programm & prozess.grundlagen der programmierung von bildern
digitale medien | hfk bremen | winter 2006/07 | 4 sws 6 ects
dienstag 14 bis 18 uhr | speicher XI 2.10.080 | modul 301-1
vak (uni) 03-06-G-515.53

frieder nake | 218 3525 | 9595 1267 | nake@informatik.uni-bremen.de

24 Dezember 2006

Etwas zu:

Funktion Algorithmus Programm und so

Zu Anfang des Kurses hatte es die Frage gegeben „Was ist ein Algorithmus?" Jeder Anwesende sollte kurz nachdenken und dann das zu Papier bringen, was ihn oder ihr in der damaligen Situation zu diesem Stichwort in den Sinn käme. Seitdem haben wir das Thema und etliche mit ihm verwandte mehrfach hin und her gewendet. Für eine Einführung in die begrifflichen und praktischen Grundlagen der Informatik ist es unabdingbar, ein gutes Verständnis zu Begriffen wie Algorithmus, Programm, Programmiersprache, Daten und Berechenbarkeit und deren Umfeld reifen zu lassen. Deswegen ist es auch jetzt noch angebracht, die damals gesammelten Antworten zu notieren. Sie werden ergänzt durch einige Erklärungen, Erläuterungen und gar Definitionen, über die wir im Verlauf des Kurses bisher sprechen konnten.

1. Studentische Notate zum Stichwort „Algorithmus"

Den Teilnehmenden war vorgegeben worden, die Phrase „Ein Algorithmus ist …" zu einem Satz oder auch mehr zu ergänzen. Sie sollten dazu in kleinen Gruppen zu zweit oder zu dritt diskutieren und die Ergebnisse auf dem ausgeteilten Blatt Papier notieren. In einer beliebigen Reihenfolge sind hier die Antworten zusammengestellt, in leicht redigierter Form und, wo das notwendig erschien, mit einer Erläuterung in eckigen Klammern [...].

Ein Algorithmus ist …

- eine schrittweise Lösung eines Problems mittels einer endlichen Anzahl von Regeln. Algorithmen sind Verfahren, mit denen sich Probleme verschiedenster Art lösen lassen. Beispiel: Sortieren von CDs/DVDs in einem Regal.
- Alcharesmi – arabisches Wort. Folge von mechanisch ausführbaren Anweisungen. Christoph sagt: Mechanisch ausführbare Folge von maschinenlesbaren / -ausführbaren Anweisungen mit endlich vielen Schritten. Ein Prozess Ist eine Instantiierung von Algorithmen. Funktion der mathematische Spezialfall
- beschreibt einen zu beschreitenden (vorher definierten) Weg, welcher zur Erreichung eines Ziels zu gehen ist.
- (gut □schlecht; lang □kurz) Unter einem Algorithmus versteht man allgemein eine genau definierte Handlungsvorschrift zur Lösung eines Problems oder einer bestimmten Art von Problemen (Wiki). „Anleitung" – „exakt beschriebene Vorgehensweise mit endlich vielen eindeutig beschriebenen Schritten".
- die Beschreibung einer Methode zum Lösen eines Problems bzw. einer Aufgabe. Durch Iteration bzw. Rekursion wird das Problem schrittweise gelöst bzw. das Ergebnis aufgebaut. [Hier ist ein schönes Diagramm eingefügt, das Eingabe, Prüfung auf Trivialfall, Abarbeitung und Ausgabe in einer Iteration in Beziehung setzt. Weiter wird erläutert:]
  - Ein- und Ausgabegrößen sind spezifiziert. ⇨ Ausgabe genügt der Spezifikation, solange die Eingabe der Sp. genügt
  - jeder Schritt ist effektiv ausführbar/genau beschrieben
  - Algorithmen terminieren/liefern ein Ergebnis
- [ein Diagramm ist skizziert, in dessen Mitte das Wort „Algorithmus" in einem großen Kreis steht. Davon gehen Tentakeln ab, die im Uhrzeigersinn von links oben an um den Kreis herum mit den folgenden Worten markiert sind:] Lösung, Rezept, Produktion, fester Vorgang, wieder verwendbar
- [gegeben ist hier eine nummerierte Liste von Stichpunkten. Bei ihnen ist mir ein Malheur passiert: beim Kopieren sind die ersten vier Punkte verloren gegangen, das Original finde ich im Augenblick nicht. Die Punkte:] 5. Taste auf dem Taschenrechner 6. Ich finde, das muss man nicht wissen! 7. No Comment 8. Eine

