

Organisierte Komplexität

**Mikroprozess-Analysen der Interaktionsdynamik zweier
Psychotherapien mit den Methoden der nichtlinearen
Zeitreihenanalyse**

Inaugural-Dissertation
in der Fakultät Psychologie
der Otto-Friedrich-Universität Bamberg

vorgelegt von

Guido Strunk

aus

Dorsten

Bamberg, den 11.06.2004

Tag der mündlichen Prüfung: 19.01.2005

Dekan: Univ. Prof. Dr. Max Peter Baumann

Erstgutachter: Apl. Prof. Dr. Günter Schiepek

Zweitgutachter: Apl. Prof. Dr. Harald Schaub

8.2.7 Chaos und Ordnung in nominalen Datensätzen

Die bisher besprochenen zeitreihenanalytischen Verfahren stellen allesamt hohe Anforderungen an die Datenqualität der zu untersuchenden Zeitreihen. Neben zum Teil erheblichen Zeitreihenlängen werden auch hohe Anforderungen an die Qualität und Auflösung der Daten gestellt. So sollten die Zeitreihen möglichst rauschfrei und mit möglichst hoher Auflösung erhoben worden sein. Als Skalenniveau sollte zumindest Intervallskalenqualität vorliegen.

Inzwischen sind jedoch Verfahren vorgeschlagen worden, die bereits für nominale Datensätze brauchbare Informationen über die Komplexität und Ordnung der erfassten Prozesse liefern können. Im Folgenden sollen die Grundideen solcher Verfahren ausführlicher dargestellt werden.

8.2.7.1 Problemstellung

Im Rahmen der Darstellung verschiedener mathematischer Methoden zur Bestimmung von Kennwerten nichtlinearer dynamischer Systeme wurde bereits mehrfach auf die von Shannon formulierte Definition des Informationsgehalts von Werteabfolgen eingegangen (vgl. Gleichung 48 und S. 359f.; Shannon 1948). Der Informationsgehalt einer Werteabfolge ergibt sich danach als Summe des Produktes der Wahrscheinlichkeit für das Auftreten eines Wertes und dem Logarithmus für diese Wahrscheinlichkeit. Gleichung 48 sei hier zur Verdeutlichung noch einmal angeführt:

$$I_s = -\sum_{i=1}^N P(s_i) \log_2 P(s_i).$$

Der Informationsgehalt des folgenden Satzes lässt sich nach dieser Gleichung mit 3,78 Bit bestimmen (vgl. Tabelle 11).

Wie hoch ist der Informationsgehalt dieses Satzes?

Shannons Informationsdefinition berücksichtigt keine dynamischen Aspekte

Obwohl die Shannonsche Informationsdefinition zu den wichtigsten Standardberechnungen für den Informationsgehalt von Werteabfolgen gehört, handelt es sich bei dem Verfahren nur um eine unvollständige Charakterisierung. Es zeigt sich nämlich, dass jede zufällige Abfolge der Buchstaben des genannten Satzes zu dem gleichen Informationsgehalt führt. In diesem Sinne unterscheidet sich der vorgestellte Satz nicht von der folgenden Buchstabenfolge:

eeaiiesohIettcrshoszatsWiamrfsohnngnddtSleei

Damit stellt sich die Frage, wie der Informationsbegriff um die Berücksichtigung der dynamischen Ordnung der Zeichenabfolgen erweitert werden kann. Eine Möglichkeit liegt darin, die Berechnung der Wahrscheinlichkeiten nicht auf ein einzelnes Symbol zu beschränken, sondern auf die Abfolge von Symbolen, also auf Zeichenpaare, Dreier- oder Viererkombinationen zu erweitern, wie es oben schon mehrfach vorgeschlagen wurde (vgl. S. 359ff. und S. 377).

Zeichen	Häufigkeit	P(Zeichen)	$\log_2(P(\text{Zeichen}))$	Produkt
a	3	0,0698	-3,8406	-0,2681
c	1	0,0233	-5,4235	-0,1264
d	2	0,0465	-4,4266	-0,2058
e	6	0,1395	-2,8417	-0,3964
f	1	0,0233	-5,4235	-0,1264
g	1	0,0233	-5,4235	-0,1264
h	3	0,0698	-3,8406	-0,2681
i	5	0,1163	-3,1041	-0,3610
l	1	0,0233	-5,4235	-0,1264
m	1	0,0233	-5,4235	-0,1264
n	2	0,0465	-4,4266	-0,2058
o	3	0,0698	-3,8406	-0,2681
r	2	0,0465	-4,4266	-0,2058
s	6	0,1395	-2,8417	-0,3964
t	4	0,0930	-3,4266	-0,3187
w	1	0,0233	-5,4235	-0,1264
z	1	0,0233	-5,4235	-0,1264
Summe:	43	1,0000		-3,7788

Tabelle 11: Bestimmung des Informationsgehaltes des Satzes: „Wie hoch ist der Informationsgehalt dieses Satzes?“

Der Informationsgehalt des Satzes ergibt sich nach der Shannonschen Informationsdefinition über die Häufigkeitsverteilung der im Satz verwendeten Buchstaben und beläuft sich auf 3,78 Bit.

