ZufallsVar; DME1R: DME1);
```

Die Wirkung dieser Prozedur ist entspricht der der Prozedur 'DME1LAufteilen' (Abschnitt 5.8.8), wobei die Variable 'DME1R' eine Dichte maximaler Entropie der Ordnung 1 über dem Intervall $[\delta \cdot (\text{DME1R.Grenze}), \infty)$ spezifiziert.

5.8.10 Gewichtete Addition von Verteilungen

```
PROCEDURE SummWGewichtet(X: ZufallsVar; Prob: Real; Y: ZufallsVar);
```

Mit dieser Prozedur können Pseudo-Verteilungen gewichtet addiert werden. Die Zufallsvariable Y ist sowohl Eingabe- als auch Ausgabeparameter. Berechnet wird die (Pseudo-) Verteilung der Zufallsvariablen Y. Zu den durch ihre bisherige Pseudo-Verteilung gegebenen Wahrscheinlichkeiten und Momenten werden die durch zu X gehörenden Wahrscheinlichkeiten und Momente addiert, gewichtet mit der Wahrscheinlichkeit 'Prob'.

Für die Verteilungsfunktionen F_X und F_Y der beiden Zufallsvariablen X und Y gilt:

$$F'_Y(x) \approx F_Y(x) + p \cdot F_X(x), \quad \text{für alle } x \in I\!R \text{ und } p = \text{'Prob'},$$

wobei F'_Y die Verteilungsfunktion von Y **nach** Anwendung der Prozedur bezeichnet.

Bedingung für die Benutzung der Prozedur ist, daß der diskrete Wertebereich der Zufallsvariablen X Teilmenge des diskreten Wertebereiches der Zufallsvariablen Y ist und beide Wertebereiche die gleiche Dimension haben. Ferner muß der stetige Wertebereich der Zufallsvariablen X Teilmenge des stetigen Wertebereiches der Zufallsvariablen Y, und falls diese nicht leer sind, mit der gleichen Intervallbreite δ aufgeteilt sein.

Die Operation liefert im allgemeinen eine Approximation, da die DME-0- bzw. VME-0-Bereiche von 'X' denen von 'Y' angepaßt werden.

Ein Beispiel für die Anwendung der Prozedur 'SummWGewichtet' findet sich in Abschnitt 6.8.

5.9 Das Modul 'RndVar1'

Das Modul 'RndVar1' enthält Prozeduren für die tabellarische Ausgabe und für die graphische Darstellung von Verteilungen. Für die graphische Darstellung von Verteilungen wird die Seitenbeschreibungssprache PostScript benutzt.

Allen formalen Parametern vom Typ 'ARRAY OF CHAR' können Zeichenkettenkonstanten und Zeichenkettenvariablen, die als 'ARRAY [a..b] OF CHAR' deklariert wurden als aktuelle Parameter übergeben werden. Es gibt keine Längenbeschränkung. Zeichenkettenkonstanten der Länge 1 sind bei einigen Compilern nicht zugelassen, da sie als CHAR-Typ interpretiert werden und somit ein Typkonflikt vorliegt. Durch Anhängen eines Leerzeichens an einzelne Zeichen wird das Problem umgangen.

Exportierte Prozeduren:

5.9.1 Ausgabe von Verteilungen in Tabellenform

```
PROCEDURE WriteZV(X            : ZufallsVar;
                  Kommentar    : ARRAY OF CHAR;
                  Verteilungen: BOOLEAN);
```

Informationen über die Zufallsvariable X werden in die aktuelle Ausgabedatei geschrieben. Wurde die Ausgabe nicht mit der Prozedur 'OpenOutput' (Abschn. 5.2.2) umgeleitet, erfolgt die Ausgabe auf dem Bildschirm. Die Zeichenkette 'Kommentar' erscheint in der Kopfzeile der Ausgabe. Wird dem formalen Parameter 'Verteilungen' der Wert TRUE übergeben, so wird auch der gesamte Wertebereich der Zufallsvariablen mit allen Wahrscheinlichkeiten ausgegeben, anderenfalls erscheinen lediglich Angaben über die Art und Größe des Wertebereichs, sowie die Norm und das 1. Moment der Verteilung.

5.9.2 Ausgabe von Verteilungen als PostScript-Graphik

```
PROCEDURE WriteZVtoPS(X                    : ZufallsVar;
                      FileName,Caption     : ARRAY OF CHAR;
                      Height, Width        : Real);
```

Die Prozedur 'WriteZVtoPS' ist auf Zufallsvariablen mit eindimensionalem Wertebereich anwendbar und erzeugt in Form einer PostScript-Datei eine Graphik, die die Verteilung der Zufallsvariablen X darstellt. Außerdem wird eine TEX-Datei angelegt, mit deren Hilfe eine einfache Einbindung der PostScript-Graphik in ein TEX- oder LATEX-Dokument ermöglicht wird. Die Namen der Dateien werden durch den formalen Parameter 'FileName' festgelegt, der den betriebssystemspezifischen Konventionen genügen muß und *keine* Namenserweiterung (Extension, z.B. '.tex') aufweisen darf. Die Namenserweiterungen '.ps' bzw. '.tex' für die PostScript- und die TEX-Datei werden von der Prozedur selbständig an den Dateinamen angehängt. Die Zeichenkette, die mit dem Parameter 'Caption' übergeben wird, erscheint als Bildunterschrift im TEX-Dokument. Die Parameter 'Height' und 'Width' geben die Größe der Graphik in Millimetern an. In die TEX-Datei fügt das Programmsystem die PostScript-Datei mit dem Befehl `\special{insert <FileName>}` ein.

Der Benutzer kann die Datei '<FileName>.tex' in ein LATEX-Dokument mit dem LATEX-Kommando `\input{<FileName>}` einbinden, wenn die beiden Dateien zuvor in das Verzeichnis kopiert wurden, in dem auch die Dateien des LATEX-Dokuments stehen.

Wichtig: Eine solche Einbindung ist natürlich nur dann sinnvoll, wenn die Ausgabe des Dokumentes auf einem PostScript-fähigen Drucker erfolgt. Die beschriebene Syntax bezieht sich auf den Treiber 'dvilzr'. Bei Verwendung anderer Druckertreiber kann eine Änderung des Programmcodes von 'WriteZVtoPS' erforderlich sein, da nicht alle Treiber den `\special`–Befehl verarbeiten können.

5.9.3 Ausgabe von kumulativen Verteilungsfunktionen als PostScript-Graphik

```
PROCEDURE WriteVertFktToPS(X       : ZufallsVar;
                           FileName: ARRAY OF CHAR;
                           Caption : ARRAY OF CHAR;
                           Height,
                           Width   : Real);
```

Die Prozedur 'WriteVertFktToPS' erzeugt zwei Dateien. Eine der Dateien hat die Endung '.ps' und ist eine PostScript-Datei, mit der die kumulative Verteilungsfunktion der Zufallsvariablen X auf einem PostScript-fähigen Drucker ausgegeben werden kann. Die andere Datei besitzt die Endung '.tex' und wird benutzt, um die PostScript-Datei in ein LATEX-Dokument einzubinden. Diese Prozedur wird ähnlich wie die Prozedur 'WriteZVtoPS' (Abschnitt 5.9.2) benutzt.

5.10 Das Modul 'RndVar2'

Mit dem Modul 'RndVar2' werden Prozeduren zur Zuweisung von Zufallsvariablen, Berechnung von Momenten, Normierung von Verteilungen, Spiegelung von Verteilungen am Ursprung sowie Minimum- und Maximum-Operationen bereitgestellt. Die Prozeduren, die als Ergebnis Verteilungen berechnen, sorgen zuvor für die Nullsetzung der Verteilungen der Ergebnis-Zufallsvariablen (siehe 'Setze0Vert' (Abschn. 5.7.5), 'S0V' (Abschn. 5.7.6), und die Prozeduren 'InitDskrZV1D', 'InitDskrZVMD', 'InitStetZV' sowie 'InitGemZV' (Abschnitt 5.7.7 ff.)).

5.10.1 Zuweisen von Zufallsvariablen

```
PROCEDURE ZuweisenZV(X,Y: ZufallsVar);
```

Die Zufallsvariable Y erhält die gleiche Verteilung wie die Zufallsvariable X. Voraussetzung ist, daß der Wertebereich der Zufallsvariablen X Teilmenge des Wertebereiches der Zufallsvariablen Y ist.

5.10.2 Berechnung der Norm

```
PROCEDURE Moment0(X: ZufallsVar): Real;
```

Diese Funktion liefert als Ergebnis die Norm der Verteilung einer beliebigen Zufallsvariablen X.

5.10.3 Berechnung des Erwartungswertes einer eindimensionalen Verteilung

```
PROCEDURE Erwartung1D(X: ZufallsVar): Real;
```

Die Funktion berechnet für Zufallsvariablen mit eindimensionalem Wertebereich deren Erwartungswert, also das 1. Moment.

5.10.4 Berechnung der Erwartungswerte von Randverteilungen mehrdimensionaler Zufallsvariablen

```
PROCEDURE Erwartung(X: ZufallsVar; VAR Erw: RTupel);
```

Sei X eine d-dimensionale Zufallsvariable. Die Prozedur gibt in der Variablen 'Erw' als Resultat die Erwartungswerte der d Randverteilungen von X zurück. 'Erw' ist ein Feld mit d Komponenten.

5.10.5 Berechnung höherer Momente eindimensionaler Verteilungen

```
PROCEDURE Momente1D(X: ZufallsVar; VAR MVektor: ARRAY OF Real;
                                        MStart: CARDINAL);
```

Die Prozedur berechnet Momente einer eindimensionalen Zufallsvariablen X. Besitzt das Feld 'MVektor' k Komponenten und der Wert von 'MStart' beträgt j, so werden die Momente $E[X^j], \ldots, E[X^k]$ berechnet und in 'MVektor' abgelegt.

5.10.6 Berechnung von Varianzen

```
PROCEDURE Var(X: ZufallsVar): Real;
```

Die Funktion berechnet die Varianz der Zufallsvariablen X.

5.10.7 Berechnung von Covarianzen

```
PROCEDURE Covarianz(X: ZufallsVar): Real;
```

Diese Funktion berechnet die Covarianz zweier eindimensionaler Zufallsvariablen, deren gemeinsame Verteilung der Verteilung von X entspricht. Deshalb muß X eine zweidimensionale Zufallsvariable sein.

5.10.8 Korrigieren der Norm einer Verteilung

```
PROCEDURE WMasseKorrektur(X: ZufallsVar; faktor: Real);
```

Alle Wahrscheinlichkeiten in der Verteilung der Zufallsvariablen X sowie die 1. Momente im Falle von stetigen Verteilungen werden um den Multiplikator 'faktor' verändert. 'WMasseKorrektur' wird von der Prozedur 'NormiereZV' zur Normierung von Verteilungen benutzt.

In der folgenden Gleichung sei $r =$'faktor'. F_X sei die Verteilungsfunktion von X vor, F'_X die Verteilungsfunktion nach der Anwendung der Prozedur 'WMasseKorrektur'. Dann bewirkt die Anwendung der Prozedur:

$$F'_X(x) = r \cdot F_X(x), \quad \text{für alle } x \in I\!R\ .$$

5.10.9 Normieren von Verteilungen

```
PROCEDURE NormiereZV(X: ZufallsVar);
```

Die Verteilung von X wird durch Anwendung der Prozedur 'WMasseKorrektur' normiert. Nach Anwendung der Prozedur gilt annähernd:

$$\lim_{x \to \infty} F_X(x) = 1$$

Abweichungen können in ungünstigen Fällen durch Rundungsfehler entstehen, wenn nämlich die einzelnen Wahrscheinlichkeiten von Elementen des Wertebereiches von X um Größenordnungen verschieden sind. Dies kann z.B. bei Poisson-verteilten Zufallsvariablen der Fall sein.

Durch Stellenextinktion bedingt kann es jedoch dazu kommen, daß durch die Anwendung der Prozedur 'NormiereZV' „negative Wahrscheinlichkeiten“ erzeugt werden. Für diesen Fall exisiert eine weitere Prozedur zur Normierung von Verteilungen:

```
PROCEDURE NormiereZVRundFehlKorr(X: ZufallsVar);
```

Im Unterschied zur oben beschriebenen Prozedur werden „negative Wahrscheinlichkeiten“ auf den Wert Null gesetzt. Treten Werte auf, die kleiner als -10^{-13} sind, so wird davon ausgegangen, daß diese meist auf Programmierfehler seitens des Modellierers zurückzuführen sind. Sind die „negativen Wahrscheinlichkeiten“ größer als -10^{-13}, sind diese offenbar durch Rechenungenauigkeiten entstanden. In beiden Fällen wird eine jeweils passende Warnung ausgegeben und die Programmausführung fortgesetzt. In Fällen, in denen der Programmierer signifikante Rundungsfehler vermutet, kann er die Prozedur 'NormiereZVRundFehlKorr' verwenden.

5.10.10 Normieren bei Überschreiten einer Toleranzgrenze

```
PROCEDURE NormiereZVBeiAbweichung(X: ZufallsVar; MaxAbw: Real);
```

Diese Prozedur nimmt eine Normierung der Verteilung von X vor, falls die Norm der Verteilung absolut um mehr als 'MaxAbw' vom Sollwert 1 abweicht.

5.10.11 Minimum einer Zufallsvariablen und einer Zahl

```
PROCEDURE MinZV1D(X: ZufallsVar; k: INTEGER; Y: ZufallsVar;
                  Normiere: BOOLEAN);
```

Voraussetzung für die Ausführung dieser Prozedur ist, daß X und Y eine eindimensionale diskrete oder gemischte Verteilung besitzen und k ein Element des diskreten Wertebereichs von Y ist. Wenn Y eine diskret unendliche Zufallsvariable ist, so muß k im VME-0-Bereich von Y liegen. Die Prozedur berechnet

$$Y := \text{ Min}(X, k) \quad .$$

Alle Elemente der Grundmenge von X, die größer sind als k, werden von Y auf k abgebildet.

Es ist zulässig, daß die aktuellen Parameter von 'X' und 'Y' gleich sind, also

$$X := \text{ Min}(X, k)$$

kann direkt realisiert werden.

Der Parameter 'Normiere' legt fest, auf welche Weise die Wahrscheinlichkeit $P\{Y = k\}$ ermittelt wird. Falls 'Normiere' = FALSE gewählt wird, gilt:

$$P\{Y = k\} := \int_k^\infty dF_X(x),$$

anderenfalls wird gleichzeitig eine Normierung der Verteilung von Y vorgenommen und

$$P\{Y = k\} := 1 - \lim_{\varepsilon \searrow 0} \int_{-\infty}^{k-\varepsilon} dF_X(x)$$

berechnet.

5.10.12 Maximum einer Zufallsvariablen und einer Zahl

```
PROCEDURE MaxZV1D(X: ZufallsVar; k: INTEGER; Y: ZufallsVar;
                  Normiere: BOOLEAN);
```

Die Prozedur 'MaxZV1D' arbeitet wie die Prozedur 'MinZV1D' (Abschn. 5.10.11), realisiert jedoch die Operation

$$Y := \text{ Max}(X, k).$$

5.10.13 Minimum und Maximum zweier stetiger Zufallsvariablen

```
PROCEDURE MinStetZV(X,Y: ZufallsVar; M: ZufallsVar);
```

Diese Prozedur berechnet das Minimum M (bzw. eine 101–Approximation des Minimums) zweier gegebener unabhängiger stetiger Zufallsvariablen X und Y. Voraussetzung für die Benutzung dieser Prozedur ist, daß bei der Initialisierung von X, Y und M mittels der Prozedur `InitStetZV` die Grenzen des DME-0-Bereiches und die Intervallbreite δ auf diesem Bereich jeweils identisch gewählt werden, d.h. die bei der Beschreibung von `InitStetZV` angegebenen Parameter a,b und delt.