aufeinanderfolgende Reihe? 9. Weiß ich nicht! 10. Rennt schreiend weg *** 11. Eine nichtssagende Phrase 12. Äh, ja, Äh ... bewegliche Zahlen!

– [das vorgegebene Blatt enthält hier offensichtlich Notate, wie sie bei einem Brainstorming entstehen. Hier und da sind zusätzlich Linien, Unterstreichungen, Pfeile, Tabellenformeln u.ä. notiert. Die erkennbaren Textbestandteile:] (Rezept) (Vorschrift) (von Maschinen) mechanisch ausführbare Folge von Anweisungen, die vorgegebene Datenstrukturen in gewünschte andere Datenstrukturen überführt. Algorithmus soll in der Regel nach endlicher Zeit stoppen (Beweis: man kann nicht nachweisen, ob ein Programm hält auf Turingmaschinen (Man kann Stoppen nicht beweisen). Im Begriff des Algorithmus liegt der Nichtbeweis durch Algorithmus! (Halteproblem). Super-Turing-Maschinen können das Halteproblem lösen.

2. Einiges zu grundlegenden Begriffen

„Grundlagen der Programmierung von Bildern" – so lautet der Untertitel unseres Kurses. Dem entsprechend besteht unsere Anstrengung aus einem ständigen Wechsel zwischen der Entwicklung von (allgemeinen) Begriffen und der Gewinnung von (persönlichen) Fertigkeiten. Das Eine aber gibt es nicht ohne das Andere.

Programmieren wird zu Recht als die entscheidende Fertigkeit angesehen, über die Informatiker und Informatikerinnen verfügen müssen. Beim Programmieren kann es einerseits nicht bleiben; andererseits ist ohne eine gewisse Sicherheit im Programmieren eine kompetente informatische Tätigkeit auch kaum vorstellbar.

Die Entwicklung interaktiver Medien ist die Aufgabe von AbsolventInnen des Studiums der Digitalen Medien. Sie müssen durchaus programmieren können, falls sie sich von anderen Gestaltenden abheben wollen, die heutzutage längst in ihrer alltäglichen Arbeit ständig Anwendungsprogramme verwenden und das oft virtuos und erfindungsreich tun. Heute gibt es wenige professionelle oder amateurhafte Tätigkeiten, bei denen nicht auch hochentwickelte Anwendungsprogramme eine Rolle spielten.

Die Studierenden der Digitalen Medien müssen über dieses Niveau hinaus gelangen, wenn sie sich nicht mit dem abfinden wollen, was jede andere professionelle Person auch kann. Deswegen ist entweder neben einer erfindungsreichen und sicheren Gestaltungsarbeit (Abgrenzung zum gewöhnlichen Informatiker) eine höher entwickelte formale Begriffswelt erforderlich (Abgrenzung zur gewöhnlichen Designerin).

Wie sich über einen solchen relativ hilflosen Abriss der besonderen Fertigkeiten einer Absolventin der Digitalen Medien deren Profil an Fertigkeiten und Fähigkeiten herausbilden mag, ist heute wohl noch im Fluss der Entwicklung. Erwarten möchte ich, dass der Klärungsprozess stark davon abhängen wird, was der Begriff „Digitale Medien" denn nun umfassen soll. Dass die Interaktion, die Interaktivität, die Gestaltung von Interaktivität, die Gestaltung von interaktiven Medien („Interaction Design" halt, in einem Wort) dabei zentral bedeutsam sein werden, das meine ich sehr wohl.

