Kapitel 1
Grundbegriffe

Denn eben wo Begriffe fehlen,
Da stellt ein Wort zur rechten Zeit sich ein.
Johann Wolfgang von Goethe, Faust I, Szene IV

In diesem Kapitel klären wir Grundbegriffe, mit denen die Informatik umgeht. Wir beginnen mit Begriffen wie Nachricht und Information, die wir nicht definieren, sondern nur erläutern können. Viele Grundbegriffe stammen aus der Systemtechnik; wir führen sie hier ohne weiteren Bezug ein. Am Ende sehen wir, daß Algorithmen nicht nur Rechenvorschriften im landläufigen Sinne sind.

1.1 Signal, Datum, Information

Mit einem Satz wie „Ich habe meine Meinung zum Thema *X* gesagt" teilen wir in der Umgangssprache mit, daß wir unsere subjektive Wertung eines Sachverhalts *X* einem anderen, dem Empfänger unserer Mitteilung, weitergegeben haben. Wir haben Schallwellen zur Weitergabe benutzt, wie das Wort *sagen* zeigt. Bei einem Telefongespräch wären diese Schallwellen zwischenzeitlich in elektromagnetische Impulse oder Lichtwellen verwandelt worden, bevor sie beim Empfänger wieder als Schallwellen aus dem Hörer kamen. Wir hätten unsere Mitteilung auch schriftlich, als Text, auch unter Einschluß von Bildern, formulieren können. Dann wären beim Empfänger Lichtwellen angekommen, da das Auge Geschriebenes durch die Modulation der beleuchtenden Lichtwellen erfaßt.

Dem Empfänger sind also nur Schall- oder Lichtwellen, d. h. physikalische Größen, zugänglich geworden. Die Darstellung einer Mitteilung durch die zeitliche Veränderung einer physikalischen Größe heißt ein **Signal**[1]. Die Eigenschaften des Signals, die sich dabei ändern, z. B. die Frequenz oder Amplitude einer Welle, heißen **Signalparameter**. Auf dieser Ebene heißt die Weitergabe von Mitteilungen **Signalübertragung**.

Die dauerhafte Darstellung einer Mitteilung auf einem physikalischen Medium heißt eine **schriftliche Darstellung** oder **Inschrift**. Das physikalische Medium heißt **Schriftträger**. Das Wort *Schrift* wird hier allgemeiner verwandt als

1. lat. *signum*, Zeichen.

in der Umgangssprache: Auch Aufzeichnungen auf einem Magnetband oder die Aufzeichung eines Musikstücks auf einer Schallplatte sind Inschriften.

Signale und Inschriften und deren Verarbeitung sind die technischen Gegenstände[2], auf denen die Informatik aufbaut. Die Beispiele zeigen, daß es nur auf einige Eigenschaften dieser Gegenstände ankommt, die für uns die Bedeutung der Gegenstände ausmacht. Wir nennen dies die **Abstraktion** der Gegenstände und sagen, „wir haben uns einen Begriff von dem Gegenstand gemacht".

Wenn wir bei der Darstellung und Weitergabe einer Mitteilung vom verwandten Medium und den Einzelheiten der Signale und Signalparameter abstrahieren, heißt die Mitteilung eine **Nachricht**. Nachrichten sind die Grundgegenstände, mit denen wir uns in der Informatik befassen. Signale und Inschriften bilden das konkrete Medium zur Wiedergabe von Nachrichten.

Im eingangs zitierten Satz besteht die Nachricht aus 41 Zeichen, wenn wir die Zwischenräume mitzählen. Daß es sich um 8 Wörter handelt (von denen eines nur der Buchstabe *X* ist), ist bereits eine weitergehende Interpretation der Zeichenfolge: Wir haben die Konvention benutzt, daß Zwischenräume Wörter trennen und konnten daher deren Anzahl feststellen. Würden wir den Satz wie im Mittelalter ohne Zwischenräume schreiben:

Ichhabeine
ineMeinu
ngzumThe
maXgesagt

so müßten wir seinen Sinn verstehen, um ihn korrekt in Wörter zu gliedern[3].

Eine Nachricht kann verschiedene Bedeutungen haben. Im Falle unseres Satzes z. B. die Bedeutungen *Text* (also kein Bild), *8 Wörter*, *deutschsprachiger Text*, *Satz*, *Aufforderung zum Nachdenken*. Die Bedeutung ergibt sich teils aus allgemeinen Konventionen, etwa der Erkenntnis, daß Schriftzeichen vorliegen und der Text in Wörter gegliedert ist. Um zu sehen, daß ein vollständiger Satz der deutschen Sprache vorliegt, bedarf es zusätzlicher Fähigkeiten. Daß eine Aufforderung zum Nachdenken vorliegen könnte, ist eine subjektive Wertung, die sich vielleicht aus dem Zusammenhang eines Gespräches ergibt, aber selbst dann nicht zwingend ist. Die Kenntnisse, die man benötigt, um einer Nachricht

2. Wir verwenden in diesen Vorlesungen das Wort *Gegenstand* in der allgemeinen Bedeutung *Ding*, *Person*, *Thema*, *Sachverhalt* (Beziehung zwischen Dingen, Personen, Themen oder anderen Sachverhalten). Dinge und Sachverhalte können dabei sowohl Gegenstände der realen Welt als auch einer gedachten Modellwelt sein.

3. In der japanischen Schrift tritt dieses Problem heute noch auf. Da Kanji-Zeichen jeweils ganze Wörter bedeuten, kann man sie ohne Zwischenraum schreiben. Mit Hiragana-Zeichen notierte Vor- und Nachsilben gehen aber ineinander über: Der Leser muß wissen, wo ein Wort aufhört und das nächste anfängt.

Bedeutung zuzuordnen, nennen wir einen **Kontext** oder ein **Bezugssystem**. Unterschiedliche Bezugssysteme können der gleichen Nachricht unterschiedliche Bedeutungen zuordnen.

Die zugeordnete Bedeutung heißt eine **Information**. Man gewinnt sie durch **Interpretation** von Nachrichten auf der Grundlage eines Bezugssystems. Solche Interpretationen können wir zum Beispiel als Paare (Nachricht, Bedeutung) in ein Wörterbuch eintragen und wie Vokabeln auswendig lernen. Dies ist jedoch nur ein Spezialfall der Angabe einer **Interpretationsvorschrift**[4], deren Anwendung auf verschiedene Nachrichten die jeweils zugeordnete Information liefert.

Eine Nachricht ohne zugeordnete Information ist sinnlos. Umgekehrt sind Informationen immer durch Nachrichten repräsentiert. Wenn wir Nachrichten speichern, übertragen oder verarbeiten, um hierdurch Informationen zu speichern, zu übertragen oder zu verarbeiten, so betrachten wir eigentlich das Paar ⟨Nachricht, zugeordnete Information⟩. Man nennt dieses Paar ein **Datum**[5]. Ein Datum ist also eine bedeutungstragende Nachricht. Eine Verarbeitung von Nachrichten ist eine **Datenverarbeitung**, wenn die Verarbeitung **bedeutungstreu** ist, d. h., wenn ihr Ergebnis auf der Ebene der Informationen Bedeutung besitzt.

Das Wort *Datenverarbeitung* besitzt auch in der Umgangssprache die Bedeutung *Verarbeitung von Daten mit technischen Hilfsmitteln*. Hingegen ist die Bedeutung der Wörter *Nachricht*, *Information* und *Informationsverarbeitung* fließend. Auch in Fachsprachen werden die Wörter *Nachricht*, *Datum* und *Information* oft mit überlappender Bedeutung gebraucht. Das Wort *Information* hat darüberhinaus in der Shannonschen Informationstheorie noch eine andere präzise Bedeutung, auf die wir in B.3.1 eingehen. Am schillerndsten ist der Begriff *Informationstechnik*, der auch die technische Mikroelektronik, die Signalverarbeitung und die Datenübertragung umfaßt.

Interpretationsvorschriften sind selbst Informationen, die durch Nachrichten repräsentiert werden. So wird °C im Satz „die Temperatur beträgt 20°C“ als Vorschrift verstanden, die vorangehende Zahl als Gradzahl in der Celsius-Skala zu interpretieren. Wir unterscheiden hier zwischen

- Datum,
- Information,
- Umdeutung der Information als Interpretationsvorschrift,
- Ergebnis der Anwendung der Interpretationsvorschrift.

Dies wird bei mathematischen Formeln deutlich: x^2 kann verstanden werden als

- der Buchstabe x mit oberem Index 2;
- ein Polynom mit der Unbekannten x;
- die Vorschrift „quadriere x“;
- das Quadrat von x.

4. Oft bezeichnet das Wort *Interpretation* sowohl den Vorgang des Interpretierens, als auch dessen Ergebnis, als auch die Interpretationsvorschrift.

5. lat. *datum*, das Gegebene.

Die Aufzählung zeigt, daß die gleiche Nachricht auf verschiedenen Abstraktionsstufen interpretiert werden kann. Das jeweilige Bezugssystem bestimmt, welche Information gemeint ist. Im Beispiel folgen diese Interpretationen festen Regeln. Daher sind die Interpretationen automatisch ausführbar. Auf einer höheren Abstraktionsstufe gibt es jedoch im allgemeinen eine Interpretation, die sich nicht mehr automatisch erreichen läßt. Im Beispiel könnte das die Antwort auf die Frage sein, warum wir x quadrieren wollen.

Im Gegensatz zu Nachrichten, die in *verschiedenen* Bezugssystemen unterschiedliche Bedeutungen haben, besitzen **mehrdeutige Nachrichten** im *gleichen* Bezugssystem mehrere verschiedene Interpretationen. So kann unter der Position einer Person ihre Rangstufe in einer betrieblichen Hierarchie verstanden werden. Es kann aber auch der Ort gemeint sein, an dem sich die Person aufhält.

Von mehrdeutigen Nachrichten unterscheidet man **widersprüchliche** oder **inkonsistente Nachrichten**, denen sich in einem bestimmten Bezugssystem kein Sinn zuordnen läßt. Aus dem Altertum ist das Beispiel „Alle Kreter sind Lügner, sagte ein Kreter“ bekannt. Der widersprüchliche Begriff der *Menge aller Mengen*, die sich selbst als Element enthalten müßte, ist ein Beispiel, daß es auch in der Mathematik schwierig sein kann, Widersprüche zu vermeiden.

1.1.1 Wissen

Informationen, wie man Daten interpretiert, bezeichnen wir zusammenfassend als **Wissen**. Dazu zählt sowohl die unmittelbare Kenntnis der durch ein Datum gegebenen Information als auch die Kenntnis von Interpretationsvorschriften zur Erzeugung solcher Informationen. Wir sprechen von **Faktenwissen** bzw. **synthetischem** oder **prozeduralem Wissen**, um diese beiden Fälle zu unterscheiden. Die Daten zur Darstellung des Wissens heißen auch eine **Wissensrepräsentation** oder **Datenstruktur**.

Wissen ist **statisch** oder **dynamisch**. Statisches Wissen beschreibt zeitlich unveränderliche Gegenstände und Sachverhalte. Dynamisches Wissen beschreibt zeitlich veränderliche Gegenstände und Sachverhalte zu bestimmten Zeitpunkten. Wir können statischem Wissen die Zeit hinzufügen, zu der es gültig ist (oder war oder sein wird). Diskretes dynamisches Wissen können wir als eine Folge von statischen Kenntnissen auffassen: jede einzelne statische Aussage beschreibt einen Zustand, der in der zeitlichen Abfolge vorkommt (oder zumindest vorkommen könnte). Umgekehrt ist das Gesamtwissen über einen Ablauf statisches Wissen: ein Steinwurf gehorcht ballistischen Gesetzen, aus denen wir für jeden Zeitpunkt das dynamische Wissen ableiten können, wo sich der Stein gerade befindet.

Die Einzelheiten eines Sachverhalts oder Ablaufs sind nicht alle interessant. Den Grad der Auflösung von Einzelheiten, die man noch berücksichtigen will, bezeichnet man als die **Granularität** oder **Körnigkeit** der Untersuchung. Um

aus der Beobachtung eines Blitzes den Schluß zu ziehen, daß ein Gewitter naht, benötigen wir keine „feinkörnigen“ Kenntnisse über den physikalischen Ablauf des Blitzes. Für die Schlußfolgerung genügt Wissen einer wesentlich gröberen Körnigkeit. Was jeweils als fein- oder grobkörniges Wissen anzusehen ist, hängt vom Bezugssystem und den beabsichtigten Schlußfolgerungen ab.

Gewöhnlich müssen wir die Körnigkeit, mit der wir einen Gegenstand oder Ablauf in Raum und Zeit erfassen wollen, bewußt und explizit festlegen, um nicht in uninteressanten Einzelheiten zu versinken. Dabei stellt sich jedes Mal die Frage, ob man auch wirklich alle relevanten Einzelheiten erfaßt hat. So würde man bei der Erfassung der Bauteile eines Flugzeugs zunächst Rumpf, Flügel, Triebwerke usw. nennen, aber nicht die Schrauben, mit denen diese Teile verbunden sind. Wenn das Flugzeug ausgerechnet wegen Materialermüdung dieser Schrauben abstürzt, zeigt sich, daß wir das Flugzeug zu grobkörnig erfaßt haben. Wenn wir einem Vortrag zuhören, bemühen wir uns, die Hintergrundgeräusche zu überhören. Hier liegt der umgekehrte Fall vor. Unser Ohr nimmt das akustische Signal feinkörnig auf; die Geräusche müssen weggefiltert werden.

Die Körnigkeit kann nicht beliebig fein gewählt werden. Die Länge eines Holzstücks können wir mit dem Metermaß selbst bei Einsatz einer Lupe nicht viel genauer als auf einen Zehntel Millimeter angeben. Wir wissen zwar, daß das Holzstück eine auf beliebig viele Dezimalstellen genau angebbare Länge hat, aber wir können sie nicht *exakt* ermitteln: Jeder Versuch, etwa schon das Einspannen in eine Schublehre mit Nonius, würde die Länge mit Hilfe der Meßvorrichtung verändern. Wegen dieser gegenseitigen Einflüsse zwischen Meßprozeß und zu Messendem ist sogar unser feinkörniges Wissen über physikalische Vorgänge notwendig immer **ungenaues Wissen**. Wenn die Ungenauigkeit quantifiziert wird, etwa durch Angabe eines Intervalls, in dem der Meßwert liegt, sprechen wir von **unscharfem Wissen**[6]. Der Mensch begnügt sich oft mit unscharfem Wissen, wenn von vornherein klar ist, daß größere Genauigkeit keine zusätzliche relevante Information liefert. Man benutzt dann oft Beiwörter wie *ungefähr, ziemlich, sehr* usw..

Wieviele (Sand-)Körner machen einen (Sand-)Haufen?[7] Zerstört dann die Wegnahme eines Sandkorns die Eigenschaft *Sandhaufen*? Der Versuch, diese Fragen zu beantworten, zeigt, daß unscharfes Wissen mit der Angabe eines *Möglichkeitsgrades* verbunden ist, mit dem wir den Wahrheitsgehalt einer Aussage bewerten. Ob 10 000 Sandkörner einen Sandhaufen bilden, bewertet die Hausfrau, die ein Zimmer kehrt, anders als ein Kind im Sandkasten. Es gibt kein nachvollziehbares Experiment, mit dem wir statistisch ermitteln könnten, ob 10 000 Sandkörner einen Sandhaufen bilden: Möglichkeitsgrade sind keine Wahrscheinlichkeiten im Sinne der Wahrscheinlichkeitstheorie.

6. engl. *fuzzy knowledge*, also eigentlich *fusseliges Wissen*.

7. Die Frage wird Eubulides von Milet, ca. 400 v. Chr., zugeschrieben und ist in der Logik als „Sorites“, der Häufler, bekannt.

Prozedurales Wissen über Abläufe setzt der Mensch zur komprimierten Speicherung statischen Wissens ein: Statt die Ergebnisse der Multiplikation beliebiger Zahlen als Einmaleins-Tabellen auswendig zu lernen, genügt das Wissen, wie man ein Multiplikationsergebnis herleitet.

Die Mathematik dient der Modellbildung statischen Wissens. Sie definiert Begriffe und feststehende Beziehungen zwischen diesen Begriffen. Zeitliche Abläufe beschreibt sie beispielsweise durch Differentialgleichungen mit der Zeit t als unabhängiger Variable; damit erfaßt sie den Gesamtablauf. Dabei werden „unwichtige" Einzelheiten oft nicht erfaßt. Wenn das Prinzip *kleine Ursache, kleine Wirkung* nicht gilt, kann man das mathematische Modell allerdings nur noch benutzen, um einige wenige Schritte zu extrapolieren, bevor der Ablauf von nicht erfaßten Einzelheiten und Ursachen abhängig wird.