```
PROCEDURE MaxStetZV(X,Y: ZufallsVar; M: ZufallsVar);
```

Diese Prozedur arbeitet analog zu `MinStetZV`, liefert jedoch in M das Maximum von X und Y zurück.

Die theoretischen Grundlagen, insbesondere die Herleitung der 101–Dichten des Minimums und des Maximums findet man in [Sa98].

5.10.14 Spiegelung von Verteilungen am Nullpunkt

```
PROCEDURE MinusZV(X, Y: ZufallsVar);
```

Die Prozedur 'MinusZV' realisiert die Zuweisungsoperation

$$Y := -X \ ,$$

so daß für alle $x \in I\!R$ gilt:

$$P\{Y \leq x\} = P\{X \geq -x\} \ .$$

Die Zufallsvariable Y muß dabei einen geeignet strukturierten Wertebereich aufweisen. Die aktuellen Parameter von 'X' und 'Y' dürfen gleich sein, d.h.

$$X := -X,$$

kann direkt realisiert werden.

5.10.15 Totale Wahrscheinlichkeit mit Zufallsvariable

```
PROCEDURE TotalZV(XwennYgleichy : ARRAY OF ZufallsVar;
                  Y,X           : ZufallsVar;
                  Anzahl        : INTEGER);
```

Die Prozedur 'TotalZV' berechnet die Verteilung einer Zufallsvariablen X unter Anwendung des Satzes von der totalen Wahrscheinlichkeit aus den Verteilungen der Zufallsvariablen Y und $X \mid Y$:

$$P\{X = x\} := \sum_{y=a}^{b} P\{X = x \mid Y = y\} \cdot P\{Y = y\}, \quad \text{für alle } x \in \Omega_X \ .$$

Die Bedeutung der Parameter 'X' und 'Y' ergibt sich aus der obenstehenden Definition.

Für jedes $y \in \Omega_Y$ existiert eine bedingte Zufallsvariable $X \mid Y = y$. Alle diese Zufallsvariablen werden in einem Analyseprogramm in einem 'ARRAY [a..b] OF ZufallsVar' gespeichert, wobei der Wert von $b-a+1$ der Mächtigkeit der Menge Ω_Y entspricht. Dem Parameter 'XwennYgleichy' wird der Name dieses Feldes übergeben. Der Parameter 'Anzahl' gibt die Anzahl der bedingten Zufallsvariablen – damit auch die Mächtigkeit von Ω_Y – an. Dieser zusätzliche Parameter ist notwendig, um mögliche Fehler bei der Benutzung der Prozedur erkennen zu können.

Ein Aufruf der Prozedur kann folgende Gestalt haben:

```
MODULE Test;
    :
    :
FROM SYSTEM IMPORT ADR;
    :
    :
VAR XwennY:ARRAY [1..10] OF ZufallsVar;
    Y,X:ZufallsVar;

BEGIN
      :
      :
   (* Die Zufallsvariablen XwennY[1],...,XwennY[10],Y seien definiert *)
   TotalZV(XwennY,Y,X,10);
      :
      :
END Test.
```

5.10.16 Totale Wahrscheinlichkeit mit Häufigkeitsverteilung

```
PROCEDURE Total(XwennYa,XwennYb : ZufallsVar;
                VAR pYa,pYb     : Real;
                X               : ZufallsVar);
```

Eine weitere Möglichkeit, Verteilungen von Zufallsvariablen unter Anwendung des Satzes von der totalen Wahrscheinlichkeit zu berechnen, bietet die Prozedur 'Total'.

$$P\{X = x\} := \sum_{y=a}^{b} P\{Y = y\} \cdot P\{X = x \mid Y = y\}, \quad \text{für alle } x \in \Omega_X$$

Im Unterschied zu der oben beschriebenen Prozedur 'TotalZV' besteht hier die Möglichkeit, mit Häufigkeitsverteilungen für 'y' zu rechnen, ohne dafür explizit eine Zufallsvariable zu definieren. Diese Häufigkeiten sind in einem Feld 'pY' abgelegt, das als 'ARRAY [...] OF Real' deklariert ist. Außerdem müssen die bedingten Zufallsvariablen $X \mid Y = y,\ y \in [a : b]$, im Analyseprogramm in einem 'ARRAY [a..b] OF ZufallsVar' vorliegen. Der aktuelle Parameter von 'ZVa' ist ein Zeiger auf das erste Element im Feld 'ARRAY [a..b] OF ZufallsVar', welches bei der Summation berücksichtigt werden soll. Der aktuelle Parameter von 'ZVb' zeigt auf das letzte zu berücksichtigende Feldelement.

Die Parameter 'pYa' und 'pYb' dienen der Bestimmung der Anfangs- und der Endadresse des Bereichs von 'pY', der benutzt wird.

Ein Prozeduraufruf kann also, bei entsprechenden Variablendeklarationen, wie folgt aussehen:

```
Total(Xbedingt[i],Xbedingt[j],pY[i],pY[j],X);
```

Ein weiterer Unterschied zur oben beschriebenen Prozedur 'TotalZV' besteht darin, daß nur überprüft werden kann, ob die Anzahl der bedingten Zufallsvariablen gleich der Anzahl der Häufigkeiten ist. Das hat zur Folge, daß die Prozedur 'Total' auch mit

```
Total(Xbedingt[i],Xbedingt[j],pY[i + k],pY[j + k],X);
```

aufgerufen werden kann, ohne daß eine Fehlermeldung ausgegeben wird. Die Gefahr von Adressierungsfehlern ist dabei jedoch groß.

5.10.17 Verschieben des Bereichs der DME-0- bzw. VME-0-Approximation einer Verteilung

```
PROCEDURE VerschiebeDME0Bereich(X: ZufallsVar; a,b: Real);
```

In den Abschnitten 3.1 und 3.2 wird beschrieben, wie die Verteilung einer Zufallsvariablen gemäß DME-101- bzw. VME-101-Approximation wird mit Verteilungen maximaler Entropie der Ordnungen 0 und 1 dargestellt. Die Prozedur 'VerschiebeDME0Bereich' ändert den Bereich, auf dem eine Approximation durch DME-0 bzw. VME-0 erfolgt. Die Parameter 'a' und 'b' legen den Bereich fest, auf dem approximiert werden soll. Ist die Verschiebung zulässig, so, werden die Komponenten 'aStet' und 'bStet' bzw. 'aDiskr' und 'bDiskr' des Parameters 'X' entsprechend 'a' und 'b' geändert. Wenn bei stetigen Zufallsvariablen die Werte der Parameter 'a' und 'b' nicht Vielfache der Intervallbreite δ sind, so erfolgt ein Abbruch mit einer Fehlermeldung.

5.11 Das Modul 'RndVar3'

Das Modul 'RndVar3' enthält Prozeduren, mit denen Zufallsvariablen häufig verwendete Verteilungen zugewiesen werden können. Vor dem Aufruf der Prozeduren müssen die als Parameter übergebenen Zufallsvariablen bereits initialisiert sein (Prozeduren 'InitDskrZV1D', 'InitStetZV', usw.). Das Nullsetzen ihrer Verteilung ist vor dem Aufruf einer Prozedur aus diesem Modul nicht notwendig.

Zufallsvariablen mit einer Grundmenge der Mächtigkeit 1 („konstante“ Zufallsvariablen) werden mit der Prozedur 'EinpunktVert' aus dem Modul 'RVBasics' erzeugt (vgl. Abschnitt 5.7.17). Umfaßt die Grundmenge einer diskreten Zufallsvariablen mehr als ein Element und soll (anfänglich) einem Element die Wahrscheinlichkeit eins zugeordnet werden, so kann dies unmittelbar nach der Initialisierung oder Nullsetzung mit der Prozedur 'Pgleich' aus dem Modul 'RndVar0' geschehen (vgl. Abschnitt 5.8.5).

Exportierte Prozeduren:

5.11.1 Diskrete Gleichverteilung

```
PROCEDURE DiskrGleichVert(X: ZufallsVar; a,b : INTEGER);
```

Die diskrete Zufallsvariable X erhält eine Gleichverteilung über $[a:b]$.

$$P\{X=x\} := \frac{1}{b-a+1},\ a \le x \le b$$

5.11.2 Mehrdimensionale diskrete Gleichverteilung

```
PROCEDURE DiskrGleichVertMD(X: ZufallsVar; a,b : ITupel);
```

Die diskrete Zufallsvariable X mit der Dimension d erhält eine Gleichverteilung über $[a_1 : b_1] \times \cdots \times [a_d : b_d]$. Bevor die Prozedur aufgerufen wird, müssen die INTEGER-Tupel 'a' und 'b', welche die Ober- und Untergrenzen der Intervalle aufnehmen, durch die Prozedur 'BildeITupel$d$' erzeugt werden.

Beispiel:

```
     :
BildeITupel4(a,1,1,1,1);           (* Erzeuge eine vierdimensionale    *)
BildeITupel4(b,5,5,5,5);           (* Zufallsvariable ueber dem Quader *)
InitDskrZVMD(X,' X', ADR(pX),a,b); (* [1:5] x [1:5] x [1:5] x [1:5]    *)
     :
     :
BildeITupel4(a,1,2,3,2);           (* X erhaelt eine Gleichverteilung  *)
BildeITupel4(b,3,4,5,3);           (* ueber dem Quader                 *)
DskrGleichVertMD(X,a,b);           (* [1:3] x [2:4] x [3:5] x [2:3]    *)
     :
```

Jedem Punkt im Inneren eines so festgelegten Quaders wird als Wahrscheinlichkeit der Kehrwert des Volumens dieses Quaders zugewiesen; den übrigen Punkten wird als Wahrscheinlichkeit der Wert 'Null' zugewiesen:

$$P\{X = x\} := \begin{cases} \frac{1}{\prod_{i=1}^{d}(b_i - a_i + 1)} & , a_i \leq x \leq b_i, \quad \forall i \in [1:d], \\ 0 & , \text{sonst} . \end{cases}$$

5.11.3 Bernoulli-Verteilung

`PROCEDURE BernoulliVert(X: ZufallsVar; p: Real);`

Die Zufallsvariable X erhält eine Bernoulli-Verteilung mit Parameter p. Es gelten die Wahrscheinlichkeiten

$$P\{X = 1\} := p$$

und

$$P\{X = 0\} := 1 - p .$$

5.11.4 Binomialverteilung

`PROCEDURE BinomialVert(X: ZufallsVar; n: INTEGER; p: Real);`

Die Zufallsvariable X erhält eine Binomialverteilung mit den Parametern n und p, so daß für die Wahrscheinlichkeiten gilt:

$$P\{X = k\} := \binom{n}{k} p^k q^{n-k}, \quad k = 0, 1, 2, \ldots, n ,$$

mit $q = 1 - p$.

5.11.5 Geometrische Verteilung

`PROCEDURE GeometrVert(X: ZufallsVar; p: Real);`

Falls der Prozedur 'GeometrVert' eine Zufallsvariable des Typs 'Diskr101' übergeben wird, werden den Komponenten 'M0rechts' und 'M1rechts' die Werte der Norm und des ersten Moments der geometrischen Verteilung zugewiesen; die Komponenten 'aDiskr' und 'bDiskr' erhalten die Werte '0' bzw. '-1', d.h. der DME-0-Bereich ist leer.

Ist X jedoch eine endliche diskrete Zufallsvariable, so erhält X annähernd eine geometrische Verteilung mit Parameter p. Die Wahrscheinlichkeiten $P\{X = k\}$ werden im Intervall $[0 : b-1]$ festgelegt mit

$$P\{X = k\} := q^k p.$$

Die noch verbleibende Wahrscheinlichkeit $P\{X_{\text{geometrisch}} \geq b\}$ wird auf das größte Element des Wertebereichs von X, nämlich b, konzentriert, falls die Zufallsvariable vom Typ 'DiskrEndl' ist:

$$P\{X = b\} := 1 - \sum_{k=0}^{b-1} P\{X = k\} = P\{X_{\text{geometrisch}} \geq b\} .$$

'b' entspricht der Verbundtyp-Komponente 'X↑ .bDskr'.

Diese Art der Approximation hat den Vorteil, daß bei der Modellierung von Ankunftsprozessen mit geometrisch verteilten Ankünften und endlicher Puffergröße keine Genauigkeitsverluste auftreten. Die Modellierung wird sogar dadurch erleichtert, daß die Anwendung einer Minimum-Operation entfällt.

5.11.6 Poissonverteilung

`PROCEDURE PoissonVert(X: ZufallsVar; lambda: Real);`

Die Zufallsvariable X erhält annähernd eine Poissonverteilung mit Parameter λ, falls die Zufallsvariable X vom Typ 'DiskrEndl' ist. Die Approximation erfolgt wie bei der Prozedur 'GeometrVert', also durch Konzentration der Wahrscheinlichkeit $P\{X_{\text{Poisson}} \geq b\}$ auf b, das größte Element des Wertebereichs von X.

$$P\{X = k\} := e^{-\lambda} \frac{\lambda^k}{k!}$$

$$P\{X = b\} := 1 - \sum_{k=0}^{b-1} P\{X = k\} = P\{X_{\text{Poisson}} \geq b\}$$

Falls der Prozedur eine 'PoissonVert' eine Zufallsvariable des Typs 'Diskr101' übergeben wird, so erfolgt auf dem Bereich $[bDiskr + 1, \infty)$ eine Approximation der Poissonverteilung durch eine geometrische Verteilung.

5.11.7 Stetige Gleichverteilung