8.2.7.2 Abfolgen von Symbolen in Zeitreihen: Symbol Dynamics

Unter dem Namen *Symbol Dynamics* sind eine ganze Reihe von Verfahren vorgeschlagen worden, um die Begrenzungen der Shanonnischen Informationsdefinition zu durchbrechen (für einen Überblick: Collet & Eckmann 1980). In ihren Grundlagen gehen diese Ansätze bis auf das Jahr 1898 zurück (Hadamard 1898). Helund und Morse erweiterten den Ansatz von Hadamard um Aspekte periodischen Verhaltens in klassischen dynamischen Systemen (Morse 1921, Morse & Hedlund 1938). Im Folgenden sollen einige Methoden des Symbol Dynamics Ansatzes kurz dargestellt werden:

- **Sequenzen fester Länge:** Eine einfache Methode für die Bestimmung des Informationsgehalts einer dynamischen Symbolfolge ist die Festsetzung einer Sequenzlänge m , die die Länge eines so genannten *Wortes* definiert. Ein solches Wort besteht also aus der Abfolge von R Werten der zu untersuchenden Symbolreihe. Werden alle in der Symbolreihe vorhandenen Worte gebildet und ihre Häufigkeit bestimmt, so lässt sich der Informationsgehalt nach der Shanonnischen Informationsdefinition für die Worte bestimmen. Ein Problem dieses Verfahrens ist jedoch die Wahl eines geeigneten R . Wird R zu groß gewählt, dann kommen viele Worte nur noch einmal in der Symbolreihe vor. Wird R jedoch zu klein gewählt, werden nur wenig Informationen über die spezielle Anordnung der Symbole genutzt. In der Praxis der Anwendung dieses Maßes empfiehlt sich daher die Berechnung für verschiedene R durchzuführen und danach eines auszuwählen, das sowohl möglichst große Bereiche der Dynamik erfasst aber nicht dazu führt, dass zu viele Worte nur einmal vorkommen.

- **Häufigkeitsverteilung von Worten:** Obwohl der Informationsgehalt einer Symbolreihe, die aus n Worten der Länge R besteht direkt berechnet werden kann, empfiehlt es sich, die Häufigkeitsverteilung dieser Worte genauer zu betrachten. Handelt es sich bei der Symbolreihe um eine rein stochastische Anordnung von Symbolen, so sollten relativ unabhängig von der gewählten Wortlänge R alle Worte gleich häufig in der Symbolreihe vorkommen. Weicht die Häufigkeitsverteilung der Worte signifikant von einer Gleichverteilung ab, ist die Symbolreihe wahrscheinlich nicht zufällig.
- **Sequenzen ohne Veränderung:** Eine weitere Möglichkeit Informationen über die geordnete Komplexität einer Symbolfolge zu gewinnen besteht darin, den Prozentsatz von Worten zu bestimmen, die nur ein immer gleiches Symbol enthalten. Wird die Wortlänge zum Beispiel mit fünf festgelegt und besteht die Symbolfolge nur aus den Symbolen ‚0‘ und ‚1‘, so werden alle Worte gezählt, die entweder nur aus ‚0‘ oder nur aus ‚1‘ bestehen.
- **Forbidden Words:** Im Falle einer einfachen periodischen Ordnung in der Symbolreihe wiederholen sich die Worte fortlaufend. Andere Worte als die, die zur Beschreibung einer Periode dienen, werden nicht benötigt und sollten in der Symbolreihe auch nicht vorkommen. In diesem Sinne scheint es für jedes deterministische, nicht stochastische System eine Menge von Worten zu geben, die in der Symbolreihe eigentlich nicht vorkommen dürfen. Praktisch ist es jedoch kaum möglich alle diese Worte vorab zu bestimmen, um dann zu überprüfen, ob sie auch tatsächlich nicht in der Symbolreihe vorkommen. Es lässt sich jedoch bestimmen, wie viele Worte nur sehr selten, d.h. z.B. mit einer Wahrscheinlichkeit von nur 0,1% in der Symbolreihe vorkommen. Die Anzahl solcher seltener Symbole ist dann ein Maß für die Komplexität der Symbolreihe.

