Verwirrende Informatik I – Systeme, Informationen, Berechenbarkeit und Daten

Johannes Reich, johannes.reich@sap.com

6. Juli 2022

Zusammenfassung

Dieser Artikel ist der erste einer Viererreihe, die eine Diskussion über die Grundlagen der Informatik anregen soll. Dabei steht das Begriffsnetzwerks, das den Kern der Informatik ausmacht, im Zentrum. Dieses Begriffsnetzwerk steht in dem Spannungsfeld zum einen weitgehend in unserem alltagssprachlichen Verständnis verankert zu sein, aber zum anderen mittels des spezifischen informatischen Ansatz, sich auf das Unterscheidbare zu fokussieren, auch auf genuin informatischen Konzepten aufzubauen.

In diesem ersten Teil wird die Informatik als eine Theorie der Interaktion vorgestellt und als ihre zentralen Begriffe werden die Zustandsfunktion, das System, die Unterscheidbarkeit und damit die Information, die Berechenbarkeit sowie eine ganze Reihe mathematischer Konzepte eingeführt, allen voran das der Relation, der Funktion, der mathematischen Struktur, der Äquivalenz und der Komposition.

Der grundlegende Bezug zum Informationskonzept als Abstraktion vom Ununterscheidbaren macht die Informatik zu einer Strukturwissenschaft und bindet damit die Klarheit ihrer Begrifflichkeit letztlich an deren Bezug zur Mathematik.

Drei wichtige Konsequenzen werden aufgeführt. Erstens gibt es keine zwei verschiedenen Welten der Informatik und der Physik, die auf geheimnisvolle Weise "wechselwirken". Sondern: Es gibt nur eine Welt, die wir unterschiedlich beschreiben. Zweitens lässt sich mit Klärung der Frage, was aus Sicht der Informatik eigentlich zwischen Systemen, die über Signale miteinander kommunizieren, "transportiert" wird – nämlich die Information – auch feststellen, was insbesondere *nicht* transportiert wird – nämlich irgendeine Bedeutung. Und drittens ergibt sich ein gewisser "virtueller" Charakter rein informatisch beschreibbarer Systeme, der dort endet, wo die Abstraktion, dass die Struktur von Systemen in der Interaktion invariant bleibt und es allein auf die Unterscheidbarkeit ankommt, ihre Gültigkeit verliert.

1 Einleitung

"There is nothing more practical than a good theory."

James C. Maxwell (1831-1879)

Dieser Artikel ist der erste einer Viererreihe, die eine Diskussion über die Grundlagen der Informatik anregen soll. In diesem ersten liefere ich die Motivation und stelle die Konzepte vor, die meiner Ansicht nach die Informatik begründen, nämlich System, Information, Berechenbarkeit und Daten. Im zweiten Teil thematisiere ich die Interaktion zwischen Systemen und die Koordination innerhalb von Systemen mit Hilfe des Kompositionskonzeptes. Im dritten und vierten Teil gehe ich auf zwei Themen ein, bei denen meiner Wahrnehmung nach mehr Klarheit innerhalb der Informatik und auch zwischen Informatik und ihren Anwendungsdisziplinen notwendig ist. Zum einen auf das Verhältnis von Beschreibung und Beschriebenem in Teil drei und in Teil vier auf die Frage, welchen Beitrag die Informatik zum Verständnis des Begriffs der Semantik oder Bedeutung leisten kann.

Mein Eindruck ist, dass die Situation der heutigen Informatik mit der der Mathematik zu Beginn des 20. Jahrhunderts in gewisser Weise vergleichbar ist. Als Ausdruck ihres vergleichsweisen jungen Charakters werden wesentliche Begriffe von eher naiven, sehr unterschiedlichen und teilweise widersprüchlichen Vorstellungen geprägt.

Betrand Russel wies 1902 Gottlob Frege in einem Brief darauf hin [9], dass die naive Auffassung des Mengenbegriffs, zu jeder sinnvoll formulierbaren Eigenschaft müsse es auch eine Menge der Dinge geben, denen diese Eigenschaft zukommt, widersprüchlich war. Sein einfaches Beispiel waren die Mengen mit der Eigenschaft, sich nicht selbst zu enthalten. Diese ließen sich nicht zu einer Menge zusammenfassen, weil die Frage, ob diese Menge sich nun selbst enthalte oder nicht, zu einem Widerspruch führt.