```
PROCEDURE StetGleichVert(X: ZufallsVar; a,b: Real);
```

Die Zufallsvariable X erhält eine Gleichverteilung über dem Intervall $[a,b]$, wobei $a < b$ Vielfache von δ sein müssen, da sonst mit einer Fehlermeldung abgebrochen wird. Die Dichte ist damit

$$f_X(x) := \frac{1}{b-a}, \quad a \leq x \leq b\,.$$

5.11.8 Exponentialverteilung

```
PROCEDURE ExpVert(X: ZufallsVar; lambda: Real);
```

Die Zufallsvariable X erhält eine negativ exponentielle Verteilung mit dem Parameter λ (='lambda') und der Dichte

$$f_X(x) := \frac{1}{\lambda}e^{-\lambda x}, \quad x \geq 0\ .$$

Der DME-0-Bereich von X ist leer, auch wenn bei der Initialisierung von 'X' mit 'InitStetZV' ein nichtleerer DME-0-Bereich festgelegt wurde.

5.11.9 Hyperexponentialverteilung

```
PROCEDURE HExpVert(X: ZufallsVar; lambda1,lambda2,p: Real);
```

Die Zufallsvariable X erhält eine hyperexponentielle Verteilung mit den Parametern λ_1, λ_2 und p. Die Verteilung wird durch gewichtete Summierung zweier Exponentialverteilungen erzeugt und stückweise mit Dichten maximaler Entropie approximiert:

$$f_X(x) :\approx \frac{p}{\lambda_1}e^{-\lambda_1 x} + \frac{(1-p)}{\lambda_2}e^{-\lambda_2 x}, \quad x \geq 0\ .$$

5.11.10 Erlang-k-Verteilung

```
PROCEDURE ErlangKVert(X: ZufallsVar; k: INTEGER; lambda: Real);
```

Die Zufallsvariable X erhält eine Erlang-k-Verteilung mit Parameter λ. Für k ist ein Wert von höchstens 60 zulässig. Sollte für k ein größerer Wert gewählt werden, so wird *keine* Warnung oder Fehlermeldung zurückgegeben; darauf wurde aus Effizienzgründen verzichtet. In der Praxis wird dies jedoch keine Einschränkung darstellen.

An dieser Stelle sei außerdem darauf hingewiesen, daß die interne Darstellung einer Erlang-1-Verteilung *nicht* identisch mit der internen Darstellung einer Exponentialverteilung ist. Bei einer Erlang-k-Verteilung, und damit auch bei einer Erlang-1-Verteilung, wird der linke DME-1-Bereich gleich Null gesetzt, der DME-0– und der rechte DME-1-Bereich entsprechend mit Werten besetzt. Bei einer Exponentialverteilung ist jedoch auch der DME-0-Bereich leer; nur der rechte DME-1-Bereich wird benutzt.

5.11.11 Normalverteilung

`PROCEDURE NormalVert(X: ZufallsVar; mue, sigma : Real);`

X erhält als Verteilung eine Approximation der Normalverteilung mit den Parametern μ und σ und der Dichte

$$f_X(x) :\approx \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} .$$

5.12 Das Modul 'RndVar4'

Das Modul 'RndVar4' enthält Prozeduren, die die Faltung von Verteilungen realisieren. Es wird dabei auf Prozeduren zurückgegriffen, die von den zusätzlichen Modulen 'RndVar4a' bis 'RndVar4h' exportiert werden. Die Aufgliederung auf mehrere Module mußte vorgenommen werden, da der Umfang der Programme bestehende Beschränkungen einiger MODULA-Compiler bezüglich der Größe eines Moduls überschreitet. **Wichtig:** Um die unten aufgeführten Prozeduren benutzen zu können, brauchen die Zusatzmodule **nicht** in das Analyseprogramm eingebunden zu werden.

Bei allen Prozeduren dieses Moduls kann der Ausgabeparameter 'Y' mit einem der Eingabeparameter identisch sein.

Exportierte Variablen:

5.12.1 Benutzen der schnellen Fourier-Transformation

Mit der Variable 'SelectFFT' vom Typ BOOLEAN wird festgelegt, ob die Prozeduren 'FalteZV', 'FalteDskrZVBeschr', 'FalteZVFolge' und 'FalteDskrZVFolgeBeschr' bei der Faltung diskreter eindimensionaler Wertebereichen die schnelle Fourier-Transformation (Fast Fourier Transform, FFT) verwenden. Der Rechenaufwand für die Faltung hat dann die Größenordnung $O(n \cdot \log n)$. Wenn die Variable 'SelectFFT' den Wert FALSE hat, wird die 'Schulmethode' mit Aufwand $O(n^2)$ benutzt.

Durch Verwendung der FFT wird bei großen Wertebereichen die Ausführungszeit der Operation verkürzt. Sie hat allerdings den Nachteil, daß sie mehr Speicherplatz benötigt als die „Schulmethode“. Ab welcher Wertebereichsgröße die Verwendung der FFT vorzuziehen ist, hängt vom benutzten Compiler und von der verwendeten Rechenanlage einschließlich deren Konfiguration ab; es sollte daher vor Benutzung dieser Option eine Reihe von Vergleichstests durchgeführt werden.

Exportierte Prozeduren:

5.12.2 Beliebige Verteilungen falten

`PROCEDURE FalteZV(X1,X2,Y: ZufallsVar);`

Die Prozedur berechnet die Verteilung von Y durch Faltung der Verteilungen der unabhängigen Zufallsvariablen X_1 und X_2. Wenn die Zufallsvariable X_1 oder X_2 über einen stetigen Wertebereich verfügt, ist die Verteilung von Y eine Approximation des korrekten Ergebnisses.

Voraussetzung für die Benutzung der Prozedur mit endlichen diskreten Zufallsvariablen ist, daß jede Summe aus jeweils einem Element aus dem diskreten Wertebereich von X_1 und dem diskreten Wertebereich von X_2 im diskreten Wertebereich von Y liegt. Falls eine der Zufallsvariablen X_1 oder X_2 über einen stetigen Wertebereich verfügt, muß auch die Zufallsvariable Y einen stetigen Wertebereich besitzen. Alle stetigen Wertebereiche von X_1, X_2 und Y müssen, falls vorhanden, mit der gleichen Intervallbreite δ aufgeteilt sein.

5.12.3 Diskrete Verteilungen falten mit Einschränkung des Ergebnisraums

`PROCEDURE FalteDskrZVBeschr(X1,X2,Y: ZufallsVar);`

Diese Prozedur, die nur auf Zufallsvariablen mit diskretem Wertebereich angewendet werden kann, führt eine Faltung der Verteilungen der unabhängigen Zufallsvariablen X_1 und X_2 durch. Die Zufallsvariable Y erhält die Verteilung, die Ergebnis der Faltungsoperation ist, eingeschränkt auf den Wertebereich der Zufallsvariablen Y. Die Prozedur realisiert also die folgende Operation:

$$Y := \ \mathrm{Max}(\ \mathrm{Min}(X_1 \oplus X_2, b), a),$$

wobei $[a : b]$ der diskrete Wertebereich von 'Y' ist.

5.12.4 Addition von Zufallsvariablen und Konstanten

`PROCEDURE ZVplusKonst(X : ZufallsVar; k : INTEGER; Y : ZufallsVar);`

Die Verteilungen der Zufallsvariablen X und einer konstanten Zufallsvariablen X' mit $P\{X' = k\} = 1$ werden gefaltet. Der Wertebereich von X muß eindimensional sein. Das Ergebnis wird der Verteilung von Y zugewiesen. Bei stetiger Verteilung der Zufallsvariablen X muß k ein ganzzahliges Vielfaches der zur Approximation der Verteilung von X verwendeten Intervallbreite δ sein.

5.12.5 Eine Folge beliebiger Verteilungen falten

```
PROCEDURE FalteZVFolge(VAR Xa,Xb: ZufallsVar; Y: ZufallsVar);
```

Diese Prozedur kann verwendet werden, wenn eine Folge von Verteilungen unabhängiger Zufallsvariablen zu falten ist, die im Analyseprogramm als 'ARRAY[...] OF ZufallsVar' deklariert wurden und in diesem Feld aufeinanderfolgende Indizes besitzen.

Als Parameter 'Xa' wird die Zufallsvariable mit dem kleinsten Index in der Folge übergeben, als Parameter 'Xb' diejenige mit dem größten Index. Der Wertebereich der als Parameter 'Y' verwendeten Zufallsvariablen muß so groß gewählt sein, daß das Ergebnis der Operation der Variablen 'Y' als Verteilung zugewiesen werden kann.

Ein Aufruf der Prozedur 'FalteZVFolge' hat beispielsweise die Form:

```
FalteZVFolge(X[1],X[K],Y);
```

Dieser Aufruf entspricht der Operation

$$Y := \bigoplus_{k=1}^{K} X_k \ ,$$

sofern der Wertebereich der Zufallsvariablen 'Y' genügend groß ist.

5.12.6 Eine Folge diskreter Verteilungen mit beschränktem Ergebnisraum falten

```
PROCEDURE FalteDskrZVFolgeBeschr(VAR Xa,Xb: ZufallsVar; Y: ZufallsVar);
```

Diese Prozedur kann verwendet werden, wenn eine Folge von Zufallsvariablen mit eindimensionalen, diskreten Verteilungen zu falten ist. Die Zufallsvariablen werden im Analyseprogramm als 'ARRAY[...] OF ZufallsVar' deklariert. Die Parameterübergabe erfolgt wie bei der Prozedur 'FalteZVFolge' (Abschnitt 5.12.5). Zur Durchführung der einzelnen Faltungsoperationen wird die Prozedur 'FalteDskrZVBeschr' herangezogen. Mit Wahrscheinlichkeiten, die nicht Elementen des VME-0-Bereichs der Zufallsvariablen Y zuzuordnen sind, wird wie dort verfahren (vgl. Abschnitt 5.12.3).

5.13 Das Modul 'RndVar5'

Das Modul 'RndVar5' enthält Prozeduren, die verwendet werden können, wenn von gemischten Zufallsvariablen entweder nur deren Pseudo-Verteilungen über dem diskreten oder dem stetigen Wertebereich für weitere Berechnungen benötigt werden.

5.13.1 Nur stetigen Wertebereich betrachten

```
PROCEDURE KopiereStetig(X,Y: ZufallsVar);
```

Die Zufallsvariable Y erhält die Pseudo-Verteilung der Zufallsvariablen X über deren stetigem Wertebereich. Es erfolgt keine Normierung der Verteilung der Zufallsvariablen Y.

5.13.2 Nur diskreten Wertebereich betrachten

```
PROCEDURE KopiereDiskret(X,Y: ZufallsVar);
```

Die Zufallsvariable Y erhält die Pseudo-Verteilung der Zufallsvariablen X über deren diskretem Wertebereich. Eine Normierung der Zufallsvariablen Y wird nicht vorgenommen.

5.14 Das Modul 'RndVar6'

Das Modul 'RndVar6' stellt Prozeduren zur Verfügung, die bei der Überprüfung der Konvergenz von Iterationsergebnissen hilfreich sind.

Exportierte Typen:

5.14.1 Typ für die Speicherung von Iterationsergebnissen

```
TYPE Wertspeicher = RECORD
      Wert,
      AbsAenderung,
      ProzAenderung: Real;
END;
```

Bei Verwendung der Prozeduren 'InitWertspeicher' und 'NeuerWert' haben die Komponenten dieses Verbundtyps die folgende Bedeutung:

Wert: Enthält den Wert, der zuletzt mit Hilfe der Prozedur 'NeuerWert' berechnet wurde

AbsAenderung: Enthält den Absolutbetrag der Differenz zwischen dem letzten und dem vorletzten Wert, der mit Hilfe der Prozedur 'NeuerWert' berechnet wurde

ProzAenderung: Enthält die prozentuale Änderung vom vorletzten zum letzten berechneten Wert

Exportierte Prozeduren:

5.14.2 Speicher für Ergebnisse initialisieren

```
PROCEDURE InitWertspeicher(VAR WSp : Wertspeicher);
```

Die Aufgabe dieser Prozedur ist, alle Komponenten der Verbundvariablen 'WSp', nämlich 'Wert', 'AbsAenderung' und 'ProzAenderung', mit dem Wert '0.0' zu initialisieren.

5.14.3 Neues Ergebnis speichern

```
PROCEDURE NeuerWert(VAR WSp: Wertspeicher; NWert: Real);
```

Die Prozedur berechnet zunächst die absolute und die prozentuale Abweichung zwischen dem bisher in 'WSp' gespeicherten Wert und 'NWert' und ersetzt anschließend 'Wsp.Wert' durch 'NWert'.

5.15 Das Modul 'RndVar7'

Das Modul 'RndVar7' stellt zwei Prozeduren zur Verfügung, die zur Berechnung von Zustandswahrscheinlichkeiten von Markovketten benutzt werden können.

Exportierte Prozeduren:

5.15.1 Neue Zustandswahrscheinlichkeiten berechnen

```
PROCEDURE Uebergang (ZustWs,UebergWs: ZufallsVar);
```

Die Zufallsvariable 'ZustWs' enthalte die Verteilung der Zustandswahrscheinlichkeiten einer Markovkette zu einem beliebigen Zeitpunkt t. Dann berechnet die Prozedur 'Uebergang' mittels der durch 'UebergWs' festgelegten n-Schritt-Übergangswahrscheinlichkeiten die neue Verteilung der Zustandswahrscheinlichkeiten 'ZustWs' für den Zeitpunkt $t + n$.

Der Parameter 'UebergWs' ist zwar ebenfalls vom Typ 'ZufallsVar', er bezeichnet aber in diesem Falle keine Zufallsvariable mit zugehöriger Verteilung, sondern wird zur Speicherung der quadratischen Übergangsmatrix benutzt.

Die Variable 'ZustWs' muß eine diskrete Zufallsvariable sein. Deren Wertebereich sei als das Intervall $[a : b]$ gegeben. Das zur Variable 'UebergWs' gehörende Verteilungsfeld muß als ein Feld mit Indexmenge $[a : b] \times [a : b]$ deklariert worden sein (ARRAY [a..b], [a..b] OF Real). Dabei gilt

$$\text{UebergWs}[i, j] = P\{\text{ZustWs}^{(t+n)} = j \mid \text{ZustWs}^{(t)} = i\}.$$

Die Prozedur 'Uebergang' kann auch zustandsbeschreibende Zufallsvariablen mit mehrdimensionalen Wertebereichen verarbeiten. Deren Verteilungsfelder werden dann aufgrund der Darstellung mehrdimensionaler Felder im Arbeitsspeicher als eindimensional betrachtet.

Ein Beispiel für die Anwendung der Prozedur findet sich in Abschnitt 6.9.

5.15.2 Besondere Zustandsübergänge bei zustandsbeschreibenden Vektoren