Ein weiteres Verfahren zur Bestimmung der Ordnung in Symbolreihen wurde von Strunk (1996b; vgl. auch Strunk & Schiepek 2002) vorgeschlagen: Ähnlich wie bei der Bestimmung der K2-Entropie (siehe oben, S. 378ff.) werden Sequenzen von Wertefolgen definiert, wobei sich die Sequenzen jedoch nicht auf Punkte im Phasenraum sondern auf Messwerte der Zeitreihe beziehen. So werden zunächst Zweiersequenzen von Messwertfolgepunkten gebildet und auf ihr wiederholtes Vorkommen in der Zeitreihe hin untersucht. Jede Sequenz wird dabei mit einem Symbol (z.B. einer fortlaufenden Nummerierung) eindeutig bezeichnet. Das Verfahren bildet dann weitere Sequenzen wachsender Länge. Die Länge der Sequenz, die in der Zeitreihe gerade noch zweimal vorkommt ist dann ein Maß für die Komplexität der Zeitreihe. Als Kriterium zum Vergleich des Ergebnisses mit Zufallsabfolgen wird die Ausgangszeitreihe randomisiert und erneut die Komplexität nach dem beschriebenen Verfahren berechnet. Die Komplexität vieler randomisierter Zeitreihen folgt dabei einer Normalverteilung, sodass eine Statistik dafür angegeben werden kann, ob die Komplexität der Originalzeitreihe geordneter ausfällt als die der Surrogate. Das Verfahren hat eine große Ähnlichkeit zu Algorithmen, wie sie auch von anderen Autoren vorgeschlagen wurden (z.B. Stevens 1939, Mood 1940 für einen Überblick siehe: Bortz, Lienert & Boehnke 2000).

8.2.7.3 Die algorithmische Entropie

Einen ganz anderen Weg beschreiten Verfahren der algorithmischen Entropie um die von Shannon formulierte Informationsdefinition um die Berücksichtigung dynamischer Strukturen zu bereichern. Es handelt sich dabei jeweils um spezifische Implementierungen eines Verfahrens zur Bestimmung der so genannten *algorithmischen Entropie*. Die Grundbegriffe der algorithmischen Entropie beruhen auf Arbeiten zur algorithmischen Informationstheorie (Kolmogorov 1965, Zvonkin & Levin 1970, Chaitin 1974), die den Informationsgehalt einer Werteabfolge nach dem Informationsgehalt bemisst, der benötigt wird um die Werteabfolge zu beschreiben. In diesem Sinne ist die Wurzel aus zwei eine Zahl mit unendlich vielen Nachkommastellen, die zudem in einer extrem komplexen Abfolge auftreten. Trotz dieser augenscheinlichen Komplexität der Ziffernabfolgen kann diese durch einen einfachen Algorithmus ermittelt werden. Zur Beschreibung bzw. Erzeugung der Zahlenabfolge genügt also eine relativ einfache Formel, die bereits an anderer Stelle dieser Arbeit präsentiert wurde (vgl. Gleichung 15, S. 164).

Eine deterministische Abfolge von Symbolen sollte sich in der Regel durch einen Algorithmus erzeugen lassen, der einfacher ist als die Abfolge selbst

Die algorithmische Informationstheorie bezieht sich auf diese Überlegungen, indem davon ausgegangen wird, dass zur Beschreibung bzw. Erzeugung bestimmter komplexer aber geordneter Strukturen die Kenntnis eines einfachen Algorithmus genügt. Es ist der Informationsgehalt dieses Algorithmus, der ausreichend ist, um eine, in der Regel weit komplexere, dynamische Ordnung zu generieren. Etwas vereinfachend ist die algorithmische Entropie einer Werteabfolge also gegeben durch die mindestens erforderliche Größe eines Algorithmus, der diese Abfolge erzeugen kann.

Für die Beschreibung von Zufallsfolgen gibt es keine Abkürzungen

Während Strukturen, die sich durch eine geordnete Komplexität auszeichnen in der Regel auf einfachere Algorithmen zurückgeführt werden können, gilt dies nicht für die zufällige Abfolge von Werten. Für einen solchen Fall kann kein Algorithmus angegeben werden, der einfacher ist als die Werteabfolge selbst. Wird also ein Algorithmus zur Beschreibung einer Werteabfolge benötigt, der ebenso komplex ist wie die Werteabfolge selbst, so liegt eine maximale algorithmische Entropie und damit ein Zufallsprozess vor (Hubermann & Hogg 1986).

Auf ähnlichen Überlegungen beruhen Computerprogramme, die zur Komprimierung von Dateien benutzt werden. Eine komprimierte Datei ist im Wesentlichen eine Erzeugungsvorschrift für die Datei selbst. Gelingt die Komprimierung, so liegt ein Algorithmus vor, der viel weniger Speicherplatz einnimmt als die ursprüngliche Datei, aber alle Informationen enthält, um die Datei wieder zu erzeugen. Der Unterschied zwischen der Größe der komprimierten und der unkomprimierten Datei ist dabei ein Maß ihrer algorithmischen Entropie.