Man bemerkt an solchen unsicher-sicheren Hinweisen wohl, dass ich die Orientierung „Medieninformatik" für unglücklich halte (sie wird als besondere verschwinden, zurück in die Informatik gehen, wo sie nicht in das neu benannte Feld aufgeht). Man mag auch bemerken, dass ich „Mediengestaltung" für schon besser ansehe, aber andererseits in der Gefahr, die informatische Seite zu wenig zu betonen. – Also, nun ein wenig Begriffliches, Begriffliches, ohne das ein ***Interaction Designer*** nicht auskommen wird.

Menge

Die Mathematik kommt nicht ohne den Begriff der Menge aus. Er lässt sich jedoch nicht formal fassen, sondern wird unter Appell an den gesunden Menschenverstand aus dem alltäglichen Vorverständnis gewonnen, das wir alle haben.

> Eine Menge ist die gedankliche Zusammenfassung irgendwelcher Dinge zu einem neuen Ganzen.

Die Menge zieht quasi einen Kreis um jene Dinge herum, die in ihr zu einem gedanklichen Ganzen vereint werden. Betonung liegt auf „in Gedanken", was naturgemäß beinhaltet, dass die zusammengefassten Gegenstände selbst schon gedanklicher Art sind. Die Menge aller Bäume des Teutoburger Waldes wäre so etwas. Oder die Menge aller Schreibwerkzeuge in meinem Schreibtisch. Oder die Menge aller geraden Zahlen. Oder auch die Menge der grundlegenden Kategorien in Kants Philosophie.

Funktion

Eine Funktion ordnet den Elementen einer Menge A auf eindeutige Weise Elemente einer Menge B zu. Wenn die Funktion den Namen φ tragen soll, so schreiben wir dafür üblicherweise

$$\varphi: A \to B,$$

um auszudrücken, dass durch die Vorschrift φ den Elementen von A solche von B zugeordnet werden. Auf welche Weise das im einzelnen geschieht, ist zunächst gleichgültig. Einzig die Eindeutigkeit der Vorschrift muss gewährleistet sein. (Wir können aber auch nicht deterministische Prozesse betrachten.)

So könnte A die Menge der Teilnehmenden an unserem Kurs sein und φ das Alter in vollen Jahren angeben. Dann wäre B die Menge der positiven ganzen Zahlen (oder auch nur eines Teiles davon, da wir sicher sind, dass niemand älter als 200 Jahre wird).

Eine andere Funktion ordnet vielleicht den Teilnehmenden ihre Namen zu. Da wäre B etwas ganz anderes.

Turing-Maschine

Wir haben die Turing-Maschine kennen gelernt als eine abstrakte (Papier-)Konstruktion, die geeignet ist, bei nur wenigen Annahmen die Vorgänge zu beschreiben, die man „Berechnungen" nennen mag.

> Gegeben seien das Alphabet $A = \{0, 1, \_\}$ von drei Elementen und das Alphabet $B = \{L, R\}$. Gegeben sei weiter eine endliche Menge von Zuständen $S = \{s_0, s_1, ..., s_n\}$. Weiter sei eine Funktion δ folgender Art („Signatur") gegeben: $\delta: S \times A \to S \times A \times B$. Die Funktion δ ordnet einem gegebenen „Zustand" s_i und einem Zeichen aus A einen nächsten Zustand s_j, ein (Ausgabe-)Zeichen aus A und eine „Bewegung" („nach links", „nach rechts") aus B zu. Mit einem „Startzustand" $s_0 \in S$ und einer Teilmenge F von S (die wir die Menge der „finalen Zustände" nennen), definieren wir eine Turing-Maschine T als das Hexatupel $T = (A, B, S, s_0, F, \delta)$.