```
PROCEDURE UebergangSparse (ZustWs,UebergWs: ZufallsVar; k: INTEGER);
```

Diese Prozedur berechnet, wie auch die Prozedur 'Uebergang', aus der bisherigen Verteilung der Zufallsvariablen 'ZustWs' zu einem Zeitpunkt t und den durch 'UebergWs' gegebenen n-Schritt-Übergangswahrscheinlichkeiten die neue Verteilung für 'ZustWs' zum Zeitpunkt $t + n$, wobei die Zustände der untersuchten Markovkette durch Vektoren beschrieben werden. Die Prozedur kann jedoch nur dann angewendet werden, wenn alle Zustandsübergänge, die mit positiven Wahrscheinlichkeiten auftreten, von einem Zustand

$$\vec{i} = (i_1, \ldots, i_{k-1}, i_k, i_{k+1}, \ldots, i_d)$$

zu einem Zustand

$$\vec{i'} = (i_1, \ldots, i_{k-1}, j, i_{k+1}, \ldots, i_d)$$

führen, wobei k als Parameter der Prozedur gegeben und d als die Dimension der Zufallsvariablen 'ZustWs' festlegt ist. Es darf sich also bei einem Zustandsübergang nur eine Komponente des Zustandsvektors ändern.

Die Variable 'ZustWs' muß eine Zufallsvariable mit diskretem Wertebereich $[a_1 : b_1] \times \ldots \times [a_d : b_d]$ sein, das zur Variable 'ZustWs' gehörende Verteilungsfeld muß mit der Indexmenge $[a_1 : b_1] \times \ldots \times [a_d : b_d] \times [a_k : b_k]$ deklariert worden sein. Dabei ist festgelegt:

$$\mathrm{UebergWs}[\vec{i}, j] = P\{\mathrm{ZustWs}^{(t+1)} = (i_1, \ldots, i_{k-1}, j, i_{k+1}, \ldots, i_n) \mid \mathrm{ZustWs}^{(t)} = \vec{i}\}.$$

Auch zu dieser Prozedur findet sich in Abschnitt 6.9 ein Anwendungsbeispiel.

5.16 Das Modul 'RndVar8'

Das Modul 'RndVar8' stellt Prozeduren für die Berechnung von Randverteilungen und zur Erzeugung von gemeinsamen Verteilungen zur Verfügung.

Exportierte Prozeduren:

5.16.1 Berechnen gemeinsamer Verteilungen aus unabhängigen Verteilungen

```
PROCEDURE GemeinsamUnZV(X,Y,Z:ZufallsVar;indx:INTEGER);
```

Diese Prozedur berechnet die gemeinsame Verteilung aus den Verteilungen der Zufallsvariablen X und Y; die Zufallsvariable X ist (d-1)-dimensional, die Zufallsvariable Y eindimensional. Berechnet wird die Verteilung der d-dimensionalen Ergebniszufallsvariablen Z.

Wichtig ist, daß die Zufallsvariablen X und Y voneinander stochastisch unabhängig sind. Der Parameter 'indx' legt die Komponente von Z fest, die die Wertmenge von Y aufnimmt. Für die Wahrscheinlichkeit für ein Tupel aus der Wertmenge von Z gilt:

$$\begin{aligned} &P\{Z = (z_1, \ldots, z_{indx-1}, z_{indx}, z_{indx+1}, \ldots, z_d)\} := \\ &\quad P\{X = (z_1, \ldots, z_{indx-1}, z_{indx+1}, \ldots, z_d)\} \cdot P\{Y = z_{indx}\} \end{aligned}$$

5.16.2 Berechnen gemeinsamer Verteilungen aus bedingten Verteilungen

```
PROCEDURE GemeinsamAbhZV(XwennY : ARRAY OF ZufallsVar;
                         Y,Z    : ZufallsVar; indx : INTEGER);
```

Diese Prozedur berechnet die gemeinsame Verteilung aus einer Familie bedingter Verteilungen von Zufallsvariablen, die im Analyseprogramm in einem Feld gespeichert sind. Der Name des Feldes wird dem Parameter 'XwennY' zugewiesen. Der Parameter 'indx' legt die Komponente fest, die die Wertmenge von Y aufnimmt. Für die Wahrscheinlichkeit eines Tupels aus der Wertmenge von Z gilt:

$$\begin{aligned} &P\{Z = (z_1, \ldots, z_{indx-1}, z_{indx}, z_{indx+1}, \ldots, z_d)\} := \\ &\quad P\{XwennY = (z_1, \ldots, z_{indx-1}, z_{indx+1}, \ldots, z_d) \mid Y = z_{indx}\} \cdot P\{Y = z_{indx}\} \end{aligned}$$

5.16.3 Randverteilungen berechnen

```
PROCEDURE RandVert(X: ZufallsVar; indx: INTEGER; Y: ZufallsVar);
```

Als Parameter 'X' wird der Prozedur eine mehrdimensionale diskrete Zufallsvariable $\mathbf{X} = (X_1, X_2, \ldots, X_d)$, $d \in [1 : \text{'MaxDim'}]$, übergeben. Die eindimensionale, diskrete Zufallsvariable Y erhält die Verteilung der Zufallsvariablen X_{indx}, die wie folgt festgelegt ist:

$$P\{Y = y\} := \sum_{x \in \Omega_X | x_{indx} = y} P\{X = (x_1, \ldots, x_{indx-1}, x_{indx}, x_{indx+1}, \ldots, x_d)\},$$

wobei d die Dimension der Verteilung von X ist.

5.16.4 Mehrdimensionale Randverteilung

```
PROCEDURE RandVertMD(X,Y:ZufallsVar;indx:INTEGER);
```

Die Prozedur 'RandVertMD' berechnet aus der Verteilung einer d-dimensionalen Zufallsvariablen X die Randverteilung einer $(d-1)$-dimensionalen Zufallsvariablen. Der Parameter 'indx' legt die Komponente fest, über die summiert wird:

$$P\{Y = (x_1, \ldots, x_{indx-1}, x_{indx+1}, \ldots, x_d)\} := \sum_{x_{indx} \in \Omega_{indx}} P\{X = (x_1, \ldots, x_{indx}, \ldots, x_d)\}$$

5.16.5 Verteilung der Summe von Zufallsvariablen

```
PROCEDURE SummenVert(X, Y: ZufallsVar);
```

Dieser Prozedur ist als Parameter 'X' eine mehrdimensionale, diskrete Zufallsvariable $\mathbf{X} = (X_1, X_2, \ldots, X_d)$, $d \in I\!N$, zu übergeben. Als Verteilung der eindimensionalen Zufallsvariablen Y mit diskretem Wertebereich berechnet die Prozedur

$$P\{Y = y\} := P\{X_1 + X_2 + \ldots + X_d = y\}.$$

5.16.6 Bedingte Wahrscheinlichkeiten

```
PROCEDURE BedingtVert(X: ZufallsVar; indx: INTEGER;
                      Bedingung: ITupel; Y: ZufallsVar);
```

Der Prozedur wird als Parameter X eine diskrete Zufallsvariable mit Dimension d übergeben, deren Verteilung der gemeinsamen Verteilung der d Zufallsvariablen $X_1, X_2, \ldots, X_d$ entspricht. Als Parameter 'Y' ist eine Zufallsvariable mit eindimensionalem diskreten Wertebereich zu übergeben. Die Verteilung von Y wird durch die Prozedur wie folgt bestimmt:

$$P\{Y = y\} :=$$

$$P\{X_{indx} = y \mid (X_1, \ldots, X_{indx-1}, X_{indx+1}, \ldots, X_d) = (x_1, \ldots, x_{indx-1}, x_{indx+1}, \ldots, x_d)\}$$

Das Tupel $(x_1, \ldots, x_{indx-1}, x_{indx+1}, \ldots, x_d)$ mit $d-1$ Komponenten wird der Prozedur als Parameter 'Bedingung' übergeben.

6 Beispiele für die Programmierung von Operationen

Ein Reihe von Beispielen für die Programmierung von Operationen auf Zufallsvariablen mit Hilfe der Programmbibliothek ist in den folgenden Abschnitten zusammengestellt. Die aufgeführten Operationen werden in mehr oder weniger abgewandelter Form bei vielen Analyseprogrammen Verwendung finden. Das Ziel der Zusammenstellung ist, dem Modellierer den Einstieg in die Benutzung der Bibliothek zu erleichtern und zu beschleunigen.

Die aufgeführten Beispiele behandeln zumeist Aufgabenstellungen, wie sie in der Warteschlangentheorie häufig vorkommen. Bei der Umsetzung der Operationen in der Schreibweise von Algorithmen wird auf eventuelle Besonderheiten der Implementierungen eingegangen. Die Programmstücke enthalten nur die wesentlichen Bestandteile, wie die Deklaration und Initialisierung von Variablen und Zufallsvariablen, sowie die Befehle, die die jeweiligen Operationen realisieren.

Im folgenden wird mit Ω_X grundsätzlich die Wertmenge einer Zufallsvariablen X bezeichnet.

6.1 Definition eindimensionaler Verteilungen

Besonders im Initialisierungsabschnitt eines Analyseprogramms müssen für Zufallsvariablen Verteilungen festgelegt werden. Beispiele hierfür sind die Verteilungen des Anfangszustands eines Systems, der Bedienzeit und der Zwischenankunftszeit von Kunden.

Für die Definition von Verteilungen eindimensionaler Zufallsvariablen gibt es drei unterschiedliche Verfahren, nämlich die Verwendung der Prozeduren zur Erzeugung von Standard-Verteilungen, die Benutzung von selbsterstellten Funktionen vom Typ 'PMF', sowie das explizite Zuweisen von Wahrscheinlichkeiten und Momenten an die Elemente des Wertebereiches einer Zufallsvariablen.

6.1.1 Explizites Zuweisen von Wahrscheinlichkeiten

Die Methode eignet sich besonders, wenn Zufallsvariablen einen sehr kleinen Wertebereich besitzen. Sie ist aber auch anzuwenden, wenn die Definition der Verteilung mit Hilfe der ersten beiden obengenannten Methoden nicht möglich ist, etwa für beliebige Häufigkeitsverteilungen. Die Zufallsvariablen müssen statisch erzeugt worden sein.

Bevor die Wahrscheinlichkeiten zugewiesen werden, muß sichergestellt sein, daß die Verteilung noch keine Wahrscheinlichkeiten enthält, d.h. nullgesetzt ist. Hierzu wird die Prozedur 'Setze0Vert' aus dem Modul 'RVBasics' aufgerufen. Auf die Anweisungen zur Festlegung der Wahrscheinlichkeiten folgend muß die Prozedur 'VertVeraendert' aufgerufen werden, um anzuzeigen, daß sich die Verteilung nicht mehr im nullgesetzten Zustand befindet, falls *nicht* die von der Programmbibliothek zur Verfügung gestellten Prozeduren benutzt werden.

Beispiel:

Die Anzahl N der Kunden in einer Warteschlange mit ν Plätzen soll zu Beginn einer Analyse mit der Wahrscheinlichkeit 1 gleich null sein.

Definitionen:

N, Zufallsvariable mit $\Omega_N = \{0, 1, \ldots, \nu\}$;

Operationen:

$N := 0$;

Programm:

```
MODULE ...;

FROM SYSTEM   IMPORT ADR;
FROM Reals    IMPORT Real,RK;
FROM RVBasics IMPORT ZufallsVar,InitDskrZV1D,Setze0Vert,VertVeraendert;
...

VAR N  : ZufallsVar;
    pN : ARRAY [0..nue] OF Real;
...

BEGIN
InitDskrZV1D(N,'N ',ADR(pN),0,nue);
...

Setze0Vert(N);          (* notwendig, wenn N vorher schon benutzt wurde *)
pN[0] := RK(1.0);
VertVeraendert(N);
...
```

Das Verfahren läßt sich auch auf Zufallsvariablen mit stetigen Wertebereichen anwenden. Hierbei müssen allerdings einige zusätzliche Dinge beachtet werden, wie das folgende Beispiel zeigt:

Beispiel:

Die Dichte einer Zufallsvariablen X soll im negativen Abschnitt der reellen Achse durch eine Exponentialfunktion und über dem Intervall $[0, 1)$ durch eine konstante Funktion gegeben sein. Die Wahrscheinlichkeit von $P\{X \geq 1\}$ soll 0 betragen.

Definitionen:

X: Zufallsvariable mit $\Omega_X = (-\infty, 1)$

Operationen:

Die Verteilung von X sei gegeben durch die Dichte f_X,

$$f_X(x) := \begin{cases} \frac{1}{4}e^{\frac{1}{2}x} & \text{für } x < 0 \\ \frac{1}{2} & \text{für } 0 \leq x < 1 \\ 0 & \text{sonst} \end{cases}$$

Programm:

```
MODULE ...;

FROM SYSTEM   IMPORT ADR;
FROM Reals    IMPORT Real,RK;
FROM RVBasics IMPORT ZufallsVar,InitStetZV,Setze0Vert,VertVeraendert;
...

VAR X : ZufallsVar;           (* Zufallsvariable X                      *)
    fX: ARRAY[0..9] OF Real;  (* Array fuer die stetige Verteilung ueber *)
                              (* endlichen Intervallen                   *)
    i     : INTEGER;          (* Schleifenvariable                       *)
    delta : Real;             (* Intervallaenge fuer die 101-Approx.     *)
...

BEGIN
...
delta := RK(0.1);

InitStetZV(X,'X ',ADR(fX),0,9,delta); (* Initialisierung der ZV *)
...