Die algorithmische Entropie ergibt sich aus der erfolgreichen Anwendung von Komprimierungsverfahren

Berechnungsmöglichkeiten für die algorithmische Entropie ergeben sich also durch die Anwendung von Verfahren zur verlustfreien Komprimierung von Dateien. Ein solcher Komprimierungsalgorithmus ist z.B. die *Grammar Complexity* (Ebeling & Jiménez-Montano 1980, Jiménez-Montano 1984).

8.2.7.3.1 Grammar Complexity

Am besten wird der Algorithmus der Grammar Complexity verständlich, wenn er anhand eines Beispiels erläutert wird (das Beispiel folgt im Wesentlichen den Ausführungen von Rapp et al. 1991):

Gegeben sei eine Wertefolge x_i mit folgenden Werten:

$$x = 0101101011001001.$$

In einem ersten Schritt sucht der Algorithmus nach sich in der Zeitreihe wiederholenden Paaren, die durch ein Symbol ersetzt werden. Das Symbol und seine Bedeutung wird in eine Symbolliste eingetragen. Wenn keine Paare mehr gefunden werden, die mehr als zwei Mal in der Zeitreihe vorkommen, wird nach Tripeln gesucht, die mindestens zwei Mal in der Zeitreihe vorkommen. Auch diese werden durch ein Symbol ersetzt. In den folgenden Schritten werden mit aufsteigendem n n -Tupel gesucht und durch Symbole ersetzt, wenn es jeweils zwei gleiche davon geben sollte. Der Algorithmus bricht ab, wenn keine weitere Ersetzung mehr möglich ist. Für die als Beispiel gegebene Wertefolge ergibt sich zunächst, dass die Abfolge (0,1) mehrfach vorkommt. Sie wird durch das Symbol a ersetzt:

$$\begin{aligned} x &= a a 1 a a 1 0 a 0 a \\ a &= 0 1. \end{aligned}$$

Da keine weiteren Paare gefunden werden können, die mindestens drei Mal in der Wertefolge vorkommen wird nun nach Tripeln gesucht. Es zeigt sich, dass $(a, a, 1)$ zwei Mal in der Zeitreihe identifiziert werden können. Im Gegensatz zu Paaren, bedeutet das Auffinden von gleichen Abfolgen mit mehr als nur zwei Elementen eine Verkürzung der Zeitreihe, sodass $(a, a, 1)$ durch b ersetzt wird.

$$\begin{aligned} x &= b b 0 a 0 a \\ a &= 0 1 \\ b &= a a 1. \end{aligned}$$

Dass eine Ersetzung von *zwei* Paaren durch *ein* Symbol zu keiner Verkürzung führt, wird deutlich, wenn bedacht wird, dass die Paare jeweils durch ein Symbol ersetzt werden und in der Ersetzungstabelle dieses Symbol durch die beiden Werte des Paares repräsentiert werden müssen. Insgesamt werden also vier Elemente benötigt um die Wertefolge aus der Komprimierung erneut zu erzeugen. Vier Elemente waren es jedoch bereits vor der Komprimierung.

Für die Beispielzeitreihe lässt sich nach der letzten Ersetzung kein weiteres Tripel und auch kein 4-Tupel finden, sodass die Komprimierung hier abbricht. Als Vereinfachung können sich wiederholende Elemente noch als Potenzen geschrieben werden, sodass sich aus $b b$ in Potenzschreibweise b^2 ergibt (würde $b b b$ vorliegen würde daraus b^3):

$$\begin{aligned} x &= b^2 0 a 0 a \\ a &= 0 1 \\ b &= a^2 1. \end{aligned}$$

Es folgt die Berechnung der Grammar Complexity, die gegeben ist als Summe der verbleibenden Symbole, wobei auch die Symbole rechts des Gleichheitszeichens der Ersetzungstabelle mitgezählt werden. Ebenfalls hinzugezählt wird der Betrag des Logarithmus zur Basis zwei, jeder verwendeten Potenz. Insgesamt enthält x

fünf und die beiden Symbole der Ersetzungstabelle zusammen vier Elemente. Zudem sind zwei Potenzen mit jeweils dem Wert zwei vorhanden. Zusammen ergibt sich:

$$\text{Grammar Complexity} = 5 + 2 + 2 + |\log_2 2| + |\log_2 2| = 11.$$

Für die Interpretation des ermittelten Wertes bieten sich zwei Verfahren an. Zum einen kann aus der ursprünglich gegebenen Wertefolge durch Sortierung eine völlig geordnete Symbolreihe erstellt werden:

$$x' = 11111111100000000.$$

Die Grammar Complexity wird interpretierbar im Vergleich mit maximaler Ordnung

Die Grammar Complexity für diese geordnete Abfolge beläuft sich auf 10 und weist sie damit auch als geordneter aus. Allerdings ist der Unterschied zwischen den beiden ermittelten Kennwerten nicht sehr groß. Dies liegt daran, dass die Wertabfolge für dieses Beispiel auch sehr kurz gewählt wurde. Bei längeren Zeitreihen fallen die Unterschiede weit stärker aus.