Anschaulich stellen wir uns die Turing-Maschine als ein beidseitig unendlich langes Band vor, das in Zellen gleicher Größe unterteilt ist und das durch einen Lese-und-Schreibkopf bewegt wird. Diese Bewegung wird von einer Kontrolleinheit gesteuert. Die Maschine ist eine getaktete Zustandsmaschine, d.h. sie befindet sich zu jedem Zeitpunkt in einem bestimmten Zustand aus einer endlichen Menge von möglichen Zustände. In diskreten Schritten geht sie in einen nächsten Zustand über. Ein endlicher Teil des Bandes ist anfangs mit Zeichen aus dem Alphabet A beschrieben (das Zeichen „_" interpretieren wir als „leere Zelle"). Die Kontrolleinheit befindet sich anfangs im Startzustand s_0. Sie liest stets das Zeichen, das sich in der Zelle unter dem Lese-und-Schreibkopf befindet. Aufgrund dieses Zeichens und des aktuellen Zustandes legt die Übergangsfunktion δ fest, in welchen nächsten Zustand die Kontrolleinheit übergeht, welches neue Zeichen statt des alten in das gelesene Feld geschrieben wird und in welche Richtung (nach links oder rechts) das Band um eine Zelle bewegt werden soll. Der vSchreibvorgang kann statt des bisherigen Zeichens das selbe Zeichen wieder anschreiben. Dieser Grundschritt müsste einfach zu verstehen sein. Er wird solange streng wiederholt, bis evtl. ein Endzustand (aus F) erreicht wird. Zu diesem Zeitpunkt stoppt die Maschine. Sie hat die anfängliche Beschriftung des Bandes durch eine neue ersetzt.

Bei der vermutlich etwas ungewohnt erscheinenden Notation der Turing-Maschine als eines Hexatupels (meist nur als Quintupel, da B nicht explizit notiert wird) wird manche denken, was das nun solle. Mathematiker haben sich im 20. Jahrhundert angewöhnt, ihre Strukturbildungen in dieser Weise anzugeben. Man schreibt auf, welche Mengen und Operationen zu einer Strukturbeschreibung benötigt werden und fasst sie dann in einer Klammer zusammen.

Ist das nun nicht schön und klar? Man würde doch so lesen: Eine Turing-Maschine besteht aus Alphabet {0, 1, _} und den Links- und Rechtsbewegungen L und R, sodann vor allem einer Menge von Zuständen, unter denen ein Anfangszustand und mindestens ein Endzustand ausgezeichnet sind. Entscheidend brauchen wir nur noch die Abbildung eines gelesenen Zeichens und eines erreichten Zustandes auf ein neues Zeichen, eine Bewegung und einen Nachfolge-Zustand.

Die Übergangsfunktion wird übrigens oft zerlegt in die Ausgabefunktion $\lambda: S \times A \to A \times B$ und die Zustandswechselfunktion $\delta: S \times A \to S$.

Turing-Berechenbarkeit

Die Turing-Maschine ist ein ungeheuer einfaches, unglaublich anschauliches und riesig mächtiges abstraktes Konzept. In ihm feierte der europäische Geist einen erstaunlichen Höhepunkt an präziser Schlichtheit. Niemand rechnet ***im Kopf*** so wie eine Turing-Maschine ihre Zustände aufeinander folgen lässt. Und doch beschreibt die Turing-Maschine – vom Ergebnis her betrachtet! – alle unsere Berechnungsprozesse! Ist das nicht ungeheuer? Ist das nicht von der atemberaubenden Qualität mancher genialer Musik oder eines aufregenden Gedichtes?