Natürlich kann man auch dynamisches Wissen mathematisch repräsentieren, z. B. indem man Indizes als Zeitpunkte $t_0, t_1, \ldots$ auffaßt, zu denen Werte $x_0, x_1, \ldots$ vorliegen. Jedoch ist *Zeit* kein mathematischer Begriff. Wenn wir t_i als Zeitpunkt auffassen, interpretieren wir die Mathematik in einem nichtmathematischen Bezugssystem.

Neben der Frage nach geeigneten Wissensrepräsentationen befaßt sich die Informatik vor allem mit dynamischem Wissen über zeitliche Abläufe. Sie kennt wie die Natur- und Ingenieurwissenschaften den Begriff des **Zustands** eines Gegenstands, d. h. einer Menge zeitlich veränderlicher Attribute, die einem Objekt gleichbleibender Identität zugeordnet sind. Bei der Interaktion gleichzeitig ablaufender Vorgänge muß sie sich beispielsweise damit auseinandersetzen, wie man dem von der Eisenbahn bekannten Problem ausweicht, daß der Anschlußzug wegen Verspätung des eigenen Zuges bereits abgefahren ist. Die Informatik betrachtet solche Probleme *zeitlich lokal* während des Ablaufs. Dabei kann sie den tatsächlich erreichten Zustand mit dem vorhergesagten Zustand vergleichen und Abweichungen, die aufgrund statischen Wissens nur unscharf, nur mit einer gewissen Wahrscheinlichkeit oder gar nicht vorhergesagt werden konnten, in die weitere Arbeit einbeziehen.

1.1.2 Analoge und digitale Signale

Physikalische Prozesse verlaufen in der Zeit: Zu jedem Zeitpunkt lassen sich die für den Prozeß charakteristischen Meßgrößen feststellen. Die Werte der Signalparameter ändern sich kontinuierlich, und wir können kontinuierlich die aus diesen Signalparametern abzuleitenden Informationen bestimmen. Wegen der Analogie zwischen kontinuierlichem physikalischen Prozeß und dieser Art der Signalverarbeitung spricht man von **analogen Signalen** und **analoger Signalverarbeitung**. Wenn wir stattdessen die Werte der Signalparameter und die daraus

abgeleiteten Daten und Informationen nur zu diskreten Zeitpunkten bestimmen, sprechen wir von **diskreten** oder **digitalen Signalen**, **Daten** und **Informationen**.

Die Informatik beschäftigt sich in der Hauptsache mit der Verarbeitung digitaler Daten und Informationen. Folgende Überlegungen zeigen, daß dies keine wesentliche Einschränkung ist:

- Die Quantenmechanik lehrt uns auf der Ebene feinster Körnigkeit, daß physikalische Prozesse als Folgen von Quantensprüngen, also als diskrete Prozesse, aufgefaßt werden können. Sie lehrt uns überdies, daß Meßwerte nicht beliebig genau sein können, weil jede Messung einen Eingriff in den gemessenen Prozeß darstellt und die Meßwerte verändert (HEISENBERGsche Ungenauigkeitsrelation).
- Nach dem Abtasttheorem können wir jede periodische Funktion der Form $f(t) = \frac{a_0}{2} + \sum_{k=1}^{\infty} (a_k \cos k\omega_0 t + b_k \sin k\omega_0 t)$ mit einer Grenzfrequenz $\omega_g = n\omega_0$ aus diskreten Meßwerten rekonstruieren, wenn die Abtastfrequenz, die Anzahl der Meßwerte pro Zeiteinheit, mindestens $2\omega_g$ beträgt. Wegen der beschränkten Genauigkeit physikalischer Meßapparaturen werden sehr hohe Frequenzen nicht korrekt wiedergegeben: die Messung schneidet hohe Frequenzen, so sie existieren, ab. Die Voraussetzung des Abtasttheorems, nämlich die Existenz einer Grenzfrequenz, ist praktisch immer gegeben. Daß die digitale Aufzeichnung eines Musikstücks gegenüber einer analogen Aufzeichnung die Wiedergabetreue nicht herabsetzt, erläutert diese Überlegung.
- Zwar können wir analoge Eingangssignale mit entsprechenden physikalischen Apparaturen kontinuierlich in Ausgangssignale umformen. Soll diese Umformung allerdings als Informationsverarbeitung gewertet werden, so muß die Interpretation der Signale als Daten bzw. Informationen bedeutungstreu sein. Dies kann nur durch Experimente nachgewiesen werden, bei denen wir jeweils endlich viele Meßwerte bestimmen und prüfen, daß diese richtig interpretiert wurden. Wenn wir aus solchen Experimenten auf einen kontinuierlichen Zusammenhang zwischen physikalischen Signalen und Informationen schließen, so handelt es sich um eine Idealisierung. Diese läßt sich bei geeigneter mathematischer Modellierung zwar leichter handhaben; die Gültigkeit der Idealisierung haben wir aber in Wahrheit nur durch Extrapolation aus endlich vielen Messungen endlicher Genauigkeit begründet.

1.1.3 Codierung von Daten

Gehen wir nicht von Signalen und Nachrichten, sondern von Informationen aus, so stellt sich die Frage, wie diese durch Nachrichten dargestellt werden können. Eine hierfür geeignete Nachricht heißt eine **Codierung** der Information: die Zahl

zwei (eine Information) kann durch das Wort zwei, die Ziffer 2, aber auch durch zwei Striche || codiert werden. Um die Frage überhaupt erörtern zu können, muß die Information bereits als Nachricht vorliegen. Wir sprechen daher allgemeiner von einer Codierung von Daten und sagen umgekehrt, die Information oder das Datum liege *in codierter Form* vor.

Unterschiedliche Informationen müssen wir durch unterscheidbare Nachrichten codieren. Sie heißen **Zeichen**[8] und die verfügbare Menge von Zeichen ein **Zeichenvorrat** oder **Code**[9]. Daten werden durch einzelne Zeichen oder eine Folge von Zeichen, eine **Zeichenreihe**, dargestellt. Statt Zeichenreihe sagen wir oft **Wort** oder **Text**. Das Paar (Zeichen, Bedeutung) heißt ein **Symbol**[10].

Zeichen werden durch Signale oder Inschriften wiedergegeben. In endlicher Zeit können wir nur endlich viele verschiedene Zeichen darstellen oder erkennen. Die Zeichenvorräte der Informatik sind daher endlich und beschränkt. In der theoretischen Informatik ebenso wie in der Mathematik gehen wir gelegentlich vom Denkmodell unbeschränkter Zeichenvorräte aus; dies bedeutet meist nur, daß uns bei Bedarf immer noch ein zusätzliches, bisher nicht verwandtes Zeichen zur Codierung einer neuen Information zur Verfügung stehen soll.

Zeichen erscheinen immer in einem Kontext, einem Bild. Dabei sind die einzelnen Zeichen oft nicht exakt voneinander trennbar. Auch kann die Interpretation einzelner Zeichen von ihrer Position im Text oder Bild abhängen. Handelt es sich um Meßwerte, so können auch die Geschwindigkeit oder die Zeitpunkte des Einlaufens der Werte wichtig sein. Auch das Ausbleiben eines Meßwerts kann eine Codierung für ein Datum sein. Bei einer Verkehrsampel würde beispielsweise das Ausbleiben des Umschaltens über einen längeren Zeitraum die Mitteilung ‚Ampel defekt' codieren. Ist dagegen nur die Glühbirne des grünen Lichts ausgefallen, so könnte man die Grünphase immer noch am Abschalten des roten Lichts erkennen. Dies ist ein Beispiel einer **redundanten Codierung**[11]: Dabei kann man den Meßwerten das Datum auf verschiedene Weise entnehmen; fallen Werte aus oder sind sie grob verfälscht, so kann man die Inkonsistenz bemerken oder die Information sogar trotzdem noch erhalten. Redundanz der Codierung ist nicht nur in diesem Beispiel eine wichtige Hilfe, um Datenfehler zu erkennen oder sogar zu korrigieren, um die Sicherheit eines datenverarbeitenden Vorgangs zu steigern.

8. Das englische Wort *token* wurde Anfang des 20. Jahrhunderts in der Philosophie und Psychologie als Übersetzung des Wortes Zeichen in der hier benutzten Bedeutung eingeführt.

9. Die Bedeutung der Wörter Zeichen, Zeichenvorrat und Code ebenso wie die weiter unten gebrauchten Begriffe Bit, bits, Binärzeichen usw. sind in deutschen und internationalen Normen festgehalten, um Mißverständnisse zu vermeiden. Für die vorstehenden Begriffe finden sich die Definitionen in der Norm DIN 44 300.

10. griech. σύμβολον, Zeichen.

11. lat. *redundare*, im Überfluß vorhanden sein.

Die Verarbeitung digitaler Daten abstrahiert zumeist von der physikalischen Repräsentation der Zeichen und verwendet die abstrakte zweiwertige Grundmenge $\mathbb{B} = \{\mathtt{O}, \mathtt{L}\}$ als Code. Man nennt ihn einen **Binärcode**. Die beiden Werte heißen **Binärzeichen**. Die Frage, wie diese Zeichen einem Signal entnommen werden, ist kontextabhängig, wie wir an Beispielen sahen. Eine Größe, die einen dieser beiden Werte annehmen kann, heißt ein **Bit**. Die Länge einer binär codierten Zeichenreihe wird in **Bits** gemessen. Bits werden wie eine physikalische Dimension verwendet. Zum Beispiel geben wir die Anzahl der pro Sekunde übertragenen Binärzeichen in [Bits/sec] an. Mit Binärcodes kann man beliebige Zahlwerte codieren. Im Anhang B beschreiben wir, wie dies unter verschiedenen Randbedingungen geschehen könnte.

Der **Entscheidungsgehalt** eines Zeichenvorrats mit n Zeichen ist die kleinste Anzahl H von Entscheidungen, mit denen man feststellen kann, welches der n Zeichen vorliegt. Man mißt H in **bits**. Es gilt $H = 1$ für $n = 2$ und $H \leqslant \operatorname{ld} n$ für $n > 2$, vgl. B.3.1[12].

Das Zehnersystem oder das Sexagesimalsystem der Babylonier, das wir in der Einteilung von Stunden in Minuten und Sekunden benutzen, sind andere Beispiele für Zeichenvorräte. Sie sind geordnet und heißen **Alphabete**.

Das lateinische, arabische, griechische, hebräische oder kyrillische Alphabet ist ein Alphabet in diesem Sinne. Mit Hilfe der Ordnung eines Alphabets können wir Zeichenreihen aufsteigend oder absteigend lexikographisch anordnen, wie dies beispielsweise im Telefonbuch oder in einem Wörterbuch geschieht. Im Deutschen ist dabei die Anordnung nach DIN 5 007 geregelt: Die Buchstaben sind nach dem lateinischen Alphabet geordnet; Groß- und Kleinschreibung werden nicht unterschieden. Ziffern folgen nach den Buchstaben. ß wird wie ss behandelt. Die Umlaute ä, ö, ü zählen wie a, o bzw. u. *Küche* kommt also vor *Kuchen*. Nur in Namensverzeichnissen, z. B. dem Telefonbuch, werden ä, ö und ü wie Ligaturen behandelt und als ae, oe bzw. ue eingeordnet.

Wir benutzen für die Textverarbeitung Zeichenvorräte wie ASCII[13], eine Ausprägung des ISO-7-Bit-Codes[14], oder seine Obermenge ISO-Latin-1, festgelegt in der Norm ISO 8859-1, der 256 Zeichen umfaßt, darunter die Sonderzeichen der westeuropäischen Sprachen, vgl. die Codetabelle B.1, S. 363. Die Normen legen eine Binärcodierung der Zeichen durch 7 bzw. 8 Bits fest. Die resultierende Anordnung der Zeichen macht die Zeichenvorräte zu Alphabeten.

Für die gesprochene Sprache sind nicht Buchstaben, sondern Silben die elementaren Zeichen, die wir weiter interpretieren. Diese setzen sich aus **Phonemen** zusammen, die den Buchstaben entsprechen. Das erwähnte Hiragana der Japaner

12. ld bedeutet *logarithmus dualis* und bezeichnet den Logarithmus zur Basis 2.
13. ASCII: *American Standard Code for Information Interchange*.
14. ISO: *International Standard Organization*.

ist eine geschriebene Silbenschrift mit 48 Zeichen.

Der umfangreichste Code, der in der Natur vorkommt, ist der genetische Code der Erbanlagen in den DNS- (bzw. RNS-) Ketten der Chromosomen von Lebewesen. Dieser chemisch repräsentierte Code $\{A, T, G, C\}$ hat die vier Basen Adenin, Thymin, Guanin und Cytosin als Grundelemente. Im allgemeinen bestimmt ein Triplett, z. B. GAT, eine Aminosäure in einem Eiweißmolekül. In einigen Fällen sind jedoch mehrere Tripletts der gleichen Aminosäure zugeordnet.

Die Bestimmung der Zeichen aus gegebenen Signalen ist nicht immer so einfach wie bei der Verkehrsampel: Bei handschriftlich geschriebener Sprache kann die gleiche handschriftliche Figur nicht nur bei verschiedenen Schreibern verschiedene Zeichen darstellen; sogar beim gleichen Schreiber kann das Zeichen vom Kontext abhängen. Solche Beispiele sind Spezialfälle des Problems der **Mustererkennung**. Sie befaßt sich damit, wie man einem Bild, einem Text oder einem physikalischen Prozeß, der diskrete oder kontinuierliche Signale liefert, Informationen entnehmen kann. Andere Beispiele sind die automatische Erkennung der Postleitzahlen beim Sortieren von Briefen oder die Überwachung eines Roboters durch Analyse der Bilder einer Fernsehkamera.

Die bisherigen Beispiele für Codes beziehen sich auf die unterste Ebene der Erkennung von Zeichen aus Signalen. Dieselben Überlegungen kann man auch auf höheren Abstraktionsstufen anstellen. Etwa beim Übergang

$$\text{handschriftliche Figur} \rightarrow \text{Buchstabe} \rightarrow \text{Wort} \rightarrow \text{Satz}$$

ergeben ein oder mehrere Zeichen der einen Stufe zusammen jeweils ein Zeichen der nächsten Stufe. Umgekehrt wird jeweils ein Zeichen der einen Stufe durch eine Folge von Zeichen der darunterliegenden Stufe codiert. Passende Codierung ist oft entscheidend für das rasche Wiederauffinden und Wiedererkennen der codierten Daten und Informationen und kann daher über die Komplexität und Effizienz eines informationsverarbeitenden Vorgangs entscheiden.

1.1.4 Von der Signal- zur Informationsverarbeitung

Bisher haben wir uns mit der Interpretation eines einzelnen Signals oder einer Folge von Signalen befaßt. Die Informationsverarbeitung beschäftigt sich aber vor allem mit der Verknüpfung von Daten unterschiedlicher Herkunft zur Gewinnung neuer Informationen. Im wesentlichen können wir zwei verschiedene Formen der Verknüpfung unterscheiden.

Wenn wir die vierte Primzahl bestimmen, oder 3 + 4 berechnen und als Ergebnis 7 erhalten, sprechen wir von einer **selektierenden** oder **transformierenden Informationsverarbeitung**. Die ursprünglichen Daten bzw. die damit verbundenen Informationen sind im Ergebnis nicht mehr enthalten. Das Ergebnis stellt

eine neue Information dar[15]. Die ursprünglichen Informationen lassen sich aus dem Ergebnis nicht oder nur teilweise zurückgewinnen.

Davon unterscheiden wir die **strukturierende** oder **relationale Informationsverarbeitung**, die Beziehungen zwischen vorhandenen Informationen herstellt und dabei die ursprünglichen Informationen als Bestandteil des Ergebnisses beibehält. Ein Beispiel ist das Zusammenfügen von Worten zu einem Satz: Der Satz enthält neue Information, die aber unter Verwendung der durch die Worte gegebenen Informationen ausgedrückt wird. Strukturierende oder selektierende Informationsverarbeitung kann im Einzelfall auch durch Uminterpretation vorhandener Daten erreicht werden. Transformierende Informationsverarbeitung wie im obigen Beispiel erzeugt jedoch neue Daten und damit auch neue Signale oder Inschriften. Auch bei strukturierender Informationsverarbeitung ist dies der Normalfall.