Setze0Vert(X);               (* Nullsetzen der Verteilung        *)

X^.Wlinks := RK(0.5);        (* W.-Masse, die auf das Intervall   *)
                             (* (-oo,0] entfaellt                 *)
X^.Mlinks := RK(1.0);        (* 1. Moment ueber diesem Intervall  *)

FOR i := 0 TO 9 DO           (* W.-Massen, die auf die Intervalle *)
  fX[i] := RK(0.05);         (* [0,0.1), [0.1,0.2),...,[0.9,1.0)  *)
END;                         (* entfallen                         *)

X^.aStet := 0;               (* Benutzten Indexbereich angeben    *)
X^.bStet := 9;

VertVeraendert(X);           (* Verteilung als nicht nullgesetzt  *)
                             (* markieren                         *)
...
```

Entsprechend können Zuweisungen an Verteilungen von Zufallsvariablen mit diskret unendlichem Wertebereich vorgenommen werden.

Manchmal kommt es vor, daß Wahrscheinlichkeiten aus diskreten Verteilungen von Zufallsvariablen verändert werden müssen, auf deren Verteilungsfelder im Analyseprogramm nicht mittels Feldindizierung zugegriffen werden kann. Dies ist unter anderem bei Zufallsvariablen der Fall, die mit den Prozeduren 'GenZV' oder 'DynZV' erzeugt worden sind. Anstelle einer Zuweisung 'pX[i] := p' wird dann der Prozeduraufruf 'Pgleich(X,p)' verwendet (siehe auch Abschnitt 5.8.5).

6.1.2 Benutzung von Funktionen vom Typ 'PMF'

Vom Modul 'RndVar0' wird der Prozedur-Typ 'PMF' exportiert:

```
TYPE PMF  = PROCEDURE(INTEGER): Real;
```

Prozeduren dieses Typs können benutzt werden, um eindimensionale diskrete Verteilungen von Zufallsvariablen festzulegen. Hierbei wird die vom selben Modul exportierte Prozedur 'BesetzTWbereichPMF' verwendet.

```
PROCEDURE BesetzTWbereichPMF(X          : ZufallsVar;
                             Start, Ende: INTEGER;
                             Prob       : PMF);
```

Wie die Prozedur benutzt wird, zeigt das folgende einfache

Beispiel:

Definitionen:

X: Zufallsvariable mit $\Omega_X = \{1, 2, 3, 4\}$;

Operationen:

$P\{X = x\} := \frac{5-x}{10}$, für $1 \leq x \leq 4$;

Programm:

```
MODULE ...;

FROM SYSTEM   IMPORT ADR;
FROM Reals    IMPORT Real,RK;
FROM RVBasics IMPORT ZufallsVar,InitDskrZV1D,Setze0Vert;
FROM RndVar0  IMPORT BesetzTWbereichPMF;
...

VAR X  : ZufallsVar;
    pX : ARRAY[1..4] OF Real;
...

PROCEDURE Absteigend(x: INTEGER): Real;
BEGIN
  RETURN RK(FLOAT(5-x))/RK(10);  (* Die Prozedur realisiert eine    *)
END Absteigend;                  (* Probability Mass Function (PMF) *)
...
BEGIN

  InitDskrZV1D(X,'X ',ADR(pX),1,4);
  ...
  Setze0Vert(X);              (* Sicherstellen, dass die Verteilung *)
                              (* nullgesetzt ist                    *)

  BesetzTWBereichPMF(X,1,4,Absteigend);   (* PMF verwenden          *)
  ...
END ...  .
```

Auch bei dieser Variante muß der Modellierer für die Nullsetzung der Verteilung sorgen, bevor neue Wahrscheinlichkeiten zugewiesen werden. Die Prozedur 'BesetzTWBereichPMF' sorgt nicht für die Nullsetzung der Verteilung. Die Markierung der Verteilung als 'verändert' wird von der Prozedur automatisch vorgenommen.

6.1.3 Erzeugung von Standardverteilungen

Das Modul 'RndVar3' exportiert Prozeduren, die Zufallsvariablen häufig benötigte Verteilungen zuweisen. Die Verwendung dieser Prozeduren ist das einfachste Verfahren, Verteilungen zu definieren.

Die Beschreibung der Prozeduren findet sich im Abschnitt 5.11. Als Beispiel sei hier stellvertretend die Verwendung der Prozedur 'HExpVert' (Hyperexponentialverteilung) aufgeführt. Die Prozedur 'HExpVert' ist wie folgt deklariert:

```
PROCEDURE HExpVert(ZV: ZufallsVar; lambda1,lambda2,p: Real);
```

Beispiel:

Die Bedienzeit S eines Bedieners in einem Warteschlangensystem sei hyperexponentialverteilt mit den Parametern $\lambda_1 = \frac{1}{2}$, $\lambda_2 = \frac{3}{4}$ und $p = \frac{1}{3}$.

Definitionen:

S: Zufallsvariable mit $\Omega_S = I\!R^+$;

Operationen:

$f_S(x) := \frac{p}{\lambda_1} e^{-\lambda_1 x} + \frac{1-p}{\lambda_2} e^{-\lambda_2 x}$;

Programm:

```
MODULE ...;

FROM SYSTEM   IMPORT ADR;
FROM Reals    IMPORT Real,RK;
FROM RVBasics IMPORT ZufallsVar,InitStetZV;
FROM RndVar3  IMPORT HExpVert;
...
VAR S  : ZufallsVar;
    fS : ARRAY [0..99] OF Real;

    delta : Real;
...

BEGIN
  delta := RK(0.1);                          (* Intervallaenge fuer DME-1 *)
  ...
  InitStetZV(S,'S ',ADR(fS),0,99,delta);     (* ZV initialisieren         *)
  ...

  HExpVert(S,RK(0.5),RK(0.75),RK(1.0)/RK(3.0)); (* Verteilung definieren *)
  ...
```

Bei Verwendung der vorgegebenen Prozeduren müssen also durch den Modellierer weder die Verteilungen nullgesetzt werden noch ist eine Markierung von Verteilungen als verändert erforderlich. Diese Aufgaben werden vollständig von der benutzten Prozedur übernommen.

6.2 Definition mehrdimensionaler Verteilungen

Die Festlegung der Verteilungen mehrdimensionaler Zufallsvariablen kann wie bei eindimensionalen Zufallsvariablen durch direkte Zuweisung von Wahrscheinlichkeiten an Elemente des Wertebereiches geschehen. Die Verwendung der Prozedur 'GemeinsamAbhZV' stellt eine weitere Möglichkeit dar. Beide Verfahren werden im folgenden durch ein Beispiel verdeutlicht.

Beispiel: Gegeben sei ein Pollingsystem mit $K = 5$ Warteschlangen zu je $\nu = 5$ Plätzen. Die Anzahl der Kunden in den einzelnen Warteschlangen zu Beginn der Betrachtung wird durch die K-dimensionale Zufallsvariable $\mathbf{N}$ beschrieben. Deren Verteilung wird derart gewählt, daß sich mit einer Wahrscheinlichkeit von 1.0 in einer der Warteschlangen zwei Kunden aufhalten und in allen anderen genau ein Kunde. Alle Warteschlangen sind mit gleicher Wahrscheinlichkeit diejenige, in der sich zwei Kunden befinden.

Definitionen:

$K = 5$;

$\nu \in I\!N$;

$\mathbf{N}$: Zufallsvariable mit $\Omega_N = [0 : \nu]^K$;

Operationen:

$\nu := 5;$

$$P\{\mathbf{N} = (n_1, n_2, \ldots, n_K)\} := \begin{cases} \frac{1}{K} & \text{für alle } (n_1, n_2, \ldots, n_K) \in \Omega_N \text{ mit} \\ & n_i = 2 \text{ für genau ein } i \text{ und} \\ & n_j = 1 \text{ für alle } j \neq i, \\ & i, j \in [1 : K] \\ & \\ 0 & \text{sonst} \end{cases}$$

Programm:

```
MODULE ...;

FROM SYSTEM   IMPORT ADR;
FROM Reals    IMPORT Real,RK;
FROM Tupel    IMPORT ITupel,BildeITupel5;
FROM RVBasics IMPORT ZufallsVar,InitDskrZVMD,Setze0Vert,VertVeraendert;
...

CONST nue = 5;

VAR   N : ZufallsVar;
      pN: ARRAY [0..nue],[0..nue],[0..nue],[0..nue],[0..nue] OF Real;

      ug,og : ITupel;
      p     : Real;
      ...

BEGIN
...

  BildeITupel5(ug,0,0,0,0,0);
  BildeITupel5(og,nue,nue,nue,nue,nue);
  InitDskrZVMD(N,'N ',ADR(pN),ug,og);
  ...

  Setze0Vert(N);
  p := RK(1.0)/RK(5.0);
  pN[2,1,1,1,1] := p;
  pN[1,2,1,1,1] := p;
  pN[1,1,2,1,1] := p;
  pN[1,1,1,2,1] := p;
  pN[1,1,1,1,2] := p;
  VertVeraendert(N);
  ...
END ... .
```

Beispiel:

X und Y seien stochastisch abhängige Zufallsvariablen mit den Wertmengen $\Omega_{\mathbf{X}} = [0:\xi] \times [0:\xi] \times [0:\xi]$ und $\Omega_Y = [0:\upsilon]$. Es seien $\upsilon + 1$ Verteilungen von $\mathbf{X}$ bekannt, jeweils unter der Voraussetzung, daß $Y = y, \quad \forall y \in \Omega_Y$. Aus den dreidimensionalen Verteilungen der Zufallsvariablen $\mathbf{X} \mid Y = y, y \in \Omega_Y$, und der Verteilung von Y soll ihre gemeinsame Verteilung berechnet werden. Sie ergebe sich als Verteilung der vierdimensionalen Zufallsvariablen $\mathbf{Z} = (x_1, x_2, y, x_3)$, d. h. in der Repräsentation von $\mathbf{Z}$ steht der Wert der Zufallsvariablen Y an dritter Stelle.

Definitionen:

$\xi := 5$
; $\upsilon := 10$;

$\mathbf{X} \mid Y = y, \quad y \in \{0, 1, \dots, \upsilon\}$: Zufallsvariablen mit $\Omega_{\mathbf{X}} = [0:\xi] \times [0:\xi] \times [0:\xi]$, Verteilung gegeben;

Y: Zufallsvariable mit $\Omega_Y = [0:\upsilon]$, Verteilung gegeben;

$\mathbf{Z}$: Zufallsvariable mit $\Omega_{\mathbf{Z}} = [0:\xi] \times [0:\xi] \times [0:\upsilon] \times [0:\xi]$;

Operationen:

```
for all $(x_1, x_2, x_3) \in \Omega_{\mathbf{X}}$ do
    for all $y \in \Omega_Y$ do
        $P\{\mathbf{Z} = (x_1, x_2, y, x_3)\} := P\{\mathbf{X} = (x_1, x_2, x_3) \mid Y = y\} \cdot P\{Y = y\}$;
    od;
od;
```

Programm:

```
MODULE ...;

FROM SYSTEM   IMPORT ADR;
FROM Reals    IMPORT Real,RK;
FROM Tupel    IMPORT ITupel,BildeITupel3,BildeITupel4;
FROM RVBasics IMPORT ZufallsVar,InitDskrZV1D,InitDskrZVMD;
FROM RndVar8  IMPORT GemeinsamAbhZV;

CONST xi  = 5;
      ups = 10;

VAR XwennY  : ARRAY[0..ups] OF ZufallsVar;
    pXwennY : ARRAY[0..ups] OF ARRAY[0..xi],[0..xi],[0..xi] OF Real;

    Y  : ZufallsVar;
    pY : ARRAY[0..ups] OF Real;

    Z  : ZufallVar;
    pZ : ARRAY[0..xi],[0..xi],[0..ups],[0..xi] OF Real;

    ugx,ogx,
    ugz,ogz  : ITupel;
    y        : INTEGER;

BEGIN
  ...
  (*     Initialisierungsteil      *)
  BildeITupel3(ugx, 0, 0, 0);
  BildeITupel3(ogx,xi,xi,xi);
  FOR y := 0 TO ups DO
    InitDskrZVMD(XwennY[y],'XwennY[]',ADR(pXwennY[y]),ugx,ogx);
  END;

  InitDskrZV1D(Y,'Y',ADR(pY),0,ups);

  BildeITupel4(ugz, 0, 0,  0, 0);
  BildeITupel4(ogz,xi,xi,ups,xi);
  InitDskrZVMD(Z,'Z',ADR(pZ),ugz,ogz);
  (*   Ende Initialisierungsteil   *)
  ...

  GemeinsamAbhZV(XwennY,Y,Z,3);

  ...
```

6.3 Zuweisung von Verteilungen

Häufig kommt es vor, daß Zufallsvariablen eine Verteilung erhalten sollen, die identisch ist mit der einer anderen Zufallsvariablen. Eine MODULA-2-Anweisung der Art

```
X := Y;
```

ist aber ungeeignet, da hierdurch lediglich der Variablen 'X', die als vom Typ 'ZufallsVar' und somit als 'POINTER TO ZVStrukt' deklariert ist, als Wert die Adresse der Beschreibungsstruktur der Zufallsvariablen 'Y' zugewiesen wird. Dies hat zur Folge, daß die Beschreibungsstruktur von 'X' als Speichermüll verloren geht und sich jede Veränderung der Verteilung von 'Y' sofort auf die Verteilung von 'X' auswirkt. Zudem ist in der Beschreibungsstruktur der Name einer anderen Zufallsvariablen vermerkt, in diesem Fall der Name 'Y '.

Um einer Zufallsvariablen X die Verteilung einer anderen Zufallsvariablen Y zuweisen zu können, exportiert das Modul 'RndVar2' die Prozedur 'ZuweisenZV'. Bedingung für die Anwendung dieser Prozedur ist, daß der Wertebereich von X Obermenge des Wertebereiches von Y ist. Neben den deklarierten stetigen Wertebereichen müssen zusätzlich die Intervallbreiten 'δ' für die Approximation mittels 101-Dichte übereinstimmen.

Beispiel:

Definitionen:

X, Y: Zufallsvariablen mit $\Omega_X = \Omega_Y = \Omega$;

Operationen:

$P\{X \leq x\} := P\{Y \leq x\}$ für alle $x \in \Omega$;

Programm:

```
FROM RndVar2  IMPORT ZuweisenZV;
...