Genauer muss ich allerdings sagen: für jede Funktion, von der wir naiv überzeugt sind, dass sie berechenbar sei, lässt sich eine Turingmaschine angeben. Das hieße also, dass das, was wir naiv, als gewöhnliche Menschen also, für berechenbar hielten, auch im strengen turingschen Sinne berechenbar wäre. Alltagsverständnis und mathematische Präzision kämen zusammen. Unerhört.

Auch das muss jedoch begrifflich erst genauer gesagt werden. Die von einer gegebenen Turing-Maschine ***T*** „berechnete Funktion" f_T definieren wir so: Setzt man ***T*** auf eine beliebige Anfangsfolge □ von Zeichen auf dem Band an und lässt ***T*** laufen, bis ein Endzustand aus ***F*** erreicht ist, so ist die dann auf dem Band vorliegende Endfolge □ der durch f_T berechnete Wert. Wir schreiben dafür f_T: □ l□ □. (Dies ist die Zuordnung spezifischer Werte zueinander, nicht die Signatur.)

> Eine Funktion f: ***D*** □ ***W*** mit Definitionsbereich ***D*** und Wertebereich ***W*** heißt dann Turing-berechenbar, wenn es eine Turing-Maschine ***T*** gibt, deren berechnete Funktion f_T auf ***D*** mit f übereinstimmt. D.h. es muss gelten: □***d*** □ ***D*** □ f(***d***) = f_T(***d***).

(Dabei haben wir stillschweigend angenommen, dass die Elemente der Menge ***D*** bereits so codiert sind, wie die Eingabe der Turingmaschine das verlangt, nämlich binär. Das ist stets möglich und deswegen hier nicht wichtig.)

In den 1930er Jahren wurden einige andere formal exakte Berechenbarkeits-Begriffe vorgeschlagen. Das Schöne daran: sie konnten untereinander als äquivalent bewiesen werden. Das Ergebnis wird in der These von Church zusammengefasst. Deren Inhalt lautet in etwa:

> Alles, was wir gewöhnlich für effektiv berechenbar ansehen, lässt sich auch Turing-berechnen.

Ein umgangssprachlicher Begriff hat also eine mathematisch genaue Entsprechung gefunden. Über effektive Berechenbarkeit kann es damit keinen vernünftigen Streit mehr geben. „Effektiv berechenbar" ist eine Funktion dann, wenn ich in jedem Augenblick eindeutig weiß, was als nächstes zu tun ist, und wenn jeder solche Schritt mechanisch und mit eindeutig feststellbarem Ergebnis ausführbar ist.

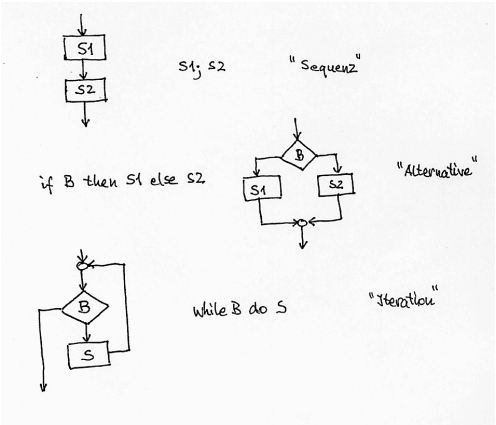
Algorithmus

Algorithmus ist der erste Grundbegriff der Informatik. Grundbegriffe einer wissenschaftlichen Disziplin sind i.d.R. nicht formal zu fassen. Sie beruhen auf dem Einverständnis derjenigen, die eine Disziplin begründen und ausüben. Solch ein Einverständnis muss einfach genug herstellbar sein. Gelegentlich gerät es ins Schwanken und muss evtl. neu gefunden und formuliert werden. Das ist der Wechsel des Paradigma (Thomas Kuhn). In unserem Fall legt man gewöhnlich Folgendes fest:

> Ein Algorithmus ist die Beschreibung eines Berechnungsschemas oder Rechenprozesses. Die Beschreibung muss endlich lang sein. Sie muss als eine Folge von Anweisungen organisiert sein. In jedem Augenblick steht eindeutig fest, welche Anweisung auszuführen ist. Die Anweisungen lassen sich fassen als effektive Anwendungen von Operationen auf Operanden.