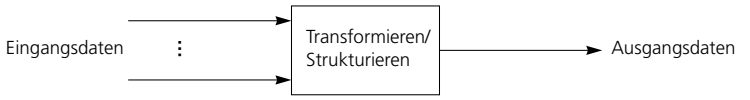

Abbildung 1.1: Erzeugung neuer Daten

Technisch müssen hierzu Eingangsdaten bzw. die sie repräsentierenden Signale einer Apparatur nach dem Schema der Abb. 1.1 zugeführt werden, die am Ausgang die neuen Daten liefert.

Informationsverarbeitung geht also stets mit der Übertragung von Daten einher und besteht eigentlich nur aus dieser Übertragung sowie einer Umcodierung, die dann die Transformation oder Strukturierung darstellt.

Auf der Ebene der Signalverarbeitung bezeichnet man die selektive Verarbeitung auch als **Filtern**. Eine Verarbeitung, aus der man die ursprünglichen Informationen zurückgewinnen kann, heißt **informationstreu** oder **verlustfrei**. Auf der Ebene der Datenverarbeitung kann sowohl eine selektierende wie eine strukturierende Verarbeitung zugleich **komprimierend** sein, d. h. sie codiert die gleiche Information mit weniger Bits. Da eine komprimierende Verarbeitung meist zugleich informationstreu sein soll, kann eine selektierende Verarbeitung nur dann komprimieren, wenn die ursprüngliche Codierung redundant war.

Technisch kann man sehr große Datenmengen oft ohne Kompression überhaupt nicht speichern oder übertragen, weil der Aufwand zu hoch wäre. Handelt es sich bei den Daten um Text, so muß die Kompression auf jeden Fall informationstreu sein. Bei der Speicherung von Bildern

15. Aus der Sicht der Informationsverabeitung ist die Information wirklich neu und vorher nicht da gewesen. Aus mathematischer Sicht ist es sinnvoll, alle überhaupt denkbaren Ergebnisse als *a priori* vorhanden anzusehen. Auch die Addition 3 + 4 wird dann zur Auswahlvorschrift, die aus den potentiellen Ergebnissen eines selektiert.

als zweidimensionale Folgen von (farbigen oder schwarzweißen) Einzelpunkten, sogenannten **Pixeln**, wäre aber ein gewisser Informationsverlust durchaus zu ertragen, da oft eine selektierende Verarbeitung folgt, etwa wenn das Wiedergabegerät eine geringere Auflösung besitzt als die ursprüngliche Aufzeichnung. Verlustfreie Kompression erreicht heute Raten zwischen 2:1 und 8:1. Nicht-verlustfreie[16] Kompression erreicht bei Bildern oder gesprochener Sprache Raten von 100:1.

1.1.5 Semiotik: Syntax, Semantik und Pragmatik

Interpretation setzt Daten untereinander und mit Gegenständen außerhalb der Informatik in Beziehung. Ziel der Interpretation ist es letztlich immer, Wissen über Gegenstände außerhalb der Informatik zu gewinnen und zu formulieren; Wissen über Gegenstände innerhalb der Informatik ist Mittel zum Zweck.

Damit sich Daten als Repräsentation von Wissen eignen, muß man in einfacher Weise das repräsentierte Wissen entnehmen und gegebenenfalls modifizieren können. Inhaltlich müssen die Daten im Sinne der Interpretationsvorschrift **widerspruchsfrei** oder **konsistent** sein: es dürfen sich keine Einsichten aus den Daten herauslesen lassen, die dem darzustellenden Wissen widersprechen. Inkonsistenz kann die Folge einer fehlerhaften Interpretationsvorschrift sein. Sie kann aber auch inhärent sein: die Daten eignen sich nicht zur Darstellung des Wissens. Im letzteren Fall sagen wir, die Daten seien *kein Modell* für dieses Wissen.

Ob Daten ein Modell für irgendwelches Wissen bilden, entscheidet sich aufgrund der Beziehungen zwischen den Daten selbst. Wenn die Daten in einem einfachen Zeichenvorrat, beispielsweise einem Binärcode, codiert vorliegen, sind zunächst nur Beziehungen erkennbar, die sich unmittelbar aus der Anordnung der Zeichen ergeben. Diese strukturellen Beziehungen heißen **syntaktische Struktur** oder kurz **Syntax** der Daten. Nur die Syntax der Daten ist beobachtbar und meßbar.

Weitergehende Beziehungen zwischen den Daten bezeichnen wir als ihre **Semantik**. Sie ergeben sich aus einer Interpretationsvorschrift. Wissen, das sich ergibt, indem man Daten zu Gegenständen oder Sachverhalten außerhalb der vorgegebenen Datenmenge in Beziehung setzt, bezeichnen wir als ihre **pragmatische Bedeutung** oder kurz **Pragmatik**.

„das Haus“ ist syntaktisch eine Anreihung von 7 Buchstaben, in die nach dem dritten Buchstaben ein Zwischenraum eingeschoben ist. Daß der Zwischenraum Wörter trennt und wir somit zwei Wörter vor uns haben, ist Bestandteil der Semantik. Daß „das Haus“ ein Begriff für Gegenstände der realen Welt ist, gehört zur Pragmatik.

Syntax, Semantik und Pragmatik sind die drei Teile der **Semiotik**[17], der

16. engl. *lossy*.
17. griech. σῆμα, Zeichen.

Lehre von den Zeichen, wie sie von Peirce und Morris[18] begründet wurde, vgl. dazu (Zemanek , 1992) und (Zemanek , 1966). Oft spricht man fälschlich von *Semantik*, wo eigentlich *Semiotik* gemeint ist.

1.2 Wirklichkeit und Modell

1 *Die Welt ist alles, was der Fall ist.*
1.1 *Die Welt ist die Gesamtheit der Tatsachen, nicht der Dinge.*
2 *Was der Fall ist, die Tatsache, ist das Bestehen von Sachverhalten.*
3 *Das logische Bild der Tatsachen ist der Gedanke.*
3.0.1 *Die Gesamtheit der wahren Gedanken sind ein Bild der Welt*
Ludwig Wittgenstein, aus dem *Tractatus logico-philosophicus*
(Wittgenstein , 1970)

Information befriedigt die menschliche Neugier. Sie verhilft zu Einsichten über Vergangenes oder Bestehendes. Eine Einsicht ist hier ein Gedanke im Sinne Wittgensteins, ein Modell, das wir uns von realen Gegenständen machen und das die Zusammenhänge und Funktionsweise dieser Gegenstände erklären soll.

Eine Einsicht, ein Gedanke oder ein Modell ist **wahr**, wenn die Tatsachen (des betrachteten Ausschnitts) der Wirklichkeit richtig wiedergegeben werden. Wir erkennen dies daran, daß wir Schlüsse aus dem Modell ziehen können, die Sachverhalten in der Wirklichkeit entsprechen. Wir haben es mit Gegenständen auf zwei verschiedenen Ebenen zu tun:

Wirklichkeit W: Dinge, Personen, Abläufe in der Zeit, Beziehungen zwischen diesen Gegenständen (Tatsachen), . . .

Modell M: Begriffe von (real existierenden oder nur gedachten) Dingen, Personen, Abläufen in gedachter Zeit, Beziehungen zwischen diesen Begriffen (logisch mögliche Sachverhalte), . . .

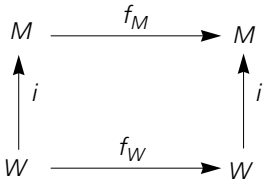

Abbildung 1.2: Beziehung zwischen Wirklichkeit und Modell

Ist i die Abbildung, die den wirklichen Gegenständen und Sachverhalten in W ihren Begriff in M zuordnet, und ist f_M eine beliebige Beziehung (auch zeitlicher Natur) zwischen den Begriffen der Gegenstände oder deren Zuständen,

18. Charles S. Peirce, 1839 – 1914, und Charles W. Morris, 1901 – 1979, amerikanische Philosophen.

so gibt es bei einem zutreffenden Modell auch eine reale Beziehung f_W so, daß beim Hintereinanderschalten der Zuordnungen $f_M \circ i = i \circ f_W$ gilt. Das Diagramm der Abb. 1.2 kommutiert.

Der Übergang in die Modellwelt muß nicht aus einem Schritt bestehen. Wir können z. B. über die wirklichen Gegenstände Daten sammeln und diese dann ein- oder mehrstufig interpretieren, um zu den Begriffen der Modellwelt zu gelangen. Insbesondere kann das Diagramm selbst mehrstufig sein: Wir können auch ein Modell als reale Welt auffassen und darüber weitere Modellierungsebenen legen. Man sieht, daß die Zuordnung i eine Verallgemeinerung der Interpretationsvorschriften aus Abschnitt 1.1 ist.

Der Mensch nutzt Modelle nicht nur, um Einsichten in Vergangenes oder Bestehendes zu gewinnen. Er möchte insbesondere Aussagen über zukünftiges Verhalten machen. Wenn der Statiker den Plan einer Brücke gutheißt, so sagt er damit aufgrund seiner Modelle vorher, daß die (noch zu bauende) Brücke nicht einstürzen wird. Zugefügte Floskeln wie „nach menschlichem Ermessen“ oder „nach dem heutigen Stand der Technik“ weisen darauf hin, daß man sich des Wahrheitsgehalts der Modellwelt nicht völlig sicher ist.

Informationsverarbeitung dient vor allem der Konstruktion von Modellen, um Aussagen über die (modellierte) Wirklichkeit zu gewinnen. Diese Aussagen kann man nutzen, um steuernd oder regelnd in die Wirklichkeit einzugreifen.

Beispiel 1.1: Wenn jemand seinen Garten gießt, so möchte er damit den Gartenboden, der nach den Kriterien seiner Modellwelt zu trocken ist, in einen weniger trockenen Zustand überführen. Nachdem er 20 Minuten gegossen hat, kommt er zu dem Schluß (in der Modellwelt!), daß der erwünschte Zustand des Gartenbodens erreicht sei. Ob dies wirklich wahr ist, bleibt offen. ♦

Wie im Beispiel nutzen wir Modellvorstellungen insbesondere, um Zustandsübergänge von Gegenständen der realen Welt zu **steuern**. Wäre der Gärtner mit der gleichen Sorgfalt vorgegangen wie der Statiker, so hätte er an mehreren Stellen die Bodenfeuchtigkeit messen müssen, um die Aussage, daß der Boden zuvor zu trocken und hernach ausreichend feucht sei, zu überprüfen. Er hätte ferner überlegen müssen, ob die (beschränkte Anzahl der) Meßstellen repräsentativ sind für den Zustand des gesamten Gartens. Er könnte möglicherweise Wasser sparen, wenn er die Feuchtigkeit während des Gießens messen und rechtzeitig die Wasserzufuhr abstellen würde. In der Sprache der Regelungstechniker läge dann ein **Regelkreis** wie in Abb. 1.3 vor. Hier ist der Prozeß des Gießens nicht mehr nur gesteuert; die Regelstrecke koppelt das Ergebnis der Wasserzufuhr über die Meßgröße *Feuchtigkeit* zurück. Die **Regelung** selbst würde der Gärtner übernehmen; das Verfahren könnte aber auch automatisiert werden.

Wenn die ‚Wirklichkeit‘ real ist und nicht selbst der Welt des Denkens entstammt, können wir den Wahrheitsgehalts eines Modells nur durch Experimente

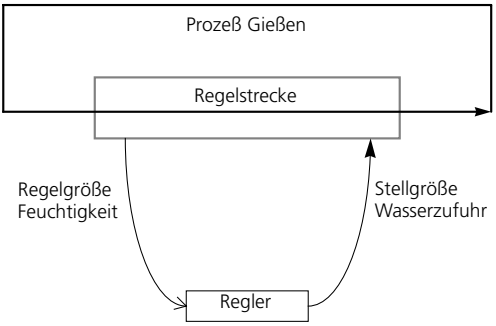

Abbildung 1.3: Regelkreis

feststellen. Diese Überprüfung des Wahrheitsgehalts heißt eine **Validierung** des Modells. Logische Schlüsse helfen nicht weiter, da sie sich ebenfalls nur auf die Modellwelt beziehen, nicht auf die Wirklichkeit. Der Mensch neigt dazu, durch weitere Abstraktion das Modell so zu ändern, daß die Informationsverarbeitung einfacher wird. Ob das neue Modell noch der Wirklichkeit entspricht, muß überprüft werden. Realitätsverlust wegen Nichtübereinstimmung von Modell und Wirklichkeit ist nicht nur in Informatikanwendungen eine häufige und meist nur schwer zu entdeckende Fehlerursache.

Anders verhält es sich, wenn die Wirklichkeit selbst unserer Denkwelt entstammt, etwa der Mathematik. Die Wirklichkeit ist dann durch eine Beschreibung, etwa eine Textaufgabe, gegeben. Wir nennen diese Beschreibung eine **Spezifikation**. Ihre Übereinstimmung mit dem Modell läßt sich mit logischen Schlüssen prüfen. In der Praxis scheitert das allerdings oft an nicht ausreichender Strukturierung der Beschreibung und daher zu großer Komplexität der Prüfaufgabe. Hier ist es die Aufgabe der Mathematik und Informatik, Techniken bereitzustellen, um die Strukturierung sowohl der Spezifikation als auch der Darstellung des Modells zu unterstützen. Die Validierung eines Modells gegen eine Spezifikation durch logische Schlüsse heißt eine **Verifikation** des Modells.

Insbesondere können wir auf dem Weg von der Wirklichkeit zum Modell Zwischenschritte einfügen, indem wir zunächst ein für unsere Zwecke vielleicht noch nicht geeignetes Modell konstruieren, das wir aber experimentell mit der Wirklichkeit vergleichen können. In weiteren Schritten wird dann dieses Modell als Wirklichkeit betrachtet und dafür ein neues Modell erstellt.

Die Konstruktion von Modellen kann zur Veränderung der Wirklichkeit führen, wenn wir mit Hilfe der Information aus dem Modell unmittelbar steuernd oder regelnd in die Wirklichkeit eingreifen. Das umfangreichste Beispiel hierfür sind unsere Telematiksysteme, wie z. B. das Telefon. Wenn wir eine Telefonnummer wählen, so wird der Verbindungsaufbau zwischen den Teilnehmern

heute weitgehend von Software gesteuert. Wenn es eine falsche Verbindung gibt, so kann dies zwar an technischen Fehlern liegen. Wahrscheinlicher ist jedoch, daß die im Modell vorgesehenen Interpretationsvorschriften nur unzureichend durch die Software realisiert wurden. In diesem Fall ist also die Realisierung des Modells, aber nicht dessen Validierung das Problem: letztere ist *per definitionem* gewährleistet.

Ob ein Modell, seine Realisierung und die verursachte Änderung der Wirklichkeit den Wünschen der jeweiligen Auftraggeber und Benutzer entspricht, ist eine andere Frage. Sie läßt sich im Prinzip bereits anhand der Spezifikationen prüfen. Die Antwort ersetzt die Validierung des Modells und heißt gewöhnlich ebenfalls Validierung. Allerdings führt die veränderte Wirklichkeit oft zu neuen oder veränderten Wünschen an das Modell und seine Realisierung. Die Validierung ist also nicht auf Dauer gültig.

1.2.1 Die Verantwortung des Informatikers

Ein Modell kann schrittweise auch technisch interpretiert werden. Jedoch bleibt die Verantwortung für sämtliche Interpretationsschritte immer beim Menschen: Er hat das Modell erdacht. Wenn es die Wirklichkeit wiedergeben soll, um hierauf Planungen und Vorhersagen aufzubauen, muß man zuvor überlegt haben, daß das Modell tatsächlich mit der Wirklichkeit übereinstimmt, und dies auch erhalten bleibt. Hierzu muß man passende Prüfmechanismen einbauen.

Modelle geben immer nur Ausschnitte der Wirklichkeit wieder. Es gibt daher Schlüsse, die man aus dem Modell *nicht* ziehen darf, da sie von im Modell nicht erfaßten Aspekten abhängen. Die genaue Charakterisierung der zulässigen und unzulässigen Schlußfolgerungen steht in der Verantwortung dessen, der das Modell erdacht hat. Man trägt dieser Verantwortung gewöhnlich Rechnung, indem man positiv den Zweck oder die Ziele des Modells angibt.

Wenn das Modell dazu dient, steuernd oder regelnd in die Wirklichkeit einzugreifen, gehört es zur Verantwortung des Konstrukteurs, zu überlegen, welche weitergehenden Konsequenzen dieser Eingriff in die Wirklichkeit haben könnte.