ZuweisenZV(Y,X);   (* die Verteilung von Y wird der ZV X zugewiesen *)
...
```

6.4 Momente von Verteilungen

Zur Berechnung der Momente von Verteilungen werden mit der Programmbibliothek die Prozeduren 'Moment0', 'Erwartung1D', 'Erwartung' und 'Momente1D' bereitgestellt. Ihre Deklarationen werden durch das Modul 'RndVar2' exportiert, ihre Beschreibungen finden sich in Abschnitt 5.10. Die Prozeduren 'Erwartung1D' und 'Momente1D' lassen sich nur auf Zufallsvariablen mit eindimensionalem Wertebereich anwenden. Die Prozedur 'Moment0' kann sehr einfach mit ein- und mehrdimensionalen Wertebereichen verwendet werden, da die Norm einer Zufallsvariablen immer ein skalarer Wert ist. Als erstes Moment einer Zufallsvariablen mit mehrdimensionalem Wertebereich wird im folgenden das Tupel der Erwartungswerte aller Randverteilungen betrachtet. Die Prozedur 'Erwartung' benötigt deshalb einen aktuellen Parameter vom Typ 'RTupel'. Die Prozedur kann zwar auf Zufallsvariablen mit eindimensionalen Wertebereichen angewendet werden, einfacher ist aber in diesem Fall die Benutzung der Funktion 'Erwartung1D'.

Beispiel: Betrachtet werden soll ein Pollingsystem mit $K = 5$ Warteschlangen; jede der Warteschlangen kann maximal $\nu = 10$ Kunden aufnehmen. Die Zufallsvariable N sei eine fünfdimensionale Zufallsvariable, welche den Inhalt der einzelnen Warteschlangen beschreibt. Die Zykluslänge ist die Anzahl von Zeiteinheiten, die der Bediener benötigt, um – startend bei einer Warteschlange W_i – alle Kunden aus den Warteschlangen $W_{((i+1) \bmod 5)+1}, W_{((i+2) \bmod 5)+1}, \cdots, W_i$ zu bedienen. Die Zykluslänge entspricht der Anzahl der in diesem Zyklus bedienten Kunden, wenn – wie hier angenommen – die Bedienung eines Kunden eine Zeiteinheit dauert. Die Zykluslänge sei durch die Zufallsvariable Z beschrieben.

Definitionen:

Pufferbelegung $\mathbf{N}$: Zufallsvariable, $\Omega_{\mathbf{N}} := \{0, 1, \ldots, 10\}^5$;

Zykluslänge Z: Zufallsvariable, $\Omega_Z := \{0, 1, \ldots, 50\}$;

$M^0_{\mathbf{N}}, E_{Z,0}, E_{Z,1} \in I\!R; (E_{Z,4}, E_{Z,5}, \ldots, E_{Z,10}) \in I\!R^7; E_{\mathbf{N}} \in I\!R^5$;

Operationen:

$M^0_{\mathbf{N}} := E[\mathbf{N}^0]$;

$E_{\mathbf{N}} := E[\mathbf{N}]$;

$E_{Z,0} := E[Z^0]$;

$E_{Z,1} := E[Z]$;

$E_{Z,i} := E[Z^i]$, für alle $i \in \{4, 5, ..., 10\}$;

Programm:

```
MODULE ...;
FROM SYSTEM   IMPORT ADR;
FROM Reals    IMPORT Real;
FROM Tupel    IMPORT ITupel,BildeITupel5,RTupel;
FROM RVBasics IMPORT ZufallsVar,InitDskrZV1D,InitDskrZVMD;
FROM RndVar2  IMPORT Moment0,Erwartung1D,Erwartung,Momente1D;
...
VAR N,Z: ZufallsVar;

    pN  : ARRAY [0..10],[0..10],[0..10],[0..10],[0..10] OF Real;
    pZ  : ARRAY [0..50] OF Real;
    MON : Real;
    EN  : RTupel;
    EZ0 : Real;
    EZ1 : Real;
    EZ  : ARRAY [4..10] OF Real;

    ug,og : ITupel;

    ...

BEGIN

  BildeITupel5(ug, 0, 0, 0, 0, 0);
  BildeITupel5(og,10,10,10,10,10);

  InitDskrZVMD(N,'N ',ADR(pN),ug,og);
  InitDskrZV1D(Z,'Z ',ADR(pZ),0,50);
  ...
  MON := Moment0(N);
  Erwartung(N,EN);

  EZ0 := Moment0(Z);
  EZ1 := Erwartung1D(Z);
  Momente1D(Z,EZ,4);       (* ab 4. Moment *)
  ...
```

6.5 Normierung von Verteilungen

Eine Verteilung, die als Ergebnis aus einer langen Kette von Berechnungen hervorgeht, ist meist mit Fehlern behaftet, die sich u.a. darin äußern, daß die Normierungsbedingung nicht erfüllt ist. Die Summe aller Wahrscheinlichkeiten über dem Wertebereich der Verteilung weicht vom Wert '1.0' ab.

Beim Normieren von Verteilungen wird der Faktor der Abweichung bestimmt und durch Division aller Wahrscheinlichkeiten durch diesen Faktor die Erfüllung der Normierungsbedingung wiederhergestellt.

VERENA bietet für die Normierung von Verteilungen drei Prozeduren an, die von dem Modul 'RndVar2' exportiert werden und auf alle Arten von Zufallsvariablen anwendbar sind. Die Prozedur 'NormiereZV' führt die Normierung in jedem Falle durch, wogegen die Prozedur 'NormiereZVBeiAbweichung' eine Verteilung nur dann normiert, wenn die als Parameter übergebene maximal zulässige absolute Abweichung vom Sollwert 1 überschritten ist. Auch die Prozedur 'NormiereZVRundFehlKorr' führt eine Normierung durch. Treten dabei negative Werte im DME-0-Bereich einer Zufallsvariablen auf, so wird dies dem Benutzer gemeldet. Sind die negativen Werte betragsmäßig klein, so wird davon ausgegangen, daß es sich um Rundungsfehler handelt. Eine entsprechende Warnung wird ausgegeben und die Ausführung des Analyseprogramms wird fortgesetzt. Treten bei der Normierung Werte auf, die betragsmäßig größer als 10^{-13} sind, so handelt es dabei vermutlich um einen Programmierfehler seitens des Modellierers. Auch in diesem Fall wird das Analyseprogramm mit einer entsprechenden Warnung fortgesetzt. In den Fällen, in denen der Modellierer Rundungsfehler vermutet, sollte die Prozedur 'NormiereZVRundFehlKorr' an Stelle der Prozedur 'NormiereZV' verwendet werden.

Die Normierung von Zufallsvariablen wie beschrieben ist in der Regel nicht Bestandteil von Analysealgorithmen sondern eine programmiertechnische Notwendigkeit.

Die Normierungsoperation kann jedoch auch bei der Berechnung der Verteilungen wie im folgenden Beispiel benutzt werden.

Beispiel:

Gegeben sei eine diskrete Zufallsvariable X mit $\Omega_X = \{1, 2, \ldots, 100\}$. Als die Verteilung der Zufallsvariablen Y sei die Verteilung von X unter der Bedingung, daß X geradzahlig ist, zu bestimmen.

Definitionen:

X: Zufallsvariable, $\Omega_X := \{1, 2, \ldots, 100\}$

Y: Zufallsvariable, $\Omega_Y := \{2, 4, \ldots, 100\}$

Operationen:

$Y := (X \mid X \in \{2, 4, \ldots, 100\});$

Programm:

```
MODULE ...;

FROM SYSTEM   IMPORT ADR;
FROM Reals    IMPORT Real;
FROM RVBasics IMPORT ZufallsVar,InitDskrZV1D;
FROM RndVar2  IMPORT NormiereZV;
...

VAR X,Y: ZufallsVar;

    pX : ARRAY [1..100] OF Real;
    pY : ARRAY [2..100] OF Real;

    i  : INTEGER;
    ...

BEGIN

  InitDskrZV1D(X,'X ',ADR(pX),1,100);
  InitDskrZV1D(Y,'Y ',ADR(pY),2,100);

  ...

  FOR i := 2 TO 100 BY 2 DO
    pY[i] := pX[i]
  END;
  NormiereZV(Y);
  ...
```

6.6 Addition von Zufallsvariablen durch Faltung ihrer Verteilungen

Die Faltungsoperation, d.h. die Addition der Verteilungen zweier Zufallsvariablen bei Unabhängigkeit, findet in fast allen Analyse-Algorithmen Anwendung, z.B. bei der Berechnung von Wartezeiten, Pufferbelegungen oder Zykluslängen. Im folgenden sind einige wichtige Anwendungsformen mit Programmbeispielen aufgeführt.

Beispiel:

Gegeben sei das Modell eines Bediensystems, dessen Bedienzeit B stetig verteilt ist. Dementsprechend ist die Wartezeit W für einen beliebigen Kunden ebenfalls stetig verteilt mit einem zusätzlichen Wahrscheinlichkeitsmassepunkt an der Stelle 0, für den Fall, daß der Kunde den Bediener frei vorfindet. Aus gegebenen Verteilungen für Bedienzeit und Wartezeit soll die Verteilung für die Verweilzeit V des Kunden im System berechnet werden.

Definitionen:

B, V: Zufallsvariablen mit $\Omega_B = \Omega_V = I\!R^+$;

W: Zufallsvariable mit $\Omega_W = I\!R_0^+$;

Operationen:

$V := W \oplus B$;

Programm:

```
MODULE ...;

FROM SYSTEM   IMPORT ADR;
FROM Reals    IMPORT Real;
FROM RVBasics IMPORT ZufallsVar,InitStetZV,InitGemZV;
FROM RndVar4  IMPORT FalteZV;
...

VAR B,W,V : ZufallsVar;

    fB : ARRAY [0..19]  OF Real;
    fW : ARRAY [0..99]  OF Real;
    pW : ARRAY [0..0]   OF Real;    (* alternativ auch pW0 : Real; *)
    fV : ARRAY [0..109] OF Real;

    delta : Real;
    ...



BEGIN

  delta := RK(0.1);

  InitStetZV(B,'B ',ADR(fB),0, 19,delta);
  InitGemZV (W,'W ',ADR(pW),0,  0,
                    ADR(fW),0, 99,delta);
  InitStetZV(V,'V ',ADR(fV),0,109,delta);
                        (* Die Anzahl der DME-0-Intervalle fuer V *)
                        (* kann beliebig gewaehlt werden          *)
  ...

  FalteZV(W,B,V);
  ...
```

Beispiel:

Aus Zeitscheibenlängen, die durch Zufallsvariablen Y_k beschrieben werden, soll die Verteilung der Dauer eines vollständigen Zyklus Z in einem zyklischen Pollingsystem ohne Umschaltzeiten und mit K Warteschlangen berechnet werden. Dabei wird angenommen, daß $Y_k,\ k \in [1:K]$, unabhängige Zufallsvariablen sind. Die Zufallsvariablen $Y_k,\ k \in [1:K]$, und damit auch Z haben endliche, diskrete Wertmengen.

Definitionen:

$K = 10$;

$\nu = 50$;

$Y_k, k \in [1..K]$: Zufallsvariable, $\Omega_{Y_k} = \{0, 1, \ldots, \nu\}$;

Z: Zufallsvariable, $\Omega_Z = \{0, 1, \ldots, \nu K\}$;

Operationen:

$Z := \oplus_{k=1}^{K} Y_k$;

Programm:

```
MODULE ...;
FROM SYSTEM   IMPORT ADR;
FROM Reals    IMPORT Real;
FROM RVBasics IMPORT ZufallsVar,InitDskrZV1D,GenZV,DelZV,TauscheZV;
FROM RndVar2  IMPORT ZuweisenZV;
FROM RndVar4  IMPORT FalteZVFolge;
...
CONST K = 10;
      nue = 50;

VAR Z : ZufallsVar;
    pZ: ARRAY[0..K*nue] OF Real;
    Y : ARRAY[1..K] OF ZufallsVar;
    pY: ARRAY[1..K] OF ARRAY [0..nue] OF Real;
    k : INTEGER;
BEGIN

  InitDskrZV1D(Z,'Z ',ADR(pZ),0,K*nue);
  FOR k := 1 TO K DO
    InitDskrZV1D(Y[k],'Y[]',ADR(pY[k]),0,nue);
  END;
  ...
  FalteZVFolge(Y[1],Y[K],Z);
  ...
```

Die Aufgabenstellung läßt sich also mit einer einzigen Operation lösen. Es gibt jedoch Fälle, bei denen die einzelnen Faltungsoperationen explizit programmiert werden sollten, um zum Beispiel Zwischenergebnisse ausgeben oder speichern und später weiterverarbeiten zu können. Für die Berechnung von Z sind dann $K - 1$ aufeinanderfolgende Faltungsoperationen notwendig.

Eine weitere Besonderheit dieses Lösungswegs ist die Benutzung der Prozedur 'FalteDskrZVBeschr', welche die Wahrscheinlichkeiten, die außerhalb des VME-0-Bereiches der Ergebnis-Zufallsvariablen liegen, auf den unteren oder oberen Randpunkt dieses Bereiches addiert (siehe auch Abschnitt 5.12.3).

Programm:

```
MODULE ...;
FROM SYSTEM   IMPORT ADR;
FROM Reals    IMPORT Real;
FROM RVBasics IMPORT ZufallsVar,InitDskrZV1D,GenZV,DelZV,TauscheZV;
FROM RndVar1  IMPORT WriteZV;
FROM RndVar2  IMPORT ZuweisenZV;
FROM RndVar4  IMPORT FalteDskrZVBeschr;

CONST K   = 10;
      nue = 50;
VAR Z       : ZufallsVar;
    pZ      : ARRAY[0..K*nue] OF Real;
    Y       : ARRAY[1..K] OF ZufallsVar;
    pY      : ARRAY[1..K] OF ARRAY [0..nue] OF Real;
    HZ1,HZ2 : ZufallsVar;   (* Hilfszufallsvariablen *)
    k       : INTEGER;
    ...
BEGIN
  InitDskrZV1D(Z,'Z ',ADR(pZ),0,K*nue);

  FOR k := 1 TO K DO
    InitDskrZV1D(Y[k],'Y[]',ADR(pY[k]),0,nue);
  END;
  ...
  GenZV(HZ1,Z);
  GenZV(HZ2,Z);
  ZuweisenZV(Y[1],HZ1);

  FOR k := 2 TO K DO
    FalteDskrZVBeschr(Y[k],HZ1,HZ2);

    (* Ausgabe eines Zwischenerg. mit Verteilung, da TRUE gesetzt *)
    WriteZV(HZ2,'Zwischenergebnis',TRUE);

    TauscheZV(HZ1,HZ2);
  END;