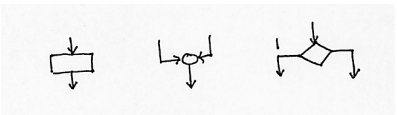
Der Witz beim Algorithmus ist, dass er für eine ganze Klasse von Einzelproblemen die Lösungen zu bestimmen gestattet. Algorithmen lassen sich nun stets mittels der Strukturelemente Sequenz, Alternative und Iteration konstruieren, wobei man sich auf eine Menge von elementaren Operationen stützt.

Man notiert diese Kontrollelemente in simplen Diagrammen und in einer Syntax, die einfach genug zu verstehen ist und in abgewandelten Formen in den Programmiersprachen nach 1960 auftritt, oft folgendermaßen:



Beachtet, dass jedes dieser drei Strukturdiagramme einen Eingangspfeil aufweist, eine Binnenstruktur besitzt und einen Ausgangspfeil. Aus ihnen können in beliebiger Komposition beliebig komplexe Strukturen aufgebaut werden. Sie müssen der einzigen Regel genügen, dass die Eingangs- mit Ausgangspfeilen identifiziert werden. Das ist eine grandioses Ergebnis (von Jacopini und Boehm). Es besitzt in meinen Augen hohen ästhetischen Wert.

Nebenbei gesagt, sind die drei elementaren Strukturen selbst wieder aus drei Primitiven aufgebaut:



Wenn wir nun weiter festlegen, dass ein Algorithmus eine Turing-berechenbare Funktion sein soll, so wird der Algorithmus-Begriff formal exakt gefasst. Man tut dies dennoch i.d.R. nicht, weil sonst viele Rechenprozesse ausgeschlossen werden. Sie laufen auf Computern in Form von Programmen, hätten aber keinen Algorithmus zur Grundlage. Beachte in diesem Zusammenhang insbesondere, dass die Rechenfolgen des hier definierten Algorithmus nicht notwendigerweise sämtliche in endlicher Zeit zum Ende kommen müssen.

Programmiersprache

Programmiersprachen sind formale Systeme, in denen Texte von einer Sorte verfasst werden, die man „Programme" nennt. Ganz so, wie andere Textformen „Gedicht" oder „Erzählung" genannt werden. Eine Programmiersprache braucht zur Notation ihrer Programme ein Alphabet – die endliche Menge jener Zeichen, die verwendet werden dürfen, um Texte der Programmiersprache zu notieren. Zur ihrer Definition gehört weiter vor allem die Definition ihrer Syntax in einer Grammatik. Etwa seit 1958 wird die Syntax einer Programmiersprache i.d.R. als eine Menge von Ersetzungsregeln gegeben: eine Zeichenkette (in der Regel ein einzelnes Zeichen) wird durch eine andere Zeichenkette ersetzt. Jede solche Regel ist von der Form

<syntactic construct> ::= <syntactic form>

Darin ist <syntactic construct> der Name einer syntaktischen Kategorie wie z.B. <program> oder <type> oder <variable> oder <statement> etc. Das Zeichen ::= liest man „ergibt sich als" oder „kann ersetzt werden durch". Die <syntactic form> ist eine syntaktische Form, in der Elemente aus dem Alphabet und andere syntaktische Kategorien aufeinander folgen können. Beispielsweise:

<compound statement> ::= <statement>; <statement>
<iterative statement> ::= while <condition> do <statement>
<conditional statement> ::= if <condition> then <statement> else <statement>

<condition> ::= ...
<statement> ::= ...