Diese Bemerkungen beziehen sich natürlich nicht nur auf Modelle und Eingriffe in die Wirklichkeit, die Leistungen der Informatik sind oder diese einschließen. Jeder Ingenieur, Arzt, Naturwissenschaftler usw. muß sich dieser Verantwortung ebenso stellen wie der Politiker, Journalist und Schriftsteller. Oftmals ist es auch nicht allein die Aufgabe des Entwerfers eines Modells, die Konsequenzen zu erörtern, sondern es bedarf der Mitwirkung vieler weiterer Menschen. Es liegt allerdings in der Natur der Sache, daß die Diskussion über den verantwortlichen Einsatz gewisser Techniken oftmals ganz andere Gesichtspunkte in den Vordergrund rückt, als die letztendlich wichtigen, weil diese noch gar

nicht absehbar sind: Daß die Erfindung von Eisenbahn, Auto und Flugzeug als soziale Änderung die *mobile Gesellschaft* hervorbringen und damit auch die gesellschaftlichen Wertvorstellungen der Menschen in der Industriegesellschaft ändern würde, war ebensowenig absehbar wie zahlreiche gesellschaftliche Implikationen des Fernsehens oder des Einsatzes von Rechnern und Telekommunikation.

Modelle und Verfahren der Informatik sind ebenso wie andere naturwissenschaftliche oder technische Fortschritte nicht *a priori* gut oder schlecht: Dieselben Verfahren der Verwendung von Datenbanken, die unter Datenschutzgesichtspunkten geeignet sein könnten, die Würde des Menschen zu verletzen, werden auch zur Organisation von Unternehmen verwendet und sichern damit Arbeitsplätze, oder dienen dem Arzt, um aus einer Vielzahl von Fällen ähnliche Krankheitsbilder herauszusuchen, damit er einem Kranken zum Besseren verhelfen kann. Die Verantwortung des Informatikers bezieht sich darauf, beide Gesichtspunkte und ihre möglichen Implikationen abzuwägen.

In einer Welt offener Handelsgrenzen, in der sich Informationen und speziell Ergebnisse und Produkte der Informatik in Sekunden von Kontinent zu Kontinent transportieren lassen, gehört es insbesondere zur Verantwortung des Informatikers, die Konsequenzen der so ermöglichten internationalen Arbeitsteilung zu bedenken: Um eine Technik so zu formen, daß ihr Einsatz ethischen und sonstigen Ansprüchen gerecht wird, muß man die Technik zunächst in einer wirtschaftlich tragfähigen Weise beherrschen; andernfalls läuft man Gefahr, daß die Verfahren und Produkte in einem Zustand importiert werden, der den Wertvorstellungen anderer Länder und Kulturen, aber nicht den eigenen Wertvorstellungen entspricht. Gerade in der Entwicklung der Informatik in Deutschland gibt es zahlreiche Beispiele, daß Produkte eingesetzt wurden und werden, die den zuvor aufgestellten Ansprüchen nicht gerecht werden, weil man versäumt hat, die wirtschaftlichen Möglichkeiten zu schaffen, um bessere Produkte zu produzieren und zu vertreiben.

1.3 Systeme

> *The engineer, and more generally the designer, is concerned with how things* ought *to be — how they ought to be in order to* attain goals*, and to* function.
> Herbert A. Simon (Simon , 1996), Kap. 1

Das Beispiel der Telematiksysteme zeigt ebenso wie das Gartengießen, daß sich der betrachtete Ausschnitt der Wirklichkeit und seiner Modelle aus vielen Teilen zusammensetzt, die untereinander in komplizierten Wechselbeziehungen stehen. Hierfür ist der Begriff eines *Systems* grundlegend. Seine Anwendung erfordert Einzelkenntnisse, die wir uns in den nachfolgenden Kapiteln verschaffen.

Unter einem **System**[19], versteht man eine Kollektion von Gegenständen, die in einem inneren Zusammenhang stehen, samt den Beziehungen zwischen diesen Gegenständen. Die Gegenstände bezeichnen wir auch als **Systemkomponenten** oder **Bausteine** (des Systems). Dieser Begriff ist sehr allgemein und läßt sich auf natürliche Systeme genau so anwenden wie auf technische, soziale, organisatorische, theoretische Systeme, usw.

Systeme sind mit Ausnahme des Systems *Weltall* in eine Umgebung eingebettet. Es gibt eine Systemgrenze, welche die zum System gehörigen von den zur Umgebung gehörigen Gegenständen scheidet. Ein System kann **abgeschlossen** oder **offen** sein. Die Bausteine abgeschlossener Systeme haben keine Beziehungen zu den Gegenständen in der Umgebung; insbesondere ist die Systemgrenze fest und zeitlich nicht veränderlich. Bei offenen Systemen gibt es solche Beziehungen, auch kann die Zugehörigkeit von Gegenständen zum System wechseln; die Systemgrenze kann sich in der Zeit verschieben. Die Systemgrenze, auch die Grenze zwischen Systembausteinen, und die Beziehungen, die über diese Grenzen laufen, nennen wir die **Schnittstelle** (des Systems oder Bausteins). Abgeschlossene Systeme gibt es nur in der Theorie; der Begriff stellt eine Idealisierung dar. Mit ihm teilt man mit, daß man bewußt die Beziehungen zur Umgebung vernachlässigt, weil sie im gegebenen Zusammenhang unbedeutend sind.

Ein System kann **statisch** oder **dynamisch**, d. h. (**zeitlich**) **veränderlich** sein. Der Begriff *statisches System* ist ebenfalls eine Idealisierung. Er bedeutet, daß die zeitliche Veränderung im gegebenen Zusammenhang keine Rolle spielt.

Die zeitliche Veränderung kann verschiedene Aspekte umfassen:

- die Beziehungen zwischen den Bausteinen ändern sich in der Zeit;
- die Menge der Bausteine ändert sich, weil sich die Systemgrenze verschiebt: Komponenten wechseln in die Umgebung oder umgekehrt;
- die Menge der Bausteine ändert sich, weil Bausteine verschwinden oder neue Bausteine hinzugefügt werden; ein Beispiel in der Natur ist die Zellteilung.
- die Menge der Bausteine und ihrer Beziehungen bleibt gleich, aber die Eigenschaften mancher Bausteine ändern sich.

Die Änderungen eines Systems beziehen sich sämtlich auf seinen **Zustand**: das System als solches bleibt gleich, es behält seine Identität und seinen Namen. Der Zustand besteht aus den jeweils vorhandenen Komponenten, ihren Eigenschaften und ihren Beziehungen. Die Folge der Zustände bezeichnet man auch als das **Verhalten des Systems**. Wenn wir ein System in dieser Weise als identifizierbare Einheit mit zeitlich veränderlichem Zustand betrachten, ohne uns in seine Bausteine und deren Beziehungen zu vertiefen, bezeichnen wir es als **Objekt**.

19. griech. σύστημα, Vereinigung.

Die Vergangenheit eines dynamischen Systems ist eindeutig festgelegt, aber nicht unbedingt in allen Einzelheiten bekannt. Im Hinblick auf das zukünftige Verhalten unterscheidet man zwischen **deterministischen** und **indeterministischen** Systemen. Im ersten Fall legt die Vergangenheit eindeutig auch das zukünftige Verhalten fest. Im zweiten Fall ist das zukünftige Verhalten zufällig oder hängt von derzeit nicht bekannten oder nicht zum System gehörigen Einflußgrößen ab.

Offene dynamische Systeme sind in der Regel indeterministisch. Ihr zukünftiges Verhalten hängt von den im System nicht vorhersagbaren Beziehungen zur Umgebung ab.

Der Systembegriff ist **rekursiv**: Die Komponenten eines Systems können selbst wieder (**Teil-**)**Systeme** sein. Wir unterscheiden dabei zwei Fälle:

1. Für das Verständnis des Gesamtsystems sind nur die Zustände der Teilsysteme *als Ganzes* interessant; die Teilsysteme werden als Objekte aufgefaßt;
2. Für das Verständnis des Gesamtsystems muß man die Beziehungen und möglichen zeitlichen Änderungen innerhalb der Teilsysteme *im einzelnen* studieren.

Im ersten Fall sagen wir, ein Teilsystem wird als **schwarzer Kasten**[20] angesehen. Im zweiten Fall sprechen wir von einem **weißen Kasten**.

Für alle Wissenschaften sind solche rekursiven Begriffe besonders wichtig, weil sie **skalierbar** sind: Das Studium der Eigenschaften von Systemen im Kleinen liefert Erkenntnisse, die man auf größere Systeme übertragen kann.

Viele Wissenschaften befassen sich mit der Analyse existierender Systeme. Dabei setzt man zur Analyse und Modellierung Hilfsmittel nicht nur der Mathematik, sondern auch der Informatik ein. Ziel ist es jeweils, mit einem Modell des Systems das beobachtete Verhalten des Systems zu erklären und sein zukünftiges Verhalten vorherzusagen. Häufig sehen wir die Systeme als **natürliche Systeme** an, selbst wenn sie vom Menschen konstruiert wurden, wie z. B. soziale Organisationen. Hingegen sind die Modelle, die wir uns davon machen **künstliche Systeme** oder **Artefakte**[21]: Sie dienen einem ganz bestimmten Zweck, ihrem **Systemziel**, und sind auf diese Aufgabe hin optimiert. Wirtschaftliche Organisationen und alle Systeme der Technik sind Beispiele für Artefakte in der realen Welt.

1.3.1 Die Aufgaben von Informatik-Systemen

Die Aufgabenstellungen, die wir heute mit Informatik-Systemen bearbeiten, lassen sich nach ihren Systemzielen in vier Klassen einordnen:

20. Das englische *black box* ist eine Übersetzung dieses 1905 von dem österreichischen Physiker und Philosophen Ernst Mach, 1838 – 1916, (Mach , 1991), geprägten Begriffs.

21. lat. *arte factum*, durch Kunst gemacht.

- **Berechnung von Funktionen**: Berechne eine Funktion $f: A \to B$, die einen Definitionsbereich A in einen Wertebereich B abbildet.
- **Prozeßüberwachung**: Konstruiere ein (im Prinzip) endlos laufendes System $\mathcal{S}$, das Daten a von anderen Systemen (Prozessen) empfängt und Daten b an solche Systeme sendet. Die Daten b sind funktional abhängig von der bisherigen Systemhistorie, d. h. der Zeit sowie den bisher empfangenen und ausgesandten Daten. $\mathcal{S}$ speichert hierzu lokal Daten, die zumindest ausschnittsweise Informationen aus der Prozeßhistorie repräsentieren. Die ausgesandten Daten können insbesondere auch den Start oder den Halt anderer Systeme (Prozesse) bewirken.
- **Eingebettete Systeme**: Konstruiere Systeme, die im Verbund mit Bausteinen, die nicht der Datenverarbeitung dienen, eine zeitlich abgegrenzte oder endlos laufende Aufgabe lösen. Die anderen Systemkomponenten können technische Apparaturen, aber auch Menschen oder betriebliche Organisationen umfassen.
- **Adaptive Systeme**: Konstruiere ein eingebettetes System, das sich Veränderungen der Wirklichkeit anpaßt, insbesondere solchen, die das System selbst hervorruft. Das ursprüngliche Modell ist nicht auf Dauer gültig.

Diese Aufzählung ist eine Hierarchie aufsteigenden Umfangs und wachsender Komplexität der zu lösenden Aufgaben.

Prozeßüberwachung, eingebettete und adaptive Systeme bezeichnet man zusammenfassend als **reaktive Systeme**. Ihre Aufgabe besteht nicht darin, ein abschließendes Ergebnis $f(x)$ zu berechnen, sondern fortwährend auf Ereignisse und Daten aus ihrer Umwelt zu reagieren.

1.3.2 Konstruktion von Informatik-Systemen

In der Modellierung der Wirklichkeit mit Informatik-Hilfsmitteln analysiert man zunächst natürliche oder künstliche Systeme der realen Welt im Hinblick auf ihre externen und internen Eigenschaften und Verhaltensweisen (**Systemanalyse**). Darauf aufbauend entwickelt man künstliche **Modellsysteme**, die entweder das vorgefundene reale System wiedergeben, oder deren Realisierung das vorgefundene System ganz oder in Teilen ersetzen soll, oft unter Bereitstellung zusätzlicher Möglichkeiten. Das Modellsystem muß dann analysiert werden, um zu prüfen, ob es wirklichkeitsgetreu ist und die gewünschten Systemziele erreicht. Die Analyse kann in einfachen Fällen in geschlossener Form mit mathematischen Hilfsmitteln oder Methoden der theoretischen Informatik erledigt werden. In komplizierteren Fällen greift man zum Hilfsmittel der **Simulation**: Man realisiert die Systemkomponenten, ihre Beziehungen untereinander und nach außen, sowie ihr zeitliches Verhalten soweit, daß man mit Rechnerunterstützung die

gewünschten Aussagen über das Erreichen der Systemziele an Einzelfällen überprüfen kann[22].

Simulation ist eine Vorstufe der **Realisierung** des Modells mit Hilfsmitteln der Informatik. In der Simulation soll das Verhalten des Systems quantitativ ermittelt werden; z. B. wird geprüft, ob Zeitbedingungen auch eingehalten werden, ob die Teilsysteme einwandfrei zusammenarbeiten, und ob keine wesentlichen Einsatzbedingungen übersehen wurden. Aus der Simulation des Entwurfs eines integrierten Schaltkreises kann man mittelbar auch auf die funktionalen Eigenschaften des Schaltkreises schließen. In vielen anderen Fällen bleibt die eigentliche Funktion des simulierten Systems unberücksichtigt. Beim Einsatz von Datenbanken würde man also überprüfen, ob das System schnell genug auf die *typischen* Anfragen reagiert und ob hierzu die vorhandenen Betriebsmittel (Speicher, Prozessorleistung, Übertragungsgeschwindigkeit) ausreichen. Ob das System die Anfragen auch richtig beantwortet und die Datenbank in konsistentem Zustand erhält, oder ob es eine für den Anwender *akzeptable* Bedienoberfläche am Bildschirm vorsieht, wird durch die Simulation nicht beantwortet. Hierfür konstruiert man als eine andere Vorstufe der Realisierung einen **Prototyp**; dieser dient der Beurteilung der funktionalen Beziehungen zwischen den Systemkomponenten und der Systemumgebung, jedoch oft unter Verzicht auf die in der Simulation untersuchten Leistungsmerkmale.

Wenn ein Systemmodell nur konstruiert wurde, um das Verhalten eines realen Systems zu analysieren, ist die Aufgabe mit der Analyse der Modelleigenschaften und den daraus zu ziehenden Schlüssen hinsichtlich des realen Systems abgeschlossen. In den meisten Fällen dienen Systemmodelle jedoch dem Ziel, ein neues System zu konstruieren, das im Vergleich zum ursprünglichen System veränderte Systemziele verfolgt. Die **Ist-Analyse** des vorhandenen Systems liefert dann Kenntnisse der unbedingt notwendigen Eigenschaften des neuen Systems. Hinzu kommen muß eine **Soll-Analyse**, die die neuen oder veränderten Systemziele festlegt. Da das neue System, ein Artefakt, oft als schwarzer Kasten in seine Umgebung eingebettet wird, ist die interne Struktur des neuen Systems frei gestaltbar. Die Schnittstelle zwischen System und Umgebung ist dagegen festgelegt. Sie ist nicht immer optimal, wenn sie nur durch Modifikation aus dem bisherigen Systemmodell abgeleitet wird, obwohl solche Modifikationen der Normalfall der Konstruktion neuer Systeme sind. Gegebenenfalls muß das neue Systemmodell wiederum durch Simulation und Konstruktion eines Prototyps validiert werden.

Mit solchen Vorarbeiten ist der Weg bereitet, um ein neues System zu realisieren. Nach den vorstehenden Überlegungen besteht diese Realisierung aus folgenden Schritten:

22. Auf die sehr schwierige Frage, wieweit diese Einzelfälle für das Verhalten des modellierten Systems repräsentativ sind, gehen wir hier nicht ein.

- Entwurf des inneren Aufbaus des Systems (welche Komponenten, welche Beziehungen zwischen diesen und der Außenwelt, welche Zustände und Zustandsübergänge usw. sind nötig);
- gegebenenfalls rekursiver Entwurf der Teilsysteme, die als Komponenten auftreten;
- Festlegung der Verfahren, mit denen die Komponenten und ihre statischen und dynamischen Beziehungen realisiert werden sollen;
- Konstruktion der Komponenten und der Funktionen, die das Zusammenspiel regeln (die eigentliche Realisierung);
- Integration aller Komponenten und Funktionen;
- Integration des Systems in seine Umgebung.