Die Grammar Complexity wird interpretierbar im Vergleich mit zufälligen Surrogaten

Eine weitere Möglichkeit zur Interpretation der Ergebnisse wurden von Tschacher und Scheier (1995) vorgeschlagen. Sie testeten die Daten mittels eines Surrogatdatenverfahrens. Dazu bildeten sie 50 verschiedene Surrogatzeitreihen, die aus der durcheinandergewürfelten Ursprungszeitreihe bestehen und berechnen für diese ebenfalls die Grammar Complexity. Die ermittelten 50 Werte der Grammar Complexity bilden eine Normalverteilung, sodass die zu überprüfende Zeitreihe auf signifikante Unterschiede gegenüber den Surrogatdaten getestet werden kann (Tschacher & Scheier 1995).

Insgesamt bietet sich die Grammar Complexity zur Komplexitätsbestimmung nominaler Wertabfolgen auch in der Psychotherapieforschung an (vgl. Rapp et al. 1991, Tschacher & Scheier 1995, Friedlmayer, Reznicek & Strunk 1996, Thiele 1997). Zu beachten gilt jedoch, dass die Länge der untersuchten Zeitreihe ebenso einen Einfluss auf das Ergebnis hat, wie auch die Werteverteilung in der Zeitreihe.

Neben dem Verfahren der Grammar Complexity sind noch andere Algorithmen vorgeschlagen worden, die auf Verfahren der Datenkompression beruhen (für eine Vergleichende Darstellung siehe z.B. Schürmann & Grassberger 1996). Von besonderer Bedeutung sind in diesem Zusammenhang die Algorithmen LZ77 und LZ78, die von Lempel und Ziv vorgeschlagen wurden (Ziv & Lempel 1977, Lempel & Ziv 1978).

8.2.7.3.2 Ziv-Lempel Algorithmen

Während es sich bei der Grammar Complexity um einen sog. kontextfreien Algorithmus handelt, ergibt sich die Komplexität einer Symbolabfolge bei den von Ziv und Lempel vorgeschlagenen Algorithmen aus einer Art Lernprozess, bei der die Algorithmen ein Wörterbuch anlegen, welches im Verlauf der Symbolreihe wächst und dabei immer auf dem aufbaut, was die Symbolreihe bisher an Abfolgen enthielt (Ziv & Lempel 1977, Lempel & Ziv 1978).

Wie auch bei der Grammar Complexity lassen sich die von Lempel und Ziv vorgeschlagenen Algorithmen am besten durch Beispiele verdeutlichen, wobei im

Folgenden zunächst allgemein auf das Problem der Datenkomprimierung eingegangen wird. Ist die folgenden Abfolge von Symbolen gegeben:

$$x = a a a b a a b a a a,$$

so lässt sie sich mit folgender Symboltabelle leicht als Ziffernfolgen von Null und Eins kodieren:

$$\begin{aligned} a &= 0 \\ b &= 10 \\ aaa &= 11 \end{aligned}$$

Die resultierende Zahlenfolge ist dann:

$$11010111011.$$

Während die Originalfolge aus 12 Symbolen besteht, werden nach der Übersetzung durch die Symboltabelle nur mehr 11 Symbole benötigt. Die Symbolreihe wurde also komprimiert. Die Frage die sich in diesem Zusammenhang stellt ist jedoch: Warum funktioniert das? – und etwas wichtiger, unter welchen Bedingungen funktioniert das?

Würde man Jemandem die oben genannte Symbolfolge vorlesen, so könnte man z.B. sagen: vier mal a , dann zwei mal $b a a a$. In diesem Sinne geschieht eine Kompression der Daten über das Erkennen von sich wiederholenden Symbolfolgen. Das Problem der Datenkomprimierung lässt sich demnach verdichten zu der Frage, wie auf der Grundlage eines festen Algorithmus eine Regel angegeben werden kann, die solche Wiederholungen identifiziert.