Oft werden den – kontextfrei notierten – syntaktischen Regeln noch semantische Nebenabsprachen zugesellt. Sie legen evtl. Einschränkungen oder zusätzliche Effekte fest, die zu beachten sind, wenn die Regel angewandt wird. In solchen Festlegungen erscheint ein (noch) nicht formalisierter Teil der Syntax.

Programm

Ein Programm ist die wichtigste Einheit, die bei der Entwicklung von Software entsteht: Ein Text in einer Programmiersprache; eine separat ausführbare Einheit.

Programme sind janusköpfig: sie werden von Menschen geschrieben und gelesen und insofern auch von Menschen verstanden. Programme werden aber auch von Computern gelesen. Genauer heißt das, dass ein geeignetes Übersetzungs-Programm („Compiler" oder „Interpretierer") das „Quell-Programm" als Zeichenfolge durchmustert und – im Falle syntaktischer Korrektheit – in ein „Objekt-Programm" übersetzt. Quell- und Objekt-Programm sind semantisch äquivalent, d.h. sie berechnen die gleiche Funktion. Das Objekt-Programm ist ausführbar. Es steuert den Computer. Das Quell-Programm soll für Menschen leichter lesbar sein (sogar einigermaßen leicht).

Salopp können wir sagen, dass Programme Algorithmen in programmiersprachlicher Fassung sind. Durch diesen Zwang wird die Notation selbst mechanisch und eindeutig lesbar. Eine (bzw. mehrere) Programmiersprachen zu beherrschen, ist deswegen auch eine der vornehmsten Aufgaben der InformatikerInnen.

Daten

Programme sind die dynamischen Entitäten der Software, Daten sind die statischen. Oder: Programme sind die Operationen, Daten die Operanden. (Für die Operation „Java-Compiler anwenden" ist aber ein Java-Quellprogramm ein Operand. So kann in der Informatik oft ein Ding so oder so betrachtet werden.)

Daten sind Codierungen von Messungen, von Abtast- und Abgreifprozessen, mit denen Eigenschaften wirklicher Gegenstände in Signale verwandelt werden. Die gespeicherten Signale nennt man Daten.

Daten weisen selbst wieder Struktur und Typ auf, um an Ausdrucksstärke zu gewinnen. Haben Programme Kontrollstrukturen, um ihren dynamischen Ablauf organisieren zu können, so besitzen Daten Datenstrukturen, die ihre statische Ausbreitung organisieren. Der Zeit-Aspekt führt in der Software-Welt zu Programm, der Raum-Aspekt zu Daten.

Eng mit den Daten verbunden, so meinen wir, sind Information und Wissen. Das ist einer der Mythen der Informatik. Computer können nur Daten verarbeiten, lesen und ausgeben. Daten werden nur und ausschließlich für Menschen zu Information. Für uns allerdings sind die Übergänge „von Information zu Daten" und „von Daten zu Information" unvermeidlich.

Semiotisch können wir die drei Formen gut unterscheiden. Daten sind (als Signale) mit der syntaktischen Dimension von Zeichen gegeben. D.h. sie sind selbst nur der Anlass für Interpretation und damit Gewinnung von Bedeutung. Information ist mit der semantischen Dimension von Zeichen verbunden. Menschen stellen Daten in Kontexte und dadurch (konventionelle, verbindliche) Bedeutung her. Wissen ist mit der pragmatischen Dimension von Zeichen verbunden. Der Einzelne interpretiert eine Information in einer Situation für sich selbst so oder so, günstig oder ungüstig, klar oder unklar, anregend oder abstoßend, oder was es sonst noch sein mag. Auf Grund des wahrgenommenen Signals (Datum) und in Beachtung der konventionellen Information, die es hervorruft, bildet er diesen oder jenen Aspekt von Wissen, Einsicht, Bewusstmachung heraus, abhängig von Kontext Situation, Erwartung, Werdegang und mehr, um sich für eine Handlung oder ein Verhalten zu entscheiden und entsprechend zu verfahren..