Die Schritte werden nicht immer in einer festen zeitlichen Abfolge ausgeführt. Jeder Schritt muß begleitet sein von der Validierung der bisher getroffenen Entscheidungen: Ist das System, soweit bisher entwickelt, funktional richtig? Erreicht es die Systemziele? Sind die weiteren Schritte mit Aussicht auf Erfolg durchführbar?

In der Praxis gehören an den Anfang und das Ende der Gesamtkonstruktion und jeden Einzelschritts die Abschätzung der voraussichtlich mit der Konstruktion verbundenen Kosten im Vergleich zum erzielbaren Nutzen, sowie die Kontrolle der tatsächlichen Kosten im Vergleich zur Vorgabe. Da Kosten und Nutzen erheblich von der Zeitplanung abhängen, ist die Kontrolle der Termine (Meilensteine) integraler Teil der Kostenkontrolle.

Wir benötigen den Systembegriff in der Informatik also zur Beschreibung der vorgefundenen Wirklichkeit und zur Modellierung geplanter Artefakte. In den weiteren Kapiteln müssen wir uns damit befassen, welche Hilfsmittel uns zur Modellierung von Systemen, zu deren Analyse, sowie zur Konstruktion und Validierung von Informatik-Systemen zur Verfügung stehen. Hierzu benötigen wir nicht nur Methoden aus der praktischen und technischen Informatik, sondern insbesondere in großem Umfang Systembegriffe aus der Mathematik und der theoretischen Informatik.

1.4 Algorithmen

Die einfachste Aufgabe eines Informatik-Systems ist die Realisierung einer Funktion $f: A \to B$. Solche Funktionen bilden den Kern aller umfangreicheren Systeme. Wir können solche Funktionen auf verschiedene Weisen beschreiben:

- **statisch** oder **deklarativ**: Globale Darstellung der Zusammenhänge zwischen A und B unter Benutzung von Gleichungen, z. B. auch in der impliziten Form $g(a, b) = 0$, wenn $b = f(a)$. Diese Beschreibung verhilft auf den ersten Blick nicht zu einer konstruktiven Regel, um $f(a)$ zu berechnen. Vielmehr

scheint sie sich nur zur Nachprüfung zu eignen, daß ein Resultat richtig ist. Während der Modellierung ist dies aber oft nicht nur ausreichend, sondern sogar erwünscht, weil es erlaubt, die Entscheidung, wie man $f(a)$ wirklich berechnen will, noch zu verschieben. Darüberhinaus werden wir sehen, daß es vielfach doch gelingt, aus einer solchen deklarativen Beschreibung eine konstruktive Berechnungsmethode herzuleiten.

- **tabellarisch**: Für alle Werte a des Definitionsbereichs A wird der zugeordnete Wert $f(a)$ im Wertebereich B in einer Tabelle festgehalten und bei Bedarf nachgeschlagen. Technisch setzt dies voraus, daß der Definitionsbereich A endlich und *nicht allzu groß* ist, wobei es von der Anwendung abhängt, was wir unter *groß* verstehen.
- **algorithmisch**, **operativ**, **prozedural** oder **synthetisch**: Wir geben eine Beschreibung zur Berechnung des Ergebnisses für beliebige Argumente a des Definitionsbereichs A. Die Beschreibung kann in einer beliebigen Sprache abgefaßt sein (einschl. natürlicher Sprache). Sie muß als Nachricht endlicher Länge codiert sein und muß die Berechnung als Folge von endlich vielen *elementaren Operationen* mit „hinreichender Präzision“ spezifizieren. Eine solche Beschreibung heißt ein **Algorithmus**[23].

Algorithmen sind die wichtigste Art der Beschreibung von Vorschriften zur Berechnung von Abbildungen. Die ältesten uns bekannten Algorithmen finden sich in den „Elementen“ von Euklid[24]. Sie umfassen Konstruktionsverfahren für Dreiecke aus gegebenen Stücken und den euklidschen Algorithmus zur Berechnung des größten gemeinsamen Teilers zweier ganzer Zahlen.

Kochrezepte, Anweisungen zur Einnahme von Arzneien und zahlreiche Bedienungsanleitungen für einfache Geräte (öffentliche Fernsprecher, ...) sind weitere Beispiele für Algorithmen. In vielen dieser Algorithmen sind mehrere Angaben — wir nennen sie im weiteren **Argumente** — erforderlich, die aus verschiedenen Bereichen stammen. Auch kann es mehrere Ergebnisse geben, die zu unterschiedlichen Wertebereichen gehören. In manchen Fällen, z. B. bei der Einnahme von Arzneien, ist nicht angegeben, was die möglichen Ergebnisse sind; stattdessen wird implizit unterstellt, daß der Algorithmus dem Systemziel *Wiederherstellung der Gesundheit* dient, ohne im einzelnen zu spezifizieren, wie man sich dies als *Ergebnis* vorzustellen hätte. Diese Beispiele zeigen, daß die Sprachen, in der Algorithmen formuliert sind, und der Grad an Präzision der

23. Benannt nach Mohamed Ibn Musa Alchwarizmi, geboren in Chiwa (Chorism), gestorben in Bagdad nach 846. Alchwarizmi schrieb unter anderem das erste Lehrbuch der Algebra (Alchwarizmi , 1831), aus dessen Titel sich das Wort *Algebra* herleitet. Er schrieb ferner eine Anleitung zum Rechnen mit *indischen Ziffern* (Alchwarizmi , 1963), den Vorläufern der arabischen Ziffern, die wir heute benutzen. In letzterem Werk werden die vier Grundrechenarten für ganze Zahlen, Sexagesimalbrüche und gemeine Brüche in Sexagesimalschreibweise erklärt.

24. Euklid, 4./3. Jahrh. v. Chr., griechischer Mathematiker.

Beschreibung in weiten Grenzen variiert. Es lassen sich jedoch immer endlich viele Grundoperationen oder -schritte unterscheiden, die nach endlich häufiger Anwendung das gewünschte Ergebnis liefern.

Kochrezepte zeigen, daß man oft ein umfangreiches Kontextwissen besitzen muß, um die Beschreibung richtig zu verstehen. Wir bemerken Grundoperationen wie *anbräunen*, *hinzufügen*, *abgießen*, *dünsten*, *schmoren*, *braten*, *kochen* usw. Durch Wörter wie *dann* oder *danach* wird angezeigt, daß die Operationen **sequentiell** nacheinander ausgeführt werden sollen. Es gibt aber auch Formulierungen der Form „während die Brühe kocht, schneide man das Fleisch in Würfel". Dies ist eine Aufforderung bestimmte Operationen **parallel**, d. h. gleichzeitig, durchzuführen. Das Wörtchen *und* findet sich in Kochrezepten in verschiedenen Bedeutungen: Es kann Teilsätze zur Beschreibung von Tätigkeiten verbinden, die man offensichtlich nacheinander ausführen muß, etwa in dem Satz „die Kalbshaxe abtrocknen und in dem heißen Fett anbräunen". *Offensichtlich* bedeutet hier, daß der Koch oder die Köchin eine **kausale Abhängigkeit** kennt, wonach man die erste Tätigkeit vor der zweiten ausführen muß. Die Abhängigkeit könnte auch darin bestehen, daß man die gleichen Hilfsmittel, etwa ein nur einmal vorhandenes Mixgerät, für beide Tätigkeiten verwenden soll. Andererseits sagt eine Formulierung wie „das Fleisch in Stücke schneiden und die Kräuter fein wiegen", daß wir am Ende erwarten, daß beide Tätigkeiten ausgeführt sind; die Reihenfolge, in der das geschieht, ist beliebig. Wenn mehrere Personen in der Küche arbeiten, könnten die Tätigkeiten parallel ausgeführt werden. Eine einzelne Person könnte die beiden Tätigkeiten auch abwechselnd ausführen, in der Informatik sagen wir **zeitlich verzahnt**[25]. Wenn alle diese Möglichkeiten offenstehen, sprechen wir von einer **kollateralen** Ausführung der Operationen.

Manchen Tätigkeiten ist im Kochbuch die Zeitangabe zugefügt, wie lange die Operation durchgeführt werden soll. Die Zeitangabe kann auch als Bedingung formuliert sein: „rühre die Sauce, bis sie braun wird" oder „rühre die Sauce; wenn sie braun wird, füge die Fleischstücke hinzu". Auch gibt es Tätigkeiten, die man nur unter bestimmten Umständen ausführen soll: „wenn die Sauce zu dickflüssig wird, gieße einen Schuß Milch zu".

Insgesamt stoßen wir im Kochbuch bezüglich der Reihenfolge und Ausführungsbedingungen von Tätigkeiten auf folgende Situationen:

Sequentielle Ausführung: diese wird gewöhnlich nur vorgeschrieben, wenn eine kausale Abhängigkeit zwischen den Tätigkeiten besteht oder die gleichen Ressourcen verwendet werden, so daß die gleichzeitige Ausführung der Tätigkeiten sinnlos ist.

Parallele bzw. kollaterale Ausführung: diese Art der Koordination von Tätigkeiten wird oft implizit angegeben und ist dann zulässig, wenn keine kausa-

25. engl. *time-shared* oder *merged in time*.

len Abhängigkeiten zwischen den Tätigkeiten bestehen. Im allgemeinen wird gefordert, daß zu irgendeinem Zeitpunkt alle angesprochenen Tätigkeiten beendigt sind. Wir nennen diesen Zeitpunkt eine **Parallelitätsschranke** und sprechen von **Schranken-Synchronisierung**[26]. Die in der Informatik auch bekannte **Endsynchronisierung**, bei der mehrere kollaterale oder parallele Tätigkeiten abgebrochen werden, sobald eine von ihnen ihre Aufgabe erfüllt hat, kommt im Kochbuch im allgemeinen nicht vor.

Bedingte Ausführung: die Tätigkeit wird nur ausgeführt, wenn eine Bedingung erfüllt ist. Die Formulierung kann durch weitere Alternativen angereichert sein, die ausgeführt werden, wenn die Bedingung nicht erfüllt ist.

Ausführung in Schleife: die Tätigkeit wird für eine bestimmte Zeitspanne, oder solange eine vorgegebene Bedingung erfüllt bzw. nicht erfüllt ist, ausgeführt. Dies schließt den Fall „führe Tätigkeit aus, bis Zielbedingung erfüllt“ ein.

Ausführung eines Unterprogramms: führe eine Tätigkeit aus, die anderswo beschrieben ist (und eventuell bei mehreren Gelegenheiten benutzt werden kann), und verwende das Ergebnis weiter. Das Unterprogramm kann Parameter haben.

Weitere Angaben zur zeitlichen Reihenfolge gibt es nicht! Man kann alle Abläufe, die im Kochbuch beschrieben werden, mit diesen Ablaufsteuerungen beschreiben. In der Informatik ist dies nicht anders. Man kann zeigen, daß die vorstehend genannten Verfahren zur Steuerung von Abläufen von Algorithmen ausreichen, um alle Möglichkeiten zu beschreiben.

1.5 von-Neumann-Rechner

Zur Realisierung — in der Informatik sagt man meist **Implementierung** — von Algorithmen mit technischen Hilfsmitteln benutzen wir heute **Rechner**[27]. Wie der Name sagt, sollte ein Rechner ursprünglich vor allem arithmetische Rechnungen durchführen können. Dazu benötigt man zusätzlich einen **Speicher**, eine Einrichtung, in der man Eingabedaten und (Zwischen-)Ergebnisse notieren kann. Schließlich kann der Fortgang des Rechnens von Datenwerten, z. B., ob eine Größe $x > 0$ ist, abhängen. Also benötigt man eine **Programmsteuerung**, die Entscheidungen über die nächsten Rechenschritte treffen kann.

Die Idee programmgesteuerter Rechner geht auf C. Babbage[28] zurück, der ab 1833 mit mechanischen Hilfsmitteln eine *analytische Maschine* baute, um

26. engl. *barrier synchronization*.

27. Konrad Zuse nannte sie in den 30er und 40er Jahren Rechenautomaten. Warum man im Deutschen 30 Jahre später die Wörter *Rechner* und *Rechenautomat* gegen ein englisches Wort tauschte, bleibt rätselhaft.

28. Charles Babbage, 1792 – 1871, Mathematikprofessor in Cambridge, England.

programmgesteuert zu rechnen. Der erste programmgesteuerte Rechner, der die in modernen Rechnern zu findenden Prinzipien verkörpert, war die Z3 von K. Zuse[29]. Sie arbeitete elektromechanisch. Die Struktur der meisten elektronisch arbeitenden Rechner geht zurück auf die Arbeiten von J. von Neumann[30] und seiner Mitarbeiter und ist als **von-Neumann-Rechner** bekannt. Er besteht heute im wesentlichen aus vier Stücken, vgl. Abb. 1.4:

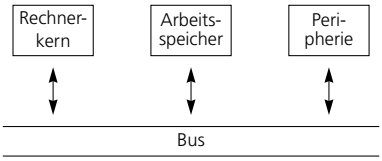

Abbildung 1.4: Princeton-Architektur eines von-Neumann-Rechners

1. dem **Rechnerkern** oder (**Zentral-**)**Prozessor**, der das Programm ausführt;
2. dem **Arbeitsspeicher**, der Programme und Daten enthält;
3. der **Peripherie**: Plattenspeicher, Drucker, Bildschirm, Tastatur, usw. und Kommunikationskanäle, die über **Ein/Ausgabe-** und **Kommunikationsprozessoren**, auch **Kanäle** oder E/A–Werke genannt, mit dem Rechnerkern und dem Arbeitsspeicher kommunizieren;
4. Verbindungswege oder **Busse**, die seriell oder parallel die vorgenannten Komponenten verknüpfen und Befehle, Adressen, Daten sowie Steuersignale zwischen ihnen übertragen können.

Ein solcher Rechner ist ein reaktives System, das im Rechnerkern fortlaufend **Befehle** ausführt. Ein Befehl ist dabei eine arithmetische Operation wie z. B. die Addition zweier Zahlen, eine logische Operation der Booleschen Algebra, die wir im nächsten Kapitel kennenlernen, oder eine der Anweisungen aus dem vorigen Abschnitt, die wir zur Steuerung der zeitlichen Reihenfolge brauchen. Ein **Programm** in einer **Maschinensprache** ist eine Folge von Befehlen, die den Rechnerkern anweist, bestimmte Operationen in bestimmter Reihenfolge auszuführen.

29. Konrad Zuse, 1910 – 1995, ursprünglich Bauingenieur, Erfinder zahlreicher Recheneinrichtungen. Er führte u. a. den Gebrauch des Dualsystems in modernen Rechnern ein und erfand 1937 die Gleitpunktdarstellung für reelle Zahlen. 1944 entwarf er mit dem Plankalkül einen Vorläufer der heutigen höheren Programmiersprachen. Ein Nachbau findet sich heute im Deutschen Museum in München.

30. Janos, Baron von Neumann, 1903 – 1957, ungarischer Mathematiker, nach David Hilbert der wohl universellste Mathematiker der ersten Hälfte des Jahrhunderts. Bahnbrechend waren seine Arbeiten über Gruppentheorie und ihre Beziehung zur Quantenmechanik, über Spieltheorie und über Automaten. Die nach ihm benannte Rechnerarchitektur formulierte er in (Burks, Goldstine und von Neumann , 1946).

Das Programm entnimmt der Rechner ebenso wie die Operanden dem Speicher, in den er auch die Ergebnisse zurückschreibt. Das Abholen eines Befehls oder Operanden aus dem Speicher und das Schreiben eines Ergebnisses nennt man einen **lesenden** bzw. **schreibenden Speicherzugriff**. Bei jedem Zugriff wird eine feste Anzahl n von Bits zwischen Rechnerkern und Speicher übertragen, z. B. $n = 64$. Der Rechnerkern kann heute Befehle um etwa den Faktor 10 schneller ausführen als Speicherzugriffe. Dieses Problem ist als **von-Neumann-Engpaß** bekannt. Man hat zahlreiche Möglichkeiten ersonnen, um die Anzahl der notwendigen Speicherzugriffe herabzusetzen. Dazu gehört insbesondere die Verwendung von **Registern**, d. h. lokalen Speicherelementen im Rechnerkern, die Benutzung eines kleinen, aber wesentlich schnelleren Zwischenspeichers[31] als Puffer, und die Verwendung getrennter Busse für den Zugriff auf Daten und Programm[32].

Der Speicher besteht aus **Speicherzellen**, die heute fast ausschließlich 8 Bits umfassen. Ein solches Oktett heißt ein **Byte**[33]. Die Speicherzellen sind beginnend mit 0 durchnumeriert. Die Nummer heißt die **Adresse** der Speicherzelle. Wenn Daten oder Befehle länger als ein Byte sind, z. B. 4 oder 8 Byte, dient die niedrigste Adresse eines der Bytes zugleich als Adresse des Datums oder Befehls.