Der LZ78 Algorithmus (Lempel & Ziv 1978) gibt eine solche Regel an, indem eine gegebene Symbolreihe so in Abschnitte zerlegt wird, das ein neuer Abschnitt jeweils den kürzest möglichen neuen Abschnitt bildet. Durch diese Vorschrift wird für die folgende Abfolge von Symbolen,

$$x = a b a b b a b a a b a a b b a b a b$$

als kürzest möglicher neuer Abschnitt a identifiziert. Als kürzester neuer Abschnitt folgt auf a ein b . Das dritte Symbol (a) kam bereits einmal vor, sodass ein Abschnitt, der erneut aus a besteht, kein neuer Abschnitt wäre. Der kürzeste neue Abschnitt wäre also nicht erneut a sondern $a b$. Auf diesen Abschnitt folgt ein b , welches jedoch auch schon einmal vorkam, sodass der nächste Abschnitt aus $b a$ gebildet wird. Erneut ist das nächste Symbol ein b und kommt damit nicht als neuer Abschnitt in Frage. Aber auch der Abschnitt $b a$ existiert bereits, sodass als neuer Abschnitt $b a a$ bestimmt wird. Insgesamt kann nach diesem Verfahren die gegebene Symbolreihe in folgende Abschnitte eingeteilt werden:

$$x = a | b | a b | b a | b a a | a b a | a a | b b | a b a b.$$

Auf diese Weise nehmen die jeweils nacheinander entsehenden Abschnitte jeweils Bezug auf vorher bereits definierte Abschnitte, sodass der dritte Abschnitt $a b$ sich aus dem ersten Abschnitt plus ein b bilden lässt. Da jeweils der kürzest mögliche

neue Abschnitt gebildet wird, kommt jeweils nur ein neues Zeichen hinzu. Damit genügt für die Darstellung jedes Abschnittes der Verweis auf einen vorherigen Abschnitt durch einen Index i und die Angabe des neu hinzugekommenen Symbols. Die für die gegebene Symbolreihe definierten Abschnitte lassen sich also wie folgt darstellen:

$x =$	a	b	a b	b a	b a a	a b a	a a	b b	a b a b
$x' =$	0a	0b	1b	2a	4a	3a	1a	2b	6b
i	1	2	3	4	5	6	7	8	9
<i>Anzahl Symbole in x</i>	1	2	4	6	9	12	14	16	20
<i>Anzahl Symbole in x'</i>	2	4	6	8	10	12	14	16	18

Tabelle 12: LZ78 Algorithmus zur Komprimierung von Symbolreihen

Die gegebene Symbolreihe x lässt sich nach dem LZ78 Algorithmus durch die Symbolreihe x' darstellen. x' umfasst dabei immer zwei Informationen. Die Zahl verweist auf einen Index i , bei dem sich bereits ein Teil der zu komprimierenden Symbolabfolge findet. Ist die Zahl 0 existiert kein solcher Abschnitt. Zum Abschnitt, auf den verwiesen wird, wird jeweils nur ein einziges neues Symbol hinzugefügt. Es zeigt sich, dass eine Komprimierung erst gegen Ende der Symbolreihe eintritt.

Da für die Darstellung jedes Abschnittes nur zwei Symbole benötigt werden, wird die Komprimierung mit jedem Abschnitt größer, kann jedoch für den Beginn der Zeitreihe sogar negativ ausfallen, d.h. dass anfangs unter Umständen mehr Symbole benötigt werden als in der Originalsymbolreihe. So werden für die Darstellung der ersten beiden Symbole $a b$ zunächst noch vier Symbole benötigt. Erst im letzten Abschnitt tritt eine Komprimierung ein, sodass nur 18 Symbole genügen um die 20 Symbole der Originalreihe zu generieren.

Die von Lempel und Ziv vorgeschlagenen Verfahren sind kontextabhängig

Es lässt sich zeigen, dass die Stärke der Komprimierung zunächst schnell ansteigt um dann auf einen konstanten Wert zu konvergieren. Der Komprimierungsfaktor für eine möglichst lange Symbolreihe strebt gegen ein Maß für die Komplexität der Symbolreihe. Dass die Komprimierung jedoch erst im Verlauf der Symbolreihe allmählich einsetzt, kennzeichnet dieses Verfahren als kontextabhängig, ein Umstand der für die oben bereits dargestellte Grammar Complexity nicht gilt. Allgemein ergibt sich für LZ78 die algorithmische Entropie h nach folgender Beziehung:

$$h = \lim_{N \rightarrow \infty} \frac{\log N}{\langle L(a) \rangle}$$

Der Term $\langle L(a) \rangle$ bezeichnet dabei die zu erwartende (d.h. in der Regel die durchschnittliche) Länge der benutzten Abschnitte und N die Länge der Symbolreihe.

Bei dem vorgestellten LZ78 Algorithmus handelt es sich um eine Vereinfachung des früheren LZ77 Algorithmus (Ziv & Lempel 1977), der weit aufwändiger in seiner Implementierung ist, jedoch bessere Komprimierungsraten erbringt.