  ZuweisenZV(HZ1,Z);
  DelZV(HZ1);
  DelZV(HZ2);
  ...
```

Beispiel:

Gegeben sei eine Warteschlange mit ν Plätzen. Bei einem vorhergehenden Betrachtungszeitpunkt war die Anzahl der Kunden in der Warteschlange verteilt wie die Zufallsvariable N. Die Anzahl der Kunden, die seit diesem Zeitpunkt bis zum aktuellen Betrachtungszeitpunkt an der Warteschlange angekommen sind, sei durch die Zufallsvariable A beschrieben. Kunden, die eine volle Warteschlange antreffen, werden abgewiesen. Es soll die Verteilung des Füllungsgrads B der Warteschlange für den aktuellen Betrachtungszeitpunkt berechnet werden, wobei $P\{B = \nu + 1\}$ die Wahrscheinlichkeit dafür angibt, daß es zum Abweisen von Kunden gekommen ist. Die Zufallsvariablen A und N seien unabhängig.

Als betrachteter Wertebereich Ω_A genügt die Menge $\{0, 1, \ldots, \nu + 1\}$. $P\{A = \nu + 1\}$ ist die Wahrscheinlichkeit dafür, daß $\nu + 1$ oder mehr Kunden angekommen sind.

Die Besonderheit bei diesem Beispiel ist, daß unmittelbar auf die Faltungsoperation mit den Operanden N und A eine Minimum-Operation folgt, da die Anzahl von ν Kunden in der Warteschlange auf keinen Fall überschritten wird. Die beiden Operationen lassen sich günstig, d.h. unter Einsparung von Rechenzeit, zusammenfassen. Hierzu findet wiederum die Prozedur 'FalteDskrZVBeschr' Verwendung.

Definitionen:

N: Zufallsvariable mit $\Omega_N = \{0, 1, \ldots, \nu\}$;

A, B: Zufallsvariablen mit $\Omega_A = \Omega_B = \{0, 1, \ldots, \nu + 1\}$;

Operationen:

$B := \text{ Min}(N \oplus A, \nu + 1);$

Programm:

```
MODULE ...;
FROM SYSTEM   IMPORT ADR;
FROM Reals    IMPORT Real;
FROM RVBasics IMPORT ZufallsVar,InitDskrZV1D;
FROM RndVar4  IMPORT FalteDskrZVBeschr;
...
CONST nue = 30;
VAR A,B,N : ZufallsVar;
    pN    : ARRAY [0..nue] OF Real;
    pA,pB : ARRAY [0..nue+1] OF Real;
BEGIN
  InitDskrZV1D(N,'N ',ADR(pN),0,nue);
  InitDskrZV1D(A,'A ',ADR(pA),0,nue+1);
  InitDskrZV1D(B,'B ',ADR(pB),0,nue+1);
  ...
  FalteDskrZVBeschr(N,A,B);
  ...
```

6.7 Minimum und Maximum

Häufig müssen die Verteilungen von Zufallsvariablen berechnet werden, die als das Minimum oder das Maximum aus einer Zufallsvariablen und einer Konstanten definiert sind. VERENA bietet mit den Prozeduren 'MinZV1D' und 'MaxZV1D' die hierzu notwendigen Operationen an. Sie sind auf beliebige eindimensionale Zufallsvariablen anwendbar.

Beispiel:

Aufgrund von Sendelimits werden Kunden in einer Warteschlange zurückgelassen. Diese Warteschlange besitze $\nu = 40$ Plätze. Die Anzahl der zurückgelassenen Kunden sei durch die Zufallsvariable N beschrieben. Zu berechnen ist die Verteilung von Y, der Anzahl der bedienten Kunden.

Definitionen:

ν : Konstante mit $\nu = 40$ Puffergröße;

ξ : Konstante mit $\xi = 15$ Sendelimit; (D.h. es können bei einem Besuch des Bedieners nicht mehr als 15 Kunden bedient werden.)

N: Zufallsvariable mit $\Omega_N = \{0, 1, \ldots, \nu\}$;

Y: Zufallsvariable mit $\Omega_Y = \{0, 1, \ldots, \xi\}$;

Operationen:

$Y :=$ Min(N,ξ);

Programm:

```
MODULE ...;
FROM SYSTEM   IMPORT ADR;
FROM Reals    IMPORT Real;
FROM RVBasics IMPORT ZufallsVar,InitDskrZV1D;
FROM RndVar2  IMPORT MinZV1D;
...
CONST nue = 40;
      xi  = 15;
VAR   N,Y : ZufallsVar;
      pN  : ARRAY[0..nue] OF Real;
      pY  : ARRAY[0..xi]  OF Real;
BEGIN
  InitDskrZV1D(N,'N ',ADR(pN),0,nue);
  InitDskrZV1D(Y,'Y ',ADR(pY),0,xi);
  ...
  MinZV1D(N,xi,Y,FALSE);
  ...
```

Der letzte aktuelle Parameter der Prozedur 'MinZV1D' gibt an, ob die Verteilung von Y gleichzeitig normiert werden soll, indem als Wahrscheinlichkeit für $Y = \xi$ der Wert $1 - P\{Y < \xi\}$ benutzt wird. Durch die Angabe von 'FALSE' wird diese Normierung nicht durchgeführt.

Beispiel:

Gegeben sei ein G/G/1-System, das vom Standpunkt eines ankommenden Kunden betrachtet wird. Die stetige Zufallsvariable C beschreibt dabei, inwieweit der Bediener mit der Bedienung zuvor angekommener Kunden vorangekommen ist. Ein positiver Wert hat die Bedeutung, daß der Bediener noch C Zeiteinheiten für die Bedienung dieser Kunden benötigt, ein negativer Wert bedeutet, daß der Bediener seit C Sekunden frei ist. Zu bestimmen ist die Verteilung der Wartezeit W für den gerade angekommenen Kunden.

Definitionen:

C: Zufallsvariable mit $\Omega_C = \mathbb{R}$;

W: Zufallsvariable mit $\Omega_W = \mathbb{R}_0^+$;

Operationen:

$W := \text{Max}(C, 0);$

Programm:

```
MODULE ...;
FROM SYSTEM   IMPORT ADR;
FROM Reals    IMPORT Real;
FROM RVBasics IMPORT ZufallsVar,InitStetZV,InitGemZV;
FROM RndVar2  IMPORT MaxZV1D;
...
VAR W,C : ZufallsVar;
    fC  : ARRAY[-100,99] OF Real;
    fW  : ARRAY[   0,99] OF Real;
    pW  : ARRAY[   0, 0] OF Real;
    delta : Real;
    ...
BEGIN

  delta := RK(0.01);
  InitStetZV(C,'C ',ADR(fC),-100,99,delta);
  InitGemZV (W,'W ',ADR(pW),   0,  0,
                    ADR(fW),   0, 99,delta);
  ...
  MaxZV1D(C,0,W);
  ...
```

6.8 Satz der totalen Wahrscheinlichkeit

Häufig erfolgt die Berechnung der Verteilungen von Zufallsvariablen aus den Verteilungen bedingter Zufallsvariablen und den Wahrscheinlichkeiten der jeweiligen Bedingungen. Es wird also der Satz der totalen Wahrscheinlichkeit angewendet.

Beispiel:

Ein solcher Fall ist gegeben, wenn die Anzahl A der während Z Zeitintervallen an einer Warteschlange ankommenden Kunden berechnet werden soll. Die Ankünfte seien dabei Poisson-verteilt, d.h. die Verteilung der Anzahl der ankommenden Kunden für eine feste Anzahl z von Intervallen ist gegeben durch die Wahrscheinlichkeiten

$$P\{A = i \mid Z = z\} := p(i;\lambda) = e^{-\lambda z}\frac{(\lambda z)^i}{i!}, i \geq 0.$$

Definitionen:

$\lambda = 4;$

Z: Zufallsvariable mit $\Omega_Z = \{0, 1, \dots, 30\}$;

$A_z, z \in \{0, 1, \dots, 30\}$: Zufallsvariablen mit $\Omega_{A_z} = I\!N$;

A: Zufallsvariable mit $\Omega_A = I\!N$;

Operationen:

$$A := \begin{cases} A_0 & \text{mit der Wahrscheinlichkeit} & P\{Z = 0\}; \\ A_1 & \text{mit der Wahrscheinlichkeit} & P\{Z = 1\}; \\ \dots & \dots & \dots \\ A_{30} & \text{mit der Wahrscheinlichkeit} & P\{Z = 30\}; \end{cases}$$

Programm:

```
MODULE ...;
FROM SYSTEM   IMPORT ADR;
FROM Reals    IMPORT Real,RK;
FROM RVBasics IMPORT ZufallsVar,InitDskrZV1D,Setze0Vert;
FROM RndVar0  IMPORT SummWGewichtet;
FROM RndVar3  IMPORT PoissonVert;
...
VAR Z,A,Az: ZufallsVar;

    pZ    : ARRAY[0..30] OF Real;  (* Wahl der Bereichsgrenzen *)
    pA,pAz: ARRAY[0..50] OF Real;  (* nach geforderter         *)
                                   (* Genauigkeit              *)
    lambda : Real;
    z      : INTEGER;
    ...
BEGIN

  lambda := RK(4.0);

  InitDskrZV1D(Z ,'Z ',ADR(pZ) ,0,30);
  InitDskrZV1D(A ,'A ',ADR(pA) ,0,50);
  InitDskrZV1D(Az,'Az',ADR(pAz),0,50);
  ...
  Setze0Vert(A);
  FOR z := 0 TO 30 DO
    PoissonVert(Az,RK(FLOAT(z))*lambda);
    SummWGewichtet(Az,pZ[z],A);
  END;
  ...
```

Im vorhergehenden Programm wurden die Verteilungen der Zufallsvariablen $(A \mid Z = z)$ während der Berechnung von A bestimmt und von einer einzigen Zufallsvariablen 'Az' im Analyseprogramm repräsentiert. Die gleiche Aufgabe läßt sich aber auch lösen, indem die Zufallsvariablen $(A \mid Z = z)$ vor der eigentlichen Berechnung von A bestimmt werden. Das hat zwar den Nachteil, daß mehr Speicherplatz aufgewendet werden muß, jedoch den Vorteil, daß alle bedingten Verteilungen ohne weiteren Rechenaufwand für spätere Berechnungen zur Verfügung stehen.

Bei dem folgenden Lösungsweg wird jede Zufallsvariable $(A \mid Z = z)$ durch eine Komponente eines 'ARRAY [0..30] OF ZufallsVar' mit Namen 'AwennZ' repräsentiert. Für die Anwendung des Satzes der totalen Wahrscheinlichkeit stehen die Prozeduren 'TotalZV' (Abschnitt 5.10.15) und 'Total' (Abschnitt 5.10.16) zur Verfügung.

Programm:

```
MODULE ...;
FROM SYSTEM   IMPORT ADR;
FROM Reals    IMPORT Real,RK;
FROM RVBasics IMPORT ZufallsVar,InitDskrZV1D;
FROM RndVar2  IMPORT TotalZV, Total;
FROM RndVar3  IMPORT PoissonVert;
...
VAR Z,A    : ZufallsVar;
    AwennZ : ARRAY[0..30] OF ZufallsVar;
    pZ     : ARRAY[0..30] OF Real;
    pA     : ARRAY[0..50] OF Real;
    pAwennZ: ARRAY[0..30] OF ARRAY[0..50] OF Real;
    lambda : Real;
    z      : INTEGER;
    ...
BEGIN
  lambda := RK(4.0);
  ...
  InitDskrZV1D(Z ,'Z ',ADR(pZ) ,0,30);
  InitDskrZV1D(A ,'A ',ADR(pA) ,0,50);

  (* Festlegung der bedingten Verteilungen (A | Z=z) *)
  FOR z := 0 TO 30 DO
    InitDskrZV1D(AwennZ[z],'AwennZ',ADR(pAwennZ[z]),0,50);
    PoissonVert(AwennZ[z],RK(FLOAT(z))*lambda);
  END;
  ...
  (* Hier folgen weitere Anweisungen, u.a. die Berechnung der Vert. von Z *)
  ...
  TotalZV(AwennZ,Z,A,31);
  ...
```

Wenn statt der Prozedur 'TotalZV' die Prozedur 'Total, verwendet wird, so ist die Zeile des Prozeduraufrufs von 'TotalZV' durch die folgende Zeile zu ersetzen:

```
  Total(AwennZ[0], AwennZ[30], pZ[0], pZ[30], A);
```

Die Verwendung der Prozedur 'Total' ist in diesem Fall nicht sehr sinnvoll, da die bedingten Wahrscheinlichkeiten bereits als Feld von Zufallsvariablen deklariert sind und damit die Verwendung von 'TotalZV' adäquat ist. Außerdem muß der Benutzer bei der Verwendung von 'Total' selbst für die Korrektheit der Indexgrenzen sorgen (vgl. dazu Abschnitt 5.10.15 und Abschnitt 5.10.16).

Die Prozedur 'Total' ist aber dort zu verwenden, wo die bedingten Wahrscheinlichkeiten in einem Feld abgspeichert vorliegen, ohne daß es sich dabei um ein Verteilungsfeld einer Zufallsvariablen handelt. Dies kommt dann vor, wenn die bedingten Wahrscheinlichkeiten sich aus beliebigen Häufigkeitsverteilungen zusammensetzen. In diesem Fall werden die Werte der relativen Häufigkeiten den einzelnen Feldkomponenten zugewiesen.

6.9 Übergangswahrscheinlichkeiten

Für die iterative Berechnung von Zustandswahrscheinlichkeiten in Markovketten aus früheren Zustandswahrscheinlichkeiten und Zustandsübergangswahrscheinlichkeiten werden vom Modul 'RndVar7' die Prozeduren 'Uebergang' und 'UebergangSparse' zur Verfügung gestellt. Ihre Parameter werden in Abschnitt 5.15.1 bzw. Abschnitt 5.15.2 beschrieben. Eine Anwendung zeigt das folgende Beispiel, wobei die Zustandsmenge Teilmenge von $I\!N \times I\!N$ ist, wie es bei einem Modell mit einem Bediener und zwei Warteschlangen der Fall ist.

Beispiel:

Gegeben ist eine Markovkette mit der Zustandsmenge $\Omega_\mathbf{S} \subset I\!N \times I\!N$ und der Zustandsübergangsmatrix $\mathbf{P}$. Wir betrachten die Markovkette zu den beiden aufeinanderfolgenden Zeitpunkten t und $t+1$, $t \in I\!N$. Der Zustand der Markovkette zu diesen beiden Zeitpunkten wird durch die beiden Zufallsvariablen $\mathbf{S}^{(t)}$ und $\mathbf{S}^{(t+1)}$ beschrieben, deren beider Wertmenge $\Omega_\mathbf{S}$ ist. Durch die Verwendung einer Numerierung der Zustände, die durch die bijektive Funktion $f : \Omega_\mathbf{S} \to \{1, \dots, |\Omega_\mathbf{S}|\}$ realisiert wird, lassen sich die Zustandswahrscheinlichkeiten $P\{\mathbf{S}^{(t)} = \mathbf{s}\}$, $\mathbf{s} \in \Omega_\mathbf{S}$, und $P\{\mathbf{S}^{(t+1)} = \mathbf{s}'\}$, $\mathbf{s}' \in \Omega_\mathbf{S}$, zu den Spaltenvektoren $\mathbf{p}^{(t)}$ und $\mathbf{p}^{(t+1)}$ zusammenfassen, deren Komponenten durch

$$p^{(t)}_{f(\mathbf{s})} := P\{\mathbf{S}^{(t)} = \mathbf{s}\}$$

und

$$p^{(t+1)}_{f(\mathbf{s})} := P\{\mathbf{S}^{(t+1)} = \mathbf{s}'\}$$

festgelegt sind. Unter Verwendung der gleichen Numerierung sind die Elemente $p_{i,j}$ der Zustandsübergangsmatrix $\mathbf{P}$ definiert mit

$$p_{f(\mathbf{s}),f(\mathbf{s}')} := P\{\mathbf{S}^{(t+1)} = \mathbf{s}' \mid \mathbf{S}^{(t)} = \mathbf{s}\}.$$

Die Verteilung der Zufallsvariablen $\mathbf{S}^{(t)}$ sei bekannt. Zu berechnen ist die Verteilung von $\mathbf{S}^{(t+1)}$, wobei im folgenden Beispiel als Zustandsmenge $\Omega_\mathbf{S}$ die Menge $\{0, 1, \dots, 5\}^2$ gewählt wird.

Definitionen:

$\mathbf{S}^{(t)}, \mathbf{S}^{(t+1)}$: Zufallsvariablen mit $\Omega_\mathbf{S} = \{0, 1, \dots, 5\}^2$;

$f : \Omega_\mathbf{S} \to \{1, \dots, |\Omega_\mathbf{S}|\}$: bijektive Funktion zur Zustandsnumerierung;

$\mathbf{p}^{(t)}, \mathbf{p}^{(t+1)}$: Vektoren von Zustandswahrscheinlichkeiten mit den Komponenten
p_i, $i \in \{1, \dots, |\Omega_\mathbf{S}|\}$;

$\mathbf{P}$: $|\Omega_\mathbf{S}| \times |\Omega_\mathbf{S}|$-Zustandsübergangsmatrix;

Operationen:

$p^{(t)}_{f(\mathbf{s})} := P\{\mathbf{S}^{(t)} = \mathbf{s}\}$; für alle $\mathbf{s} \in \Omega_\mathbf{S}$
$\mathbf{p}^{(t+1)} := \mathbf{P}^\top \cdot \mathbf{p}^{(t)}$;
$P\{\mathbf{S}^{(t+1)} = \mathbf{s}\} := p^{(t+1)}_{f(\mathbf{s})}$; für alle $\mathbf{s} \in \Omega_\mathbf{S}$

Dabei bezeichnet $\mathbf{P}^\mathsf{T}$ die Transponierte der Zustandsübergangsmatrix.

Programm:

```
MODULE ...;
FROM SYSTEM   IMPORT ADR;
FROM Reals    IMPORT Real;
FROM Tupel    IMPORT ITupel,BildeITupel2,BildeITupel4;
FROM RVBasics IMPORT ZufallsVar,InitDskrZVMD;
FROM RndVar7  IMPORT Uebergang;
...
VAR S,P   : ZufallsVar;
    pS    : ARRAY [0..5],[0..5] OF Real;
    pP    : ARRAY [0..5],[0..5],[0..5],[0..5] OF Real;
    ug,og : ITupel;
    ...
BEGIN
  BildeITupel2(ug,0,0);
  BildeITupel2(og,5,5);
  InitDskrZVMD(S,'S ',ADR(pS),ug,og);
  BildeITupel4(ug,0,0,0,0);
  BildeITupel4(og,5,5,5,5);
  InitDskrZVMD(P,'P ',ADR(pP),ug,og);

  (* Festlegung der pP[i1,i2,j1,j2];                                   *)
  (* Festlegung von aktuellen Zustandswahrscheinlichkeiten pS[i1,i2]; *)
  ...
  Uebergang(S,P);  (* Berechnung der neuen Zustandswahrsch. pS[i1,i2] *)
  ...
```

In diesem Programmabschnitt fällt auf, daß sowohl die Numerierungsfunktion f als auch die beiden Vektoren $\mathbf{p}^t$ und $\mathbf{p}^{t+1}$ fehlen. Die Numerierungsfunktion wird durch den Compiler selbst realisiert, indem er zweidimensionalen (und mehrdimensionalen) Feldern auf festgelegte Weise Arbeitsspeicher aus dem Adreßbereich eines Rechners zuordnet. Aus dem gleichen Grund kann das mehrdimensionale Verteilungsfeld pS wie ein Vektor $\mathbf{p}$ benutzt werden.

Beispiel:

Das vorangegangene Beispiel wird wie folgt abgewandelt:

Die Zustandsübergangswahrscheinlichkeiten $p_{f(i_1,i_2),f(j_1,j_2)}$ seien null für alle Paare (s_1, s_2), $(s'_1, s'_2) \in \Omega_\mathbf{S}$, mit $s_2 \neq s'_2$. Alle anderen, im allgemeinen positiven Zustandsübergangswahrscheinlichkeiten können dann in der $|\Omega_\mathbf{S}| \times |\Omega_{S_1}|$-Matrix $\mathbf{P}'$ zusammengefaßt werden mit $\Omega_{S_1} = \{0, 1, \ldots, 5\}$. Ihre Komponenten seien festgelegt durch

$$p'_{f(s_1,s_2),s'_1} := p_{(s_1,s_2),(s'_1,s'_2)}.$$

Die Berechnung der Zustandswahrscheinlichkeiten des Folgezustands kann dann, wie im nachfolgenden Programm, durch die Prozedur 'UebergangSparse' vorgenommen werden.

Definitionen:

$\mathbf{S}^{(t)}, \mathbf{S}^{(t+1)}$: Zufallsvariablen mit $\Omega_{\mathbf{S}} = \{0, 1, \ldots, 5\}^2$;
$f : \Omega_{\mathbf{S}} \rightarrow \{1, \ldots, |\Omega_{\mathbf{S}}|\}$: bijektive Funktion zur Zustandsnumerierung;
$\mathbf{P}'$: $|\Omega_{\mathbf{S}}| \times |\Omega_{S_1}|$-Matrix von Zustandsübergangswahrscheinlichkeiten wie oben angegeben;

Operationen:

```
for all $(s_1, s_2) \in \Omega_{\mathbf{S}}, s'_1 \in \Omega_{S_1}$ do
    $P\{\mathbf{S}^{(t+1)} = (s'_1, s_2)\} := \sum_{s_1 \in \Omega_{S_1}} P\{\mathbf{S}^{(t)} = (s_1, s_2)\} \cdot p_{f(s_1, s_2), s'_1}$;
od;
```

Programm:

```
MODULE ...;

FROM SYSTEM   IMPORT ADR;
FROM Reals    IMPORT Real;
FROM Tupel    IMPORT ITupel,BildeITupel2,BildeITupel3;
FROM RVBasics IMPORT ZufallsVar,InitDskrZVMD;
FROM RndVar7  IMPORT UebergangSparse;
...

VAR S,Ps : ZufallsVar;

    pS  : ARRAY [0..5],[0..5] OF Real;
    pPs : ARRAY [0..5],[0..5],[0..5] OF Real;

    ug,og: ITupel;
    ...
BEGIN

  BildeITupel2(ug,0,0);
  BildeITupel2(og,5,5);
  InitDskrZVMD(S,'S ',ADR(pS),ug,og);

  BildeITupel3(ug,0,0,0);
  BildeITupel3(og,5,5,5);
  InitDskrZVMD(Ps,'Ps',ADR(pPs),ug,og);

  (* Festlegung der pPs[i1,i2,j1];                                      *)
  (* Festlegung von aktuellen Zustandswahrscheinlichkeiten pS[i1,i2];    *)
  ...
  UebergangSparse(S,Ps,1);  (* Berechnung der neuen Zustandsw. pS[i1,i2] *)
                            (* Der dritte Parameter gibt die Nummer      *)
                            (* der Komponente im Zustandsvektor an,      *)
                            (* die sich aendern kann                     *)
  ...
```

6.10 Ausgabe von Verteilungen

Zuletzt seien noch einige Beispiele für die Benutzung der Ausgaberoutinen für Zufallsvariablen angegeben. Den Beispielen kann entnommen werden, wie tabellarische Ausgaben von Verteilungen in Dateien umgeleitet und graphische Darstellungen eindimensionaler Verteilungen in LaTeX-Dokumente eingebunden werden.

Beispiel:

Die Verteilung der Zufallsvariable X soll in tabellarischer Form in eine Datei mit Namen 'VertX.txt' ausgegeben werden. Die Struktur des Wertebereichs von X kann beliebig gewählt sein.

Programm:

```
MODULE ...;
FROM SYSTEM   IMPORT ADR;
FROM Reals    IMPORT Real;
FROM IOProc   IMPORT OpenOutput,CloseOutput;
FROM RVBasics IMPORT ZufallsVar;
FROM RndVar1  IMPORT WriteZV;
...
VAR X : ZufallsVar;
    pX: ARRAY ...... OF Real;
    fX: ARRAY ...... OF Real;
    ...
BEGIN
  ...
  OpenOutput('VertX.txt');
  WriteZV(X,'Verteilung von X',TRUE);
  CloseOutput;
  ...
```

Nach einer 'OpenOutput'-Anweisung können natürlich auch mehrere Verteilungen hintereinander in dieselbe Datei ausgegeben werden.

Beispiel:

Die eindimensionale Zufallsvariable C mit gemischtem Wertebereich soll graphisch im PostScript-Format ausgegeben und in ein LaTeX-Dokument integriert werden. Die Größe der Abbildung soll 60 mm in der Höhe und 100 mm in der Breite betragen.

Programm:

```
MODULE ...;
FROM SYSTEM   IMPORT ADR;
FROM Reals    IMPORT Real;
FROM RVBasics IMPORT ZufallsVar;
FROM RndVar1  IMPORT WriteZVtoPS;
...
VAR C : ZufallsVar;
    pC: ARRAY ...... OF Real;
    fC: ARRAY ...... OF Real;
    ...
BEGIN
  ...
  WriteZVtoPS(C,'VertC','Verteilung von C',60.0,100.0);
  ...
```

Durch den Aufruf von 'WriteZVtoPS' werden zwei Dateien erzeugt. Die erste, mit Namen 'VertC.ps', enthält alle PostScript-Befehle, die für die Ausgabe der graphischen Darstellung auf einem PostScript-fähigen Drucker notwendig sind. Sie kann dabei direkt von solch einem Drucker verarbeitet werden.

Die zweite erzeugte Datei, mit Namen 'VertC.tex', erleichtert dem Benutzer die Einbindung der Graphik in ein LaTeX-Dokument. Die Datei hat im obigen Fall folgenden Inhalt:

```
\begin{figure}[htbp]
  \begin{center}
  \setlength{\unitlength}{0.04in}
  \begin{picture}(100,60)(0,-60)
    \special{insert VertC.ps}
  \end{picture}
  \end{center}
  \caption{Verteilung von C}
  \label{VertC}
\end{figure}
```

Die Datei kann nachträglich bearbeitet werden, um zum Beispiel die Bildunterschrift oder den Namen des Labels zu ändern.

Die Einbindung der Graphik in ein Dokument wird mit dem Kommando

```
\input{VertC}
```

an gewünschter Stelle im Text vorgenommen. Mit dem Befehl

```
\ref{VertC}
```

kann im Text auf die Abbildung Bezug genommen werden. Nach dem Übersetzen der LaTeX-Datei werden Dokument und Graphik durch die Verwendung eines PostScript-fähigen Treibers (z.B. DVILZR) zusammengebunden und können ausgedruckt werden.

Literatur

(long)

[Dal84] M. Dal Cin: Programmierung in MODULA-2, B.G. Teubner, Stuttgart, 1984

[He91] M. Heckler: Wartezeitverteilungen in zyklischen Pollingsystemen, Diplomarbeit, Institut für Informatik, Universität Bonn, 1991

[Ne93] B. Neuenfels: Eine Approximation diskreter Verteilungen für die numerische Analyse von Bedienmodellen, Diplomarbeit, Institut für Informatik, Universität Bonn, 1993

[Sa98] W. Sandmann: Wartezeitverteilungen in Tandem–Warteschlangennetzen mit stückweise approximierenden Dichten maximaler Entropie, Diplomarbeit, Institut für Informatik, Universität Bonn, 1998

[Schm90] M. Schmitz: Numerische Analyse eines zyklischen Pollingsystems unter Berücksichtigung globaler Zustände, Diplomarbeit, Institut für Informatik, Universität Bonn, 1990

[Si91] D. Siepmann: VERENA - Eine Programmbibliothek zum Rechnen mit Wahrscheinlichkeitsverteilungen und ihre Erprobung an Pollingsystemen, Diplomarbeit, Institut für Informatik, Universität Bonn, 1991

[Sp82] P. P. Spies: Grundlagen stochastischer Modelle, Hanser Studienbücher, München, Wien, 1982

[St88] J. Ch. Strelen: Piecewise Approximation of Densities applying the Principle of Maximum Entropy: Waiting Times in G/G/1-Systems, Interner Bericht II/88/2, Institut für Informatik, Universität Bonn, 1988

[St89] J. Ch. Strelen, Piecewise approximation of densities applying the principle of maximum entropy: Waiting times in G/G/1-systems, Modelling Techniques and Tools for Computer Performance Evaluation, S. 421-438, R. Puigjaner und D. Poitier (Hrsg.), Plenum Press, New York, 1989

[SB89] J. Ch. Strelen und B. Bärk, An approach to the numerical analysis of multiple-queue, cyclic service systems, in: G. Stiege und J. S. Lie (Hrsg.), Messung, Modellierung und Bewertung von Rechensystemen und Netzen , S. 75-88, Springer-Verlag, Berlin, 1989

[St91] J. Ch. Strelen, Iterative Analyse von Markov-Modellen mit alternierender Aggregation und Disaggregation, in: A. Lehmann und F. Lehmann (Hrsg.), Messung, Modellierung und Bewertung von Rechensystemen , S. 320-336, Springer-Verlag, Berlin, 1991

[Wi84] N. Wirth: Programmieren in MODULA-2, Springer-Verlag, Berlin, 1984