Die Peripheriegeräte sind heute durchgängig mit einem eigenen Prozessor ausgestattet, der im Unterschied zum Zentralprozessor allerdings nur feste Spezialaufgaben mit fest vorgegebenen Programmen erledigt. Die Eingabe von Zeichen von der Tastatur oder die Ausgabe von Ergebnissen auf den Bildschirm oder Drucker wird heute genauso gesteuert wie die Kommunikation eines Rechners mit einem anderen. In vielen Fällen übernimmt dieser Spezialprozessor allerdings nur die Überwachung der Datenübertragung, während das Gerät die Daten selbständig aus dem Arbeitsspeicher holt oder dorthin schreibt. Man spricht dann von Geräten mit **direktem Speicherzugriff**[34].

Der Bus übernimmt nicht nur die Rolle der Verbindungsschiene zwischen den Einheiten. Er ist darüberhinaus für die Koordination der verschiedenen Übertragungswünsche zuständig. Dazu werden den Geräten einschließlich des Rechnerkerns Prioritäten zugeteilt. Bei konkurrierenden Übertragungsswünschen verschiedener Geräte entscheidet der Bus nach diesen Prioritäten, welcher Wunsch zuerst erfüllt wird.

Die technischen Geräte, die zusammen einen Rechner ausmachen, nennt man die **Hardware** (des Rechners), im Unterschied zu den Programmen und ihrer Dokumentation, die die **Software** bilden.

31. engl. *cache*.
32. Man spricht dann von einer Harvard-Architektur im Unterschied zu der Princeton-Architektur in Abb. 1.4 mit nur einem Bus.
33. *byte* ist auch im Englischen ein Kunstwort ohne umgangssprachliche Bedeutung.
34. engl. *direct memory access*, abgekürzt DMA.

Der Leser mache sich klar, daß heute die meisten Rechner nicht als PCs oder Arbeitsplatzrechner verkauft, sondern als sogenannte **Signalprozessoren** in technische Systeme, Automobile, Telekommunikationseinrichtungen, Fernsehgeräte, usw. eingebaut werden. Sie verrichten dort Spezialaufgaben der technischen Signalverarbeitung wie z. B. die Regelung der Benzinzufuhr zu Motoren oder die Steuerung von Videotext und der Kommunikation mit der Fernbedienung.

Trotz der Engpäße im Verkehr Prozessor-Speicher hat sich die von-Neumann-Architektur seit 50 Jahren bewährt. Dazu hat sicherlich die einfache Zerlegbarkeit in getrennt produzierbare Teilsysteme beigetragen. Einzelheiten der von-Neumann-Architektur und alternative Rechnermodelle werden in späteren Vorlesungen und vor allem in Vorlesungen über technische Informatik vertieft.

1.6 Semi-Thue-Systeme

Als einfache und zugleich allgemeine Form von Algorithmen betrachten wir die nach A. Thue[35] (Thue , 1914) benannten **Semi-Thue-Systeme**. Wir setzen einen endlichen Zeichenvorrat Σ voraus und betrachten Wörter $x = x_0 \cdots x_{n-1}$ von Zeichen x_i aus Σ. $|x| = n$ heißt die **Länge** eines solchen Wortes. Wir überführen x schrittweise in andere solche Wörter, indem wir Teilwörter $x_i \cdots x_{i+k-1}$ durch andere Wörter $y_j \cdots y_{j+l-1}$ ersetzen. Hierbei gelte $k, l \geqslant 0$, $i + k \leqslant n$. Für $k = 0$ bzw. $l = 0$ wird das *leere Wort* ε, das null Zeichen enthält, ersetzt oder als Ersetzungstext benutzt. Die Ersetzungsregeln schreiben wir

$$a \cdots b \to c \cdots d$$

und nennen $a \cdots b$ die linke, $c \cdots d$ die rechte Seite der Regel, die wir schematisch auch mit $p \to q$ bezeichnen. Es kann endlich oder abzählbar unendlich viele Regeln geben. Eine Regel heißt auf ein Wort x **anwendbar**, wenn x das Teilwort $a \cdots b$ enthält.

Beispiel 1.2: Über dem Zeichenvorrat $\Sigma = \{\mathsf{I}, +\}$ ergeben die Regeln

$$+\mathsf{I} \quad \to \quad \mathsf{I}+ \tag{1.1}$$

$$+ \quad \to \quad \varepsilon \tag{1.2}$$

einen Algorithmus, der natürliche Zahlen, dargestellt durch Folgen von Strichen I, addieren kann. Das Wort III + II wird schrittweise umgeformt:

$$\begin{aligned} \mathsf{III}+\mathsf{II} \quad &\Rightarrow \quad \mathsf{IIII}+\mathsf{I} \\ &\Rightarrow \quad \mathsf{IIIII}+ \\ &\Rightarrow \quad \mathsf{IIIII}. \end{aligned}$$

♦

35. Axel Thue, 1863 – 1922, norwegischer Mathematiker und Logiker.

Ein Übergang $l \Rightarrow r$ beschreibt dabei die Transformation, die durch Anwendung einer Regel $p \to q$ auf einen Teil der linken Seite l entsteht. r heißt aus l **abgeleitet** oder **erzeugt**. Die Transformation heißt eine direkte **Ableitung**. Wir schreiben $l \overset{+}{\Rightarrow} r$, wenn r aus l durch fortgesetzte Ableitung gewonnen werden kann. $l \overset{*}{\Rightarrow} r$ bedeutet entweder $l \overset{+}{\Rightarrow} r$ oder $l = r$.

Umgekehrt sagen wir, r kann auf l **reduziert** werden, wenn $l \overset{*}{\Rightarrow} r$.

Im Beispiel haben wir die erste Regel angewandt und dann mit der zweiten Regel das Zeichen + am Ende weggestrichen. Man sieht aber, daß die Anwendung der zweiten Regel alleine bereits das gewünschte Ergebnis liefert. Allgemein lauten die **Metaregeln**[36] zur Anwendung der Ersetzungen auf ein Wort x:

- Wenn $a \cdots b \to c \cdots d$ anwendbar ist, ersetze das Teilwort $a \cdots b$ von x durch $c \cdots d$;
- wenn $a \cdots b$ mehrfach vorkommt oder mehrere Regeln anwendbar sind, so wähle das Teilwort bzw. die Regel beliebig;
- wiederhole die Anwendung von Regeln beliebig oft.

Eine Menge $\mathcal{T} = \{p \to q\}$ von Regeln zusammen mit den vorstehenden Metaregeln heißt ein **Semi-Thue-System** oder **Textersetzungssystem**[37]. Die Menge aller r, die aus l abgeleitet werden, heißt die **formale Sprache** $L_l = L(\mathcal{T}, l)$ von l bei vorgegebenem Semi-Thue-System $\mathcal{T}$. Wenn $\mathcal{T} = \{p \to q\}$ ein Semi-Thue-System ist, so definiert auch die Menge $\mathcal{T}^{-1} = \{q \to p\}$, bei der wir die Pfeilrichtung umkehren, ein Semi-Thue-System. Das inverse System heißt ein **Reduktionssystem**, verglichen mit dem ursprünglichen **Ableitungssystem**[38].

Ein Semi-Thue-System zeigt die Grundform von **Algorithmen**: Es gibt endlich oder abzählbar unendlich viele Operationen $o_1, o_2, \ldots$; im Semi-Thue-System sind diese Operationen Ersetzungsregeln. Der Algorithmus wird ausgeführt, indem man endlich oft Operationen auf die Eingabe, in diesem Falle ein Wort x, anwendet. Zum Algorithmus gehört eine Endebedingung, die spezifiziert, wann die Anwendung von Operationen endet. Bei Semi-Thue-Systemen ist dies gemeinhin die Bedingung, daß keine Regel mehr anwendbar ist. Ein Algorithmus **terminiert** oder **hält**, sobald die Endebedingung erreicht ist. Gibt es eine Eingabe x, für die ein Algorithmus nicht hält, so heißt der Algorithmus (potentiell) nicht-terminierend[39].

36. das griechische Wort μετά bedeutet in solchen Zusammensetzungen meist *jenseits von* .... Metaregeln sind die übergeordneten Vorschriften zur Bildung oder Anwendung von Regeln.

37. engl. *string replacement system* oder *string rewrite system*.

38. Ein **Thue-System** ist ein symmetrisches Semi-Thue-System, $\mathcal{T} = \mathcal{T} \cup \mathcal{T}^{-1}$, bei dem zu jeder ableitenden Regel $p \to q$ auch die reduzierende Regel $q \to p$ zu $\mathcal{T}$ gehört. Thue-Systeme kommen in der Informatik seltener vor.

39. Die Vorstellung eines nicht-terminierenden Algorithmus ist in sich widersprüchlich, da nach Definition ein Algorithmus eine *endliche* Folge von Operationen spezifiziert und daher immer terminieren muß. Allerdings müssen wir oft erheblichen Aufwand investieren, um nachzuweisen,

Semi-Thue-Systeme sind im allgemeinen **indeterministische Algorithmen**: Zu einem Text x kann es mehrere anwendbare Regeln geben oder eine Regel kann auf verschiedene Teilwörter von x anwendbar sein. Ist hingegen in jedem Schritt die anzuwendende Operation eindeutig bestimmt, so heißt der Algorithmus **deterministisch**. Ein indeterministischer Algorithmus kann bei wiederholter Anwendung auf die gleiche Eingabe x unterschiedliche Ergebnisse liefern. Insbesondere kann ein indeterministischer Algorithmus, abhängig von dieser Auswahl, terminieren oder nicht terminieren.

Aufgabe 1.1: („Scholtens Kaffeedose", vgl. (Gries , 1989)) Gegeben sei eine Kaffeedose, die zwei Arten von Kaffeebohnen, wir nennen sie weiße und schwarze Bohnen, enthält. Ferner gebe es einen zusätzlichen Vorrat schwarzer Bohnen. Wir spielen das folgende **Kaffeedosenspiel**: Wir nehmen fortgesetzt blind zwei Bohnen aus der Dose; haben diese gleiche Farbe, so legen wir eine schwarze Bohne in die Dose zurück; haben sie verschiedene Farbe, so legen wir nur die weiße Bohne zurück.

1. Nehmen Sie zunächst an, daß die Bohnen in der Dose in irgendeiner Folge angeordnet seien, so daß ein Wort über dem Zeichenvorrat $\Sigma = \{\mathsf{weiß}, \mathsf{schwarz}\}$ vorliegt. Nehmen Sie immer zwei nebeneinander liegende Bohnen aus der Dose und ersetzen Sie sie wie angegeben. Formulieren Sie ein Semi-Thue-System für das Kaffeedosenspiel, das endet, wenn keine Regel mehr anwendbar ist.
2. Zeigen Sie, daß der Algorithmus immer terminiert.
3. Zeigen Sie, daß am Ende immer genau eine Bohne in der Dose ist. Was wissen Sie über die Farbe dieser Bohne? Zeigen Sie insbesondere, daß das Ergebnis unabhängig von der anfangs gewählten Anordnung der Bohnen ist.
4. Wir ändern die Spielregeln: Zwei schwarze Bohnen werden stets beide in die Dose zurückgelegt. Im Semi-Thue-System realisieren wir dies, indem wir die Regel $\mathsf{schwarz}\,\mathsf{schwarz} \to \cdots$ weglassen (Zwei schwarze Bohnen werden niemals zusammen entnommen). Was können Sie nun über die Terminierung des Algorithmus und die am Ende verbleibenden Bohnen beweisen? ♦

Überführt ein Semi-Thue-System $\mathcal{T}$ ein Wort x in $y = \mathcal{T}(x)$, $x \overset{*}{\Rightarrow} y$, und hält dann an, so nennen wir y auch eine **Normalform** von x. Es wäre schön, wenn wie beim Kaffeedosenspiel die Normalform immer eindeutig bestimmt ist. Das können wir bei allgemeinen Semi-Thue-Systemen gewöhnlich nicht erreichen.

daß ein Verfahren tatsächlich für alle oder zumindest für speziell vorgegebene Eingaben terminiert. Mit dem Begriff *(potentiell) nicht-terminierender Algorithmus* signalisieren wir also eigentlich, daß uns noch nicht klar ist, ob ein Algorithmus vorliegt oder nicht.

1.6.1 Markov-Algorithmen

Unabhängig von THUE erfand A. A. MARKOV[40] 1951 die nach ihm benannten Markov-Algorithmen zur Beschreibung von Textersetzungen (MARKOV , 1954). Ein **Markov-Algorithmus**, MARKOV sprach von **normalen Algorithmen**, ist ein deterministisches Semi-Thue-System mit endlich vielen Regeln und zwei verschiedenen Endebedingungen. Dazu führte MARKOV zusätzlich haltende Regeln $x \to . \; y$ ein, gekennzeichnet durch den Punkt nach dem Pfeil, und schrieb vor:

1. Wähle in jedem Schritt die erste anwendbare Regel. Falls sie auf mehrere Teilwörter anwendbar ist, wende sie auf das am weitesten links stehende Teilwort an.
2. Wende Regeln solange an, bis eine haltende Regel angewandt wurde, oder, bis keine Regel mehr anwendbar ist.

Hier bezieht sich *erste anwendbare Regel* auf die Reihenfolge, in der die Regeln angeschrieben wurden. Falls eine Regel $\varepsilon \to r$ angegeben ist, wird r am Anfang des Wortes eingesetzt, da das am weitesten links stehende leere Wort ersetzt wird. In der Anwendung wird das Anhalten des Algorithmus wegen fehlender anwendbarer Regel gewöhnlich als Fehlerhalt interpretiert.

Erlaubt man zusätzliche Zeichen $\alpha, \beta, \gamma, \ldots$, sogenannte *Schiffchen*, die weder im Eingabetext noch im Ergebnis vorkommen, so kann man mit diesen **gesteuerten Markov-Algorithmen**, wie übrigens auch mit allgemeinen Semi-Thue-Systemen, jede beliebige Berechnung beschreiben, die algorithmisch formulierbar ist.

Beispiel 1.3: Wir betrachten die Regeln

$$\begin{array}{rclrcl} \alpha\mathsf{L} & \to & \mathsf{L}\alpha, & \alpha\mathsf{O} & \to & \mathsf{O}\alpha, \\ \alpha & \to & \beta, & \mathsf{L}\beta & \to & \beta\mathsf{O}, \\ \mathsf{O}\beta & \to . & \mathsf{L}, & \beta & \to . & \mathsf{L}, \\ \varepsilon & \to & \alpha & & & \end{array} \tag{1.3}$$

über dem Zeichenvorrat $\Sigma = \{\mathsf{O}, \mathsf{L}\}$ mit den Schiffchen α und β. Für das Wort LOLL liefert der Algorithmus ($\Rightarrow$ hat die gleiche Bedeutung wie in Abschnitt 1.6):

$$\begin{array}{lllllllll} \mathsf{LOLL} & \Rightarrow & \alpha\mathsf{LOLL} & \Rightarrow & \mathsf{L}\alpha\mathsf{OLL} & \Rightarrow & \mathsf{LO}\alpha\mathsf{LL} & \Rightarrow & \mathsf{LOL}\alpha\mathsf{L} \\ & \Rightarrow & \mathsf{LOLL}\alpha & \Rightarrow & \mathsf{LOLL}\beta & \Rightarrow & \mathsf{LOL}\beta\mathsf{O} & \Rightarrow & \mathsf{LO}\beta\mathsf{OO} \\ & \Rightarrow & \mathsf{LLOO} & & & & & & \end{array}$$

♦

Aufgabe 1.2: Zeigen Sie, daß in einem Markov-Algorithmus Regeln, die auf $\varepsilon \to r$ folgen, nie angewandt werden.

40. ANDREI A. MARKOV, 1903 – 1979, russischer Mathematiker. Markov-Ketten und -Prozesse in der Stochastik gehen auf seinen gleichnamigen Vater, 1856 – 1922, zurück.