Für die bereits gegebenen Symbolreihe würde der LZ77 Algorithmus folgendermaßen vorgehen:

$$x = |a b a b b a b a a a b a a a b b a b a b.$$

Der Algorithmus beginnt, indem ein Cursor an den Anfang der Symbolreihe gesetzt wird. In diesem Zustand ist das Wörterbuch des Algorithmus leer. Es findet sich damit kein Match der rechts des Cursors folgenden Symbole mit einem Eintrag im Wörterbuch. Damit ist auch die Länge eines solchen Matches Null. Direkt nach dem längsten Match (der hier ja Null ist) findet sich das Symbol a . Nun wird der Cursor hinter das neue Symbol a gesetzt:

$$x = a | b a b b a b a a a b a a a b b a b a b .$$

Alle Symbole, die sich links vom Cursor befinden, bilden den Inhalt des Wörterbuches. Es findet sich auch jetzt kein Match der rechts des Cursors folgenden Symbole mit einem Eintrag im Wörterbuch. Damit ist auch die Länge eines solchen Matches Null. Direkt nach dem längsten Match (der hier ja Null ist) findet sich das Symbol b . Nun wird der Cursor hinter das neue Symbol b gesetzt:

$$x = a b | a b b a b a a a b a a a b b a b a b$$

zurück um 2 Symbole ←
Übereinstimmung: 2 Symbole

Diesmal findet sich ein Match der folgenden zwei Symbole mit entsprechenden Einträgen im Wörterbuch. Um diesen Match im Wörterbuch zu finden genügt es zwei Schritte vom Cursor zurück zu gehen. Genau zwei Symbole im Wörterbuch nämlich $a b$ matchen. Direkt nach dem längsten Match (der hier zwei Symbole lang ist) findet sich das Symbol b . Nun wird der Cursor hinter das neue Symbol b gesetzt:

$$x = a b a b b | a b a a a b a a a b b a b a b$$

zurück um 5 Symbole ←
Übereinstimmung: 3 Symbole

Auch diesmal findet sich ein Match, nämlich der folgenden drei Symbole mit entsprechenden Einträgen im Wörterbuch. Um diesen Match im Wörterbuch zu finden müssen fünf Schritte vom Cursor zurück gegangen werden. Genau drei Symbole im Wörterbuch nämlich $a b a$ matchen. Direkt nach dem längsten Match (der hier drei Symbole lang ist) findet sich das Symbol a . Nun wird der Cursor hinter das neue Symbol a gesetzt:

$$x = a b a b b a b a a | a b a a a b b a b a b$$

zurück um 4 Symbole ←
Übereinstimmung: 6 Symbole

Ausgehend von dieser Cursorposition kann ein Match gefunden werden, der vier Symbole vor dem Cursor liegt. Zunächst scheinen auch genau vier Symbole zu matchen. Es zeigt sich aber, dass sich nach diesen vier Symbolen noch einmal der Beginn der Symbolfolge wiederholt: auf $a b a a$ folgt $a b$ was noch einmal auf die ersten beiden Symbole verweist. In diesem Sinne fordert der Algorithmus, dass ausgehend vom Cursor vier Schritte zurück gegangen wird und dann die dort vorgefundene Sequenz insgesamt sechs Zeichen lang genutzt wird. Da die Sequenz selbst nur vier Zeichen lang ist, wird für die verbleibenden zwei Symbole von vorne angefangen. Das nächste Zeichen nach der nun beschriebenen Symbolsequenz ist ein b . Nun wird der Cursor hinter das neue Symbol b gesetzt:

$x = a b a b b a b a a a b a a a b | b a b a b$

zurück um 11 Symbole ←
 Übereinstimmung: 4 Symbole }

Elf Schritte vor dem Cursor findet sich die Folge $b a b a$, die sich in vier Symbolen mit der rechts vom Cursor stehenden Abfolge deckt. Das nächste Zeichen nach der nun beschriebenen Symbolsequenz ist ein b . Damit ist die Komprimierung abgeschlossen.

Insgesamt wurde der Cursor nun sechs mal verschoben. Bei jeder Verschiebung wurden drei Größen ermittelt:

1. Die relative Verschiebung des Cursors nach links, bis an die Stelle im Wörterbuch, an der sich eine Sequenz von Symbolen findet, die ganz oder teilweise mit der Sequenz rechts des Cursors übereinstimmt.
2. Es wird angegeben, wie viele Symbole ab der angegebenen Position übereinstimmen. Es können dabei mehr Symbole angegeben werden, als die Sequenz bis zur Cursorposition lang ist. In einem solchen Fall wird die Sequenz noch einmal wiederholt.
3. Es wird das Zeichen angegeben, welches direkt auf die bereits beschriebene Sequenz folgt.