1.6.2 Formale Systeme

Wir nennen $(U, \Rightarrow)$ ein **formales System**, wenn

- U eine endliche oder abzählbar unendliche Menge,
- $\Rightarrow \subseteq U \times U$ eine Relation ist,
- es einen Algorithmus gibt, der jedes $r \in U$ mit $l \Rightarrow r$, $l \in U$, in endlich vielen Schritten aus l berechnen kann.

r heißt dann aus l (**effektiv**) **berechenbar**. U könnte eine Menge von Wörtern und $\Rightarrow$ die Ableitungsrelation eines Semi-Thue-Systems oder eines Markov-Algorithmus sein. Wir könnten aber auch die Addition $(m, n) \Rightarrow (m + n, 0)$ auf der Grundmenge $U = \mathbb{Z} \times \mathbb{Z}$ der Paare ganzer Zahlen definieren. Hingegen können wir die Addition reeller Zahlen nicht als formales System beschreiben; einerseits ist die Menge der reellen Zahlen überabzählbar, andererseits terminieren Algorithmen zur Addition reller Zahlen nicht nach endlich vielen Schritten.

Aus Abschnitt 1.6 übertragen wir die Schreibweise $l \stackrel{*}{\Rightarrow} r$ auf formale Systeme. $(U, \stackrel{*}{\Rightarrow})$ ist selbst ein formales System. Allerdings können wir nur positiv feststellen, ob $l \stackrel{*}{\Rightarrow} r$ gilt. Haben wir ein r mit $l \not\stackrel{*}{\Rightarrow} r$ vorgegeben, so könnten wir uns nach 1000, 10 000, 100 000, ... -facher Regelanwendung immer noch Hoffnungen machen, daß sich r ergibt: der Algorithmus terminiert möglicherweise nicht. Ein formales System heißt **entscheidbar**, wenn für beliebige $l, r \in U$ effektiv festgestellt werden kann, ob $l \stackrel{*}{\Rightarrow} r$ gilt oder nicht.

Aufgabe 1.3: Die Paare (l, r) mit $l \not\stackrel{*}{\Rightarrow} r$ definieren die Relation $\not\stackrel{*}{\Rightarrow}$. Dabei ist $\not\stackrel{*}{\Rightarrow} = U \times U \setminus \stackrel{*}{\Rightarrow} = \complement \stackrel{*}{\Rightarrow}$. Zeigen Sie: Ein formales System $(U, \Rightarrow)$ ist genau dann entscheidbar, wenn auch $(U, \not\stackrel{*}{\Rightarrow})$ ein (entscheidbares) formales System ist.

Formale Systeme beschreiben die allgemeinste Form von Relationen zwischen Gegenständen, die wir mit algorithmischen Methoden modellieren und realisieren können. Wir werden an zahlreichen Beispielen sehen, daß die Grundmenge U oft eine weitergehende Struktur aufweist, die in der Ableitungsbeziehung genutzt wird.

Ein formales System, z. B. ein Markov-Algorithmus, heißt **endlich erzeugt**, wenn die Relation $l \Rightarrow r$ durch eine endliche Menge von Regeln $p \to q$ definiert wird. Allerdings bestimmen diese Regeln das formale System nicht allein: Die Metaregeln, wie die Regeln angewandt werden, gehören ebenfalls zum formalen System, wie wir am Unterschied zwischen allgemeinen Semi-Thue-Systemen und Markov-Algorithmen sehen. Ein formales System $(U, \Rightarrow)$ heißt ein **Kalkül**[41], wenn die Relation $\Rightarrow$ durch eine endliche Menge $\{p \to q\}$ von (Grund-)Regeln zusammen mit einer endlichen Menge von **Meta-** oder **Kalkülregeln** definiert ist[42]. Ein Semi-Thue-System ist also ein Kalkül.

41. lat. *calculus*, glattes Steinchen, Rechenstein.

42. In der Logik spricht man statt von einem Kalkül auch von einer **konstruktiven Theorie**. Eine **nicht-konstruktive Theorie** bezeichnet ein System $(U, \Rightarrow)$, in dem die Relation $\Rightarrow$ nicht effektiv

1.6.3 Chomsky-Grammatiken

In den 50er Jahren untersuchten N. CHOMSKY[43] und andere Linguisten Semi-Thue-Systeme mit dem Ziel, die Struktur von Sätzen in natürlicher Sprache als **Ableitungsbaum** wie in Abb. 1.5 darzustellen. Hier ist *Satz* die Normalform für

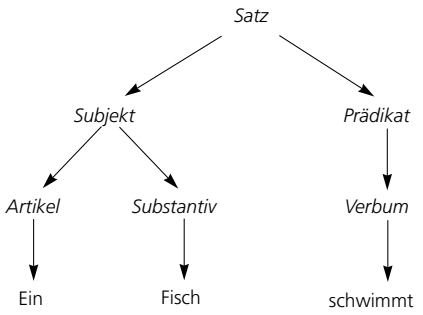

Abbildung 1.5: Schema des Ableitungsbaums eines Satzes

die Zeichenreihe Ein Fisch schwimmt. Es werden Regeln wie

$$\textit{Satz} \rightarrow \textit{Subjekt Prädikat} \tag{1.4}$$

oder

$$\textit{Substantiv} \rightarrow \text{Fisch} \tag{1.5}$$

benutzt. CHOMSKY nannte diese Semi-Thue-Systeme **Grammatiken** und ihre Regeln **Produktionen**.

In einer Grammatik unterscheidet man syntaktische Begriffe wie *Satz, Verbum* usw. von den Wörtern der zu beschreibenden Sprache. Letztere betrachtet man als Einzelzeichen eines Zeichenvorrats Σ ohne weitere innere Struktur und bezeichnet sie als **terminale Zeichen** oder kurz als **Terminale**. Die syntaktischen Begriffe bilden den Zeichenvorrat N der **syntaktischen Variablen**, **nichtterminalen Zeichen** oder **Nichtterminale**[44].

Mit der Angabe einer Grammatik G verfolgt man das Ziel, die terminalen Zeichenreihen x, $x \in T^*$, zu beschreiben, die einem ausgezeichneten syntaktischen Begriff Z, dem **Startsymbol**, **Axiom** oder **Ziel** der Grammatik G, entsprechen. Die Menge $L(G)$ dieser Zeichenreihen heißt der **Sprachschatz** der Grammatik G. Das Ziel Z wird im Unterschied zu Abschnitt 1.6, wo wir die Menge aller, also nicht nur der terminalen, aus Z ableitbaren Zeichenreihen als formale Sprache $L_Z = L(G, Z)$ bezeichneten, nicht mehr explizit erwähnt, da es durch die Grammatik eindeutig gegeben ist.

berechenbar ist, bzw. in dem nicht bekannt ist, ob $\Rightarrow$ berechenbar ist.

43. NOAM A. CHOMSKY, geb. 1928, amerikanischer Linguist.

44. engl. *terminal* bzw. *nonterminal.*

Die Vereinigung $V = N \cup \Sigma$ heißt das **Vokabular** der Grammatik bzw. der formalen Sprache. Die Menge *aller* Zeichenreihen über den Zeichenvorräten V bzw. Σ bezeichnen wir im folgenden mit V^* bzw. Σ^*; auch die leere Zeichenreihe ε ist zulässig. Produktionen sind Regeln $l \to r$ mit Zeichenreihen l, r aus V^*. Eine Zeichenreihe x aus V^*, die durch endlich viele Anwendungen von Produktionen aus dem Ziel Z abgeleitet werden kann, in Zeichen $Z \stackrel{*}{\Rightarrow} x$, heißt eine **Satzform** oder **Phrase**[45]. Die Sprache $L(G)$ besteht also aus den terminalen Phrasen.

Wir können die Phrasenstruktur sichtbar machen, indem wir alle Produktionen $l \to r$ durch $l \to \langle r \rangle$ ersetzen. Die beiden Produktionenmengen $P = \{Z \to z,\ Z \to zZz\}$ und $P' = \{Z \to z,\ Z \to Zzz\}$ strukturieren Wörter wie *zzzzz* unterschiedlich: $\langle z \langle z \langle z \rangle z \rangle z \rangle$ bzw. $\langle\langle z \rangle z z \rangle z z \rangle$. G und G' beschreiben aber beide die Menge aller Wörter, die aus einer ungeraden Anzahl von *z*'s bestehen.

Zwei Grammatiken G, G' heißen **schwach äquivalent**, wenn $L(G) = L(G')$. Zwei schwach äquivalente Grammatiken heißen **strukturäquivalent**, wenn es eine dritte schwach äquivalente Grammatik G'' gibt, und sich die Phrasenstruktur für G, G' bei allen Wörtern x aus dem Sprachschatz nur durch Zufügen von Klammerpaaren $\langle \cdots \rangle$ von der Phrasenstruktur für G'' unterscheidet. Wie bei natürlichen Sprachen sind Phrasen die bedeutungstragenden Strukturelemente eines Wortes x. Zwei strukturäquivalente Grammatiken können als Alternativen zur Beschreibung des gleichen semantischen Sachverhalts dienen, wenn nur die Phrasen für G'' bedeutungstragend sind; schwache Äquivalenz genügt nicht.

Im Sinne von Abschnitt 1.6 ist die Normalform einer Phrase x das Ziel Z. Während wir dort die Normalform n aus x herleiten, $x \stackrel{*}{\Rightarrow} n$, schreiben wir jetzt $Z \stackrel{*}{\Rightarrow} x$: Bei Grammatiken wird eine Phrase aus ihrer Normalform abgeleitet. Um herauszufinden, ob eine Zeichenreihe x eine Phrase ist, müssen wir ihre syntaktische Struktur feststellen. Dieser Vorgang heißt **Zerteilung**[46] von x. Durch Umkehr aller Pfeile erhalten wir aus einem Ableitungssystem A ein **Reduktions-** oder **Zerteilungssystem** R und umgekehrt. Ein Ableitungsbaum wie in Abb. 1.5 beschreibt sowohl, wie man den Satz Ein Fisch schwimmt aus dem Ziel *Satz* ableitet, als auch wie man ihn auf das Ziel reduziert.

Eine Grammatik $G = (\Sigma, N, P, Z)$, in der Σ, N und Z die vorstehende Bedeutung haben und P eine endliche Menge von Produktionen $l \to r$ ist, heißt eine **Chomsky-Grammatik**.

Chomsky-Grammatiken lassen sich nach der Form ihrer Produktionen $l \to r$ wie in Tab. 1.1 weiter klassifizieren.

Eine Grammatik ist eine **Chomsky-Typ 0** oder kurz eine **CH-0-Grammatik**,

45. engl. *sentential form* bzw. *phrase*. Grammatiken nach Chomsky heißen oft auch **Phrasenstrukturgrammatiken**.

46. engl. *parsing*.

Tabelle 1.1: Typen von Produktionen

Produktion	Produktions-typ	Eigenschaften	Gr.-typ
$l \to r$	allgemein	$l, r \in V^*$ beliebig	CH-0
$l \to \varepsilon$	ε-Produktion	$l \in V^*, r = \epsilon$	
$l \to r$	beschränkt	$l, r \in V^*, 1 \leqslant \|l\| \leqslant \|r\|$	CH-1
$uAv \to urv$	kontextsensitiv	$A \in N, u, v, r \in V^*, r \neq \varepsilon$	CH-1
$A \to r$	kontextfrei	$A \in N, r \in V^*$	CH-2
$A \to Bx$	linkslinear	$A, B \in N, x \in \Sigma$	CH-3
$A \to xB$	rechtslinear		CH-3
$A \to x$	terminierend	$A \in N, x \in \Sigma$	

wenn ihre Produktionen keinen Einschränkungen unterliegen. Insbesondere sind Produktionen $\varepsilon \to r$ erlaubt. Der Vergleich mit Markov-Algorithmen — die Schiffchen entsprechen in etwa den Nichtterminalen — zeigt, daß man jede berechenbare Menge als Sprache $L(G)$ einer Chomsky-0-Grammatik erhalten kann.

Eine Grammatik ist **kontextsensitiv**, vom **Chomsky-Typ 1** oder eine **CH-1-Grammatik**, wenn ihre Produktionen beschränkt oder kontextsensitiv sind. Da in einer Ableitung $Z \overset{*}{\Rightarrow} x \Rightarrow y$ stets $|x| \leqslant |y|$ gilt, kann man in endlich vielen Schritten bestimmen, ob ein Wort y vorgegebener Länge zu $L(G)$ gehört oder nicht; man muß nur die endlich vielen Wörter der Länge $|y|$ finden, die zu $L(G)$ gehören. Daher muß die Sprache $L(G)$ einer kontextsensitiven Grammatik entscheidbar sein.

Eine Grammatik ist **kontextfrei**, vom **Chomsky-Typ 2** oder eine **CH-2-Grammatik**, wenn ihre Produktionen kontextfrei sind. Eine kontextfreie Grammatik heißt **ε-frei**, wenn sie keine ε-Produktion enthält. Wir werden später sehen, daß ε-Produktionen nur benötigt werden, wenn $\varepsilon \in L(G)$. ε-freie kontextfreie Grammatiken leisten weniger als kontextsensitive Grammatiken: Die Struktur korrekter Programme in Programmiersprachen wie PASCAL, C, SATHER oder JAVA läßt sich mit einer kontextfreien Grammatik beschreiben; hingegen benötigen wir eine kontextsensitive Grammatik, wenn wir zusätzlich die Zuordnung von Bezeichnern zu Deklarationen prüfen wollen.

Eine Grammatik ist **regulär**, vom **Chomsky-Typ 3** oder eine **CH-3-Grammatik**, wenn sie neben terminierenden und ε-Produktionen entweder nur links- oder nur rechtslineare Produktionen enthält. Reguläre Grammatiken leisten weniger als ε-freie kontextfreie Grammatiken: Mit letzteren kann man feststellen, ob eine Formel wie $((a * (b + c)))$ korrekt geklammert ist. Mit einer regulären Grammatik kann man sich nur eine beschränkte Anzahl noch zu schließender Klammern merken. Wir werden noch sehen, daß linkslineare und rechtslineare

Produktionen gleich leistungsfähig sind. Mischt man beide Arten von Produktionen, so erhält man eine **lineare Grammatik**, die das Problem der korrekten Klammerung lösen kann.

Sei $G = (\Sigma, N, P, Z)$ eine beliebige Chomsky-Grammatik, die keine Produktion $\varepsilon \to r$ enthält. G heißt **separiert**, wenn auf der linken Seite der Produktionen nur Nichtterminale vorkommen. Wir können zu G stets eine strukturäquivalente, separierte Grammatik $G' = (\Sigma, N', P', Z)$ angeben. Dazu führen wir für jedes Zeichen $a \in \Sigma$ ein neues, bisher nicht benutztes Nichtterminal A ein. Ist N'' die Menge aller solchen neuen Nichtterminale, so setzen wir $N' = N \cup N''$. In den Produktionen $l \to r$ aus P ersetzen wir überall a durch A und erweitern die so modifizierte Produktionenmenge zu P', indem wir für alle $a \in \Sigma$ noch die entsprechende Produktion $A \to a$ hinzufügen.

Eine beschränkte Produktion $AB \to x_1 \cdots x_k$, $k \geqslant 2$, einer Grammatik G läßt sich unter Einführung neuer Nichtterminale A', B' strukturäquivalent durch kontextsensitive Produktionen ersetzen. Dazu setzen wir zunächst voraus, daß G separiert sei, also $A, B \in N$. Dann leisten die Produktionen $AB \to A'B$, $A'B \to A'B'$, $A'B' \to A'x_2 \cdots x_k$, $A' \to x_1$ das Verlangte. Durch vollständige Induktion sieht man, daß dieses Verfahren auf Produktionen mit linken Seiten beliebiger Länge $\geqslant 2$ verallgemeinert werden kann. Wir können also zu jeder CH-1-Grammatik mit beschränkten Produktionen eine strukturäquivalente Grammatik mit kontextsensitiven Produktionen angeben. Trivialerweise gilt auch die Umkehrung, da jede kontextsensitive Produktion beschränkt ist.

In der praktischen Anwendung sind vor allem kontextfreie und reguläre Grammatiken von Bedeutung. Nur für diese kann die Phrasenstruktur eines Wortes durch einen Ableitungsbaum wie in Abb. 1.5 vollständig angegeben werden. Wir beschränken uns nachfolgend auf diese Typen.

Wir spezifizieren solche Grammatiken meist nur durch Angabe der Produktionenmenge P und benutzen dabei folgende Konventionen:

- Die linke Seite der ersten Produktion bezeichnet das Ziel der Grammatik.
- Alle Zeichen, die nur auf der rechten Seite von Produktionen vorkommen, gehören zum terminalen Zeichenvorrat Σ.
- Alle nichtterminalen Zeichen kommen wenigstens einmal auf der linken Seite einer Produktion vor.
- Eine Menge $A \to r_1, A \to r_2, \ldots$ mit gleicher linker Seite A kann zusammengefaßt werden zu $A \to r_1 \mid r_2 \mid \ldots$ Der senkrechte Strich steht für *oder*.