Auch beim hier beschriebenen LZ77 Algorithmus handelt es sich um einen kontextgebundenen Algorithmus, für den die Komprimierung erst ab einer bestimmten Länge der Symbolreihe eintritt. Eine Diskussion verschiedener Methoden algorithmischer Entropie und die mit ihnen verbundene Möglichkeit, Zufallsprozesse von geordneter Komplexität zu unterscheiden, findet sich z.B. bei Ebeling, Steuer und Titchener (2001).

Literatur

- Bortz J, Lienert G, A. & Boehnke K (2000) *Verteilungsfreie Methoden in der Biostatistik*. Springer, Berlin
- Chaitin GJ (1974) Information Theoretic Computational Complexity. *IEEE Transactions on Information Theory*, IT20, S. 10-15
- Collet P & Eckmann JP (1980) *Iterated Maps on the Interval as Dynamical System*. Birkhäuser, Basel
- Ebeling W & Jiménez-Montano MA (1980) On Grammars, Complexity, and Information Measures of Biological Macromolecules. *Mathematical Biosciences*, 52, S. 53-71
- Ebeling W, Steuer R & Titchener MR (2001) Partition-Based Entropies of Deterministic and Stochastic Maps. *Stochastic and Dynamics*, 1 (1), S. 1-17
- Friedlmayer S, Reznicek E & Strunk G (1996) *Sozialisationschancen und Betreuungsstrukturen*. Amt für Jugend und Familie der Stadt Wien, Wien
- Hadamard J (1898) Les surfaces à courbures opposées et lignes géodésiques. *J. Math. pures appl.*, S. 27-73
- Hubermann BA & Hogg T (1986) Complexity and Adaptation. *Physica D*, 22, S. 376-384
- Jiménez-Montano MA (1984) On the Syntactic Structure of Protein Sequences and the Concept of Grammar Complexity. *Bulletin of Mathematical Biology*, 46, S. 641-659
- Kolmogorov AM (1965) Three Approaches to the Definition of the Concept Quantity of Information. *IEEE Transactions on Information Theory*, IT14, S. 662-669
- Lempel A & Ziv J (1978) Compression of Individual Sequences Via Variable-Rate Coding. *IEEE Transactions on Information Theory*, IT 24, S. 530-536
- Mood AM (1940) Distribution Theory of Runs. *The Annals of Mathematical Statistics*, 11, S. 367-392
- Morse M (1921) Recurrent Geodesics on a Surface of Negative Curvature. *Transactions of the American Mathematical Society*, 22, S. 84-110
- Morse M & Hedlund GA (1938) Symbol Dynamics. *American Journal of Mathematics*, 60, S. 815-866
- Rapp PE, Jiménez-Montano MA, Langs RJ, Thomson L & Mees AI (1991) Toward a Quantitative Characterization of Patient-Therapist Communication. *Mathematical Biosciences*, 105, S. 207-227
- Schürmann T & Grassberger P (1996) Entropy Estimation of Symbol Sequences. *Chaos*, 6 (3), S. 414-427
- Shannon CE (1948) A Mathematical Theory of Communication. *Bell System Technical Journal*, 27, S. 379-423 and 623-656
- Stevens SS (1939) On the Problem of Scales for Measurement of Psychological Magnitudes. *Journal of Unified Science*, 9, S. 94-99
- Strunk G (1996) *Die Sequentielle Plananalyse als systemwissenschaftliche Methode der Psychotherapieprozeßforschung*. Unveröffentlichte Diplomarbeit, Westfälische Wilhelms Universität
- Strunk G & Schiepek G (2002) Dynamische Komplexität in der Therapeut-Klient-Interaktion. Therapieforchung aus dem Geiste der Musik. *Psychotherapeut*, 47 (5), S. 291-300
- Thiele C (1997) Selbstorganisation und Komplexität in der Psychotherapie. *Systeme. Interdisziplinäre Zeitschrift für systemtheoretisch orientierte Forschung und Praxis in den Humanwissenschaften*, 11 (2), S. 21-32

- Tschacher W & Scheier C (1995) Analyse komplexer psychologischer Systeme. II. Verlaufsmodelle und Komplexität einer Paartherapie. *System Familie*, 8, S. 160-171
- Ziv J & Lempel A (1977) A Universal Algorithm for Data Compression. *IEEE Transactions on Information Theory*, IT 23 (3), S. 337-343
- Zvonkin AK & Levin LA (1970) The Complexity of Finite Objects and the Development of the Concepts of Information and Randomness by Means of the Theory of Algorithms. *Russian Mathematics Surveys*, 25 (6), S. 83-124