Beispiel 1.4: Gegeben sei die kontextfreie Grammatik $G_A = (\Sigma, N, P, A)$ mit

$$P = \{ \quad A \to T \mid A + T, \tag{1.6}$$

$$T \to F \mid T * F, \tag{1.7}$$

$$F \to \mathsf{bez} \mid (A)\} \tag{1.8}$$

Wenn bez beliebige einfache Operanden (Bezeichner, Konstante, usw.) bezeichnet, beschreibt G_A den Aufbau arithmetischer Ausdrücke mit den Operatoren + und $*$. Eine Formel wie $(a + b) * c + d$ kann aus dem Zielsymbol A wie folgt abgeleitet werden:

$$\begin{aligned} A &\Rightarrow A + T \Rightarrow T + T \Rightarrow T * F + T \\ &\Rightarrow F * F + T \Rightarrow (A) * F + T \Rightarrow (A + T) * F + T \\ &\Rightarrow (T + T) * F + T \Rightarrow (F + T) * F + T \Rightarrow (\text{bez} + T) * F + T \\ &\Rightarrow (\text{bez} + F) * F + T \Rightarrow (\text{bez} + \text{bez}) * F + T \\ &\Rightarrow (\text{bez} + \text{bez}) * \text{bez} + T \Rightarrow (\text{bez} + \text{bez}) * \text{bez} + F \\ &\Rightarrow (\text{bez} + \text{bez}) * \text{bez} + \text{bez}. \end{aligned}$$

Die Abb. 1.6 zeigt den zugehörigen Ableitungsbaum. Die grauen Umrandungen, die nicht zum Ableitungsbaum gehören, umschließen die Anwendung bestimmter Produktionen. ♦

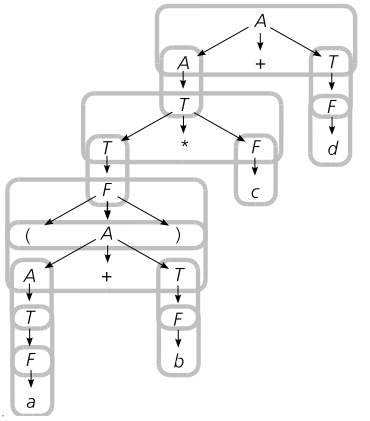

Abbildung 1.6: Ableitungsbaum zu $A \overset{*}{\Rightarrow} (a + b) * c + d$

Umgekehrt kann man $(a + b) * c + d$ mit Hilfe von P auf A reduzieren. Dabei müssen wir feststellen, daß manche Reduktionen willkürlich erscheinen. Wir könnten z. B. die Formel auch auf $(A + A) * A + A$ reduzieren. Dies ist allerdings eine **Sackgasse**: Die Reduktion läßt sich nicht weiter fortsetzen, die Formel scheint nicht aus A ableitbar zu sein.

Dies ist eine Folge des Indeterminismus bei der Anwendung von Produktionen. Später werden wir Verfahren kennenlernen, wie man den Ersetzungsprozeß in praktisch interessanten Fällen deterministisch gestalten kann.

Eine Produktion der Form $A \to B, A, B \in N$ heißt eine **Kettenproduktion**. Das vorige Beispiel enthält die Kettenproduktionen $A \to T$, $T \to F$, die

nur dazu dienen, die Vorrangregeln zwischen den arithmetischen Operatoren sicherzustellen. Kettenproduktionen haben meist keine semantische Bedeutung; sie sollten weitgehend vermieden werden. Eine kontextfreie Grammatik heißt **anständig**[47], wenn sie keine ε- und keine Kettenproduktionen aufweist. Eine kontextfreie Grammatik $G = (\Sigma, N, P, Z)$ heißt **reduziert**, wenn das Ziel Z der Grammatik nirgends auf der rechten Seite von Produktionen erscheint und wenn es zu jedem $A \in N$ Zeichenreihen $u, v, w \in \Sigma^*$ so gibt, daß $Z \overset{*}{\Rightarrow} uAv \overset{*}{\Rightarrow} uwv$. (Es gilt dann $uwv \in L(G)$.)

Beispiel 1.5: Die Grammatik aus Beispiel 1.4 kann vereinfacht werden zu

$$P = \{A \to A + A \mid A * A \mid (A) \mid \mathsf{bez}\}. \tag{1.9}$$

Die Ableitung von $(a + b) * c + d$ lautet dann

$$\begin{aligned} A &\Rightarrow A + A \Rightarrow A * A + A \Rightarrow (A) * A + A \\ &\Rightarrow (A + A) * A + A \Rightarrow (\mathsf{bez} + A) * A + A \Rightarrow (\mathsf{bez} + \mathsf{bez}) * A + A \\ &\Rightarrow (\mathsf{bez} + \mathsf{bez}) * \mathsf{bez} + A \Rightarrow (\mathsf{bez} + \mathsf{bez}) * \mathsf{bez} + \mathsf{bez}. \end{aligned}$$

Diese Ableitung entspricht dem **Kantorowitsch-Baum**[48] der Abb. 1.7. Er gibt die Ableitung eines arithmetischen Ausdrucks auf das Wesentliche beschränkt wieder: Statt des Nichtterminals tragen wir das Operatorzeichen ein; da es nur eine Produktion gibt, in der der Operator vorkommt, ergibt sich diese von selbst. Ebenso genügt die Angabe von bez statt der Produktion $A \to \mathsf{bez}$. Die notwendige Klammersetzung ergibt sich aus der Struktur des Baumes und den Vorrangregeln der Operatoren.

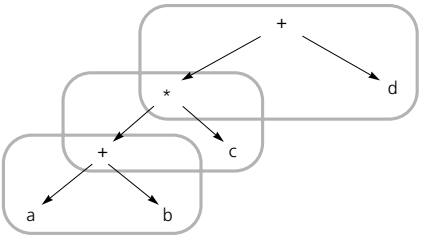

Abbildung 1.7: Kantorowitsch-Baum zu $A \overset{*}{\Rightarrow} (a + b) * c + d$

Die Produktionen (1.9) lassen für $a + b * c$ auch die Reduktion $A \Rightarrow A * A \Rightarrow A + A * A \overset{*}{\Rightarrow} \mathsf{bez} + \mathsf{bez} * \mathsf{bez}$ zu, die eigentlich zu $(a + b) * c$ gehört. Zu $a + b * c$

47. engl. *proper*.

48. Leonid Witaljewitsch Kantorowitsch, 1912–1986, russischer Mathematiker und Wirtschaftswissenschaftler, Nobelpreisträger 1975, entwickelte u.a. die Grundlagen des linearen Programmierens. Er führte diese Bäume 1956 ein.

gehören also formal zwei verschiedene KANTOROWITSCH-Bäume, von denen aber nur einer die „richtige“ Struktur wiedergibt. ♦

Aufgabe 1.4: Zeichnen Sie die beiden KANTOROWITSCH-Bäume zu $a + b * c$.

Allgemein heißt eine kontextfreie Grammatik G **mehrdeutig**, wenn es zu einem Satz der Sprache $L(G)$ mehr als einen Ableitungsbaum gibt, andernfalls heißt sie **eindeutig**. Der Vergleich der Beispiele 1.4 und 1.5 zeigt, daß es zur gleichen Sprache L mehr als eine Grammatik geben kann, von denen eine mehrdeutig, die andere eindeutig ist. Dies ist allerdings ein Spezialfall: Es gibt auch **inhärent mehrdeutige Sprachen** L, für die keine einzige eindeutige Grammatik existiert.

Verwendet man kontextfreie Sprachen zur Beschreibung der Syntax von Programmiersprachen, so werden die Nichtterminale gewöhnlich durch Wörter in natürlicher Sprache wiedergegeben. Hierzu verwendet man folgende Konventionen:

- Statt $\rightarrow$ schreibt man ::=
- Wörter, die als Nichtterminale benutzt werden, werden in $\langle\ \rangle$ eingeschlossen.
- Der senkrechte Strich |, gelesen *oder*, trennt rechte Seiten zur gleichen linken Seite von Produktionen.

Die Produktionen (1.9) lauten in dieser Schreibweise

⟨Ausdruck⟩ ::= ⟨Ausdruck⟩ + ⟨Ausdruck⟩ |
⟨Ausdruck⟩ * ⟨Ausdruck⟩ |
(⟨Ausdruck⟩) | bez

Die Notation wurde von JOHN BACKUS 1959 erfunden und von PETER NAUR erstmals zur Beschreibung von ALGOL 60 eingesetzt. Sie ist daher als **Backus-Naur-Form**, abgekürzt **BNF**, bekannt[49].

Zur rechnergestützten Verarbeitung ist es häufig günstiger, die Nichtterminale als normale Wörter zu schreiben und stattdessen die terminalen Zeichen zu klammern, z. B., indem man sie in Apostrophzeichen einschließt. Diese Konvention ermöglicht den Gebrauch von runden und eckigen Klammern sowie von $*$ und + als Metazeichen mit der Bedeutung

- $[\cdots]$ bezeichnet optionale Teile einer rechten Seite.
- $(\cdots)$ umschließt eine Gruppe von Zeichen.
- | in einer runden oder eckigen Klammer trennt alternative Klammerinhalte.
- Ein Stern nach einem Zeichen oder einer runden Klammer bezeichnet n-fache Wiederholung des Zeichens oder der Gruppe, $n = 0, 1, \ldots$.

49. PETER INGERMAN, (INGERMAN , 1967), wies darauf hin, daß der Inder PĀNINI bereits zwischen 400 und 200 v. Chr. zur BNF vergleichbare Schreibweisen zur Beschreibung einer Sanskrit-Grammatik einsetzte.

- Ein Pluszeichen nach einem Zeichen oder einer runden Klammer bezeichnet n-fache Wiederholung des Zeichens oder der Gruppe, $n = 1, 2, \ldots$.

In dieser **erweiterten Backus-Naur-Form** oder **EBNF** lauten die Produktionen (1.6) - (1.8) erweitert um das unäre + und – sowie die Subtraktion

Ausdruck ::= ['+' | '–'] Term (('+' | '–') Term)*
Term ::= Faktor ('∗' Faktor)*
Faktor ::= '(' Ausdruck ')' | 'bez'

Die Regeln der EBNF sind nicht verbindlich festgelegt, z. B. verwenden manche Autoren die Schreibweise $\{\cdots\}$ statt $(\cdots)^*$.

Statt EBNF kann man die Regeln auch graphisch durch **Syntaxdiagramme** wie in Abb. 1.8 wiedergeben. Ein Syntaxdiagramm hat genau einen Eingang und einen Ausgang. Jeder Weg vom Eingang zum Ausgang ergibt ein zulässiges Wort für den syntaktischen Begriff, der das Diagramm benennt. Ovale oder Kreise umschließen terminale Zeichen. Kästchen umschließen syntaktische Begriffe; für sie muß ein zulässiges Wort zu einem solchen Begriff eingesetzt werden. Wie man sieht, können Syntaxdiagramme auch rekursiv sein. Syntaxdiagramme benötigen oft wesentlich weniger syntaktische Begriffe als die entsprechende kontextfreie Grammatik.

Ausdruck

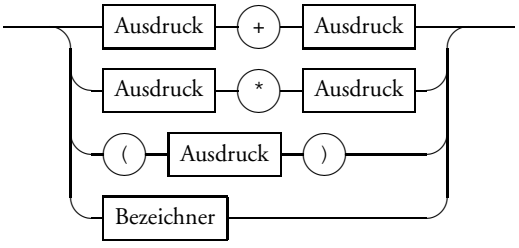

Abbildung 1.8: Syntaxdiagramm für Ausdrücke

1.7 Anmerkungen und Verweise

Wir sehen die *reale Welt* als Wirklichkeit an und Modelle als abstrakte Darstellungen davon. In der *Zwei-Welten-Lehre* des griechischen Philosophen Platon sind dagegen die Ideen die Wirklichkeit, während die reale Welt das Modell ist. Diese Auffassung wird uns in der Aussagen- und Prädikatenlogik wiederbegegnen: Man achte also jeweils darauf, was als Modell bezeichnet wird.

Für die Geschichte der Entwicklung von Rechnern vergleiche man etwa (Goldscheider und Zemanek , 1971) und (Randell , 1982). Weiterführendes über Rechenanlagen findet man in (Giloi , 1997).

Die präzise Definition des in Abschnitt 1.4 informell eingeführten Begriffs *Algorithmus* hat Mathematiker und Logiker in der ersten Hälfte des zwanzigsten Jahrhunderts sehr intensiv beschäftigt. Man fand eine Reihe unterschiedlicher Definitionen des Begriffs *algorithmisch berechenbar*. 1936 formulierte CHURCH[50], die nach ihm benannte **Churchsche These** (CHURCH , 1936): Alle Begriffsbestimmungen von algorithmisch berechenbar (einschl. der damals oder heute noch nicht bekannten) sind gleichwertig und definieren einheitlich die Begriffe **berechenbare Funktion** und **berechenbare Menge**. In jedem Fall gibt es endlich viele verschiedene Operationen (Regeln in Semi-Thue-Systemen oder Markov-Algorithmen) und eine Funktion $f(x)$ heißt für beliebige x aus dem Definitonsbereich von f berechenbar, wenn der Funktionswert durch endlich viele Anwendungen von Operationen entsprechend etwaigen Metaregeln aus x bestimmt werden kann; oft beschränkt man sich auf die Menge $\mathbb{N}$ der natürlichen Zahlen als Definitionsbereich von f. Eine Menge M heißt berechenbar, wenn es eine berechenbare surjektive Abbildung $f: \mathbb{N} \twoheadrightarrow M$ der natürlichen Zahlen (oder einer endlichen Teilmenge davon) auf M gibt. Insbesondere ist dann berechenbar, ob $x \in M$, d. h., ob es ein $n \in \mathbb{N}$ gibt mit $f(n) = x$.

Bisher konnte man alle Begriffsbestimmungen von Berechenbarkeit, darunter allgemeine Semi-Thue-Systeme, Markov-Algorithmen und Chomsky-0-Grammatiken, als äquivalent nachweisen. Reine Semi-Thue-Systeme ohne Hilfszeichen sind schwächer: Die Menge $\{a^{2^n} \mid n \geqslant 0\}$ ist berechenbar, aber nicht durch ein reines Semi-Thue-System erzeugbar, vgl. (SALOMAA , 1978).

Eine berechenbare Funktion f ist meist nur *partiell* (vgl. S. 354) definiert: Ein Markov-Algorithmus A zur Berechnung von f könnte mit dem Ergebnis L enden, falls die eingegebene Zeichenreihe zu einer bestimmten Menge M gehört. Daraus folgt nicht, daß A auch für Zeichenreihen, die nicht zu M gehören, terminiert; er könnte für $x \notin M$ auch endlos laufen. Wir bekommen also eine Antwort im Erfolgsfall, aber keine Antwort bei Mißerfolg. Trotzdem heißt nicht nur die Funktion f, sondern auch die Menge M berechenbar. Falls unser Algorithmus für beliebige Zeichenreihen (mit unterschiedlichem Ergebnis) terminiert, heißt die Funktion f, wie wir in Unterabschnitt 1.6.2 erläuterten, **entscheidbar**.

Normen legen gemeinsame Eigenschaften der Produkte verschiedener Hersteller fest, damit diese „die gleiche Sprache sprechen". Beispiele sind die Aufzeichnungsdichte für Diskettenlaufwerk und Disketten oder ein gemeinsamer Zeichencode für Programme. Normen müssen oftmals bestimmt werden, bevor überhaupt Produkte konstruiert und auf den Markt gebracht werden. Die gemeinsame Festlegung von Spezifikationen durch Auftraggeber und Hersteller setzt ebenfalls voraus, daß diese die gleiche Sprache sprechen. Andernfalls kommt es zu Mißverständnissen, die zur Unbrauchbarkeit des Ergebnisses füh-

50. ALONZO CHURCH, 1903 – 1995, amerikanischer Mathematiker und Logiker.

ren können. Während in der Mathematik die gemeinsame Terminologie durch Lehrbücher erreicht wird, erweist sich dies in der Informatik als ein zu langsamer Prozeß. Daher wird auch die Terminologie in der praktischen und technischen Informatik zunehmend durch Normen oder *de facto* Standards geregelt. Der Informatiker muß solche Produkt- und Terminologie-Normen zur Kenntnis nehmen und an ihrer Weiterentwicklung mitarbeiten.

http://www.springer.com/978-3-540-24405-9

Vorlesungen über Informatik
Band 1: Grundlagen und funktionales Programmieren
Goos, G.; Zimmermann, W.
2006, XIII, 400 S., Softcover
ISBN: 978-3-540-24405-9