Die vermeintliche Unklarheit was eine Menge "tatsächlich" ist, hielt die Mathematiker in der Folge aber nicht davon ab, sie als Grundlage der modernen Mathematik anzusehen. Tatsächlich hatte sich die Vorstellung der Mathematiker, was unter einer "genauen Kenntnis" eines mathematisches Begriffs zu verstehen ist, im Laufe dieser Diskussion geändert.

Leslie Lamports [15] hat die Tendenz der Informatik, Scheinprobleme durch unterschiedliche Formulierungen zu erzeugen, als "Whorfian Syndrome" bezeichnet. Tatsächlich muss sich diese "Whorfian Syndrome"-Informatik auf Grund ihres Querschnittscharakters mit den Konzepten vieler anderer Wissensdomänen in einer wechselseitigen Beziehung auseinandersetzen. Diese wechselseitige Beziehung birgt einerseits die Chance, einen entscheidenden Beitrag zu einem differenzierteren Verständnis unserer Welt zu leisten aber andererseits auch das Risiko, nicht nur die Informatik selbst, sondern auch unsere Alltagswelt mit falsch verstandenen informatischen Konzepten unnötigerweise erheblich zu verwirren. Für beides gibt es mittlerweile gute Beispiele.

Was scheint in der Informatik auf den ersten Blick verwirrend? Einerseits scheint alles in der Informatik auf dem Transport von Informationen zu basieren – andererseits scheint dieser aber bei der Beschreibung von Operationen, wie sie die imperativen Programmiersprachen konzeptuell dominieren, gar keine Rolle zu spielen. Einerseits scheint die Informatik durch das Konzept der Berechenbarkeit geprägt zu sein, in dem Determinismus herrscht und in dem Zustand nur passager notwendig ist – andererseits scheinen die Interaktionen zwischen komplexen Systemen "auf gleicher Augenhöhe" in der Regel aber doch nichtdeterministisch und zustandsbehaftet zu sein. Einerseits scheinen Systeme durch Interaktion zu Supersystemen zu komponieren – andererseits aber auch wieder nicht. Einerseits scheint in der Informatik ein Modell etwas zu beschreiben – andererseits scheint in der Informatik ein Modell etwas Beschriebenes zu sein. Einerseits scheint das Konzept der Semantik im Bereich der Berechenbarkeit überflüssig – andererseits scheint die Informatik zu der Aufklärung von Semantik einen wesentlichen Beitrag liefern zu können.

Diese Verwirrung lässt sich, wie ich versuche werde zu zeigen, mit einer genaueren, v.a. auf die Mathematik gestützte Betrachtung, aufklären. Bei genauerer Betrachtung ist aus informatischer Perspektive eben System nicht gleich System, Interaktion nicht gleich Interaktion, ausgetauschte Zeichen nicht gleich ausgetauschten Zeichen, Zustand nicht gleich Zustand, Entscheidung nicht gleich Entscheidung und Freiheit ist nicht gleich Freiheit.

Die mit ungenauer Betrachtung sich einschleichenden Widersprüche wirken sich hingegen sehr hinderlich auf das Verstehen aus. Beispiele dafür sind etwa das Verhältnis von Information und Daten, das Verständnis von Architektur oder die Begründung von Semantik.

Chaim Zinn [32] dokumentiert 130 verschiedene Definitionen von Daten, Information und Wissen. Sollte man etwa Informationen als "Informiertheit" in Bezug auf Daten verstehen, wie es das recht verbreitete Model der sogenannten "Wissenspyramide" (z.B. [11]) vorschlägt – oder doch eher Daten bezogen auf Informationen?

Redet ein Informatiker über Softwarearchitektur, stellt sich zum einen die Frage, ob er nicht doch über Systemarchitektur redet. Und verstehen wir unter der Architektur eines IT-Systems seine Struktur, dann stellen wir zum anderen fest, dass schon kein Konsens beim Verständnis seiner Elemente, also der Komponenten und deren Beziehung über Interfaces untereinander besteht. In sogenannten Referenzachitekturen werden häufig hierarchische Strukturen ohne klares Ordnungskriterium eingeführt (z.B. [12, 29, 8]). Die Spannung zwischen der Objektorientierung und über Nachrichten kommunizierende Prozessen wurde bisher nicht gelöst. Und auch ob ein Modell nun eine Beschreibung ist oder etwas Beschriebenes ist, wie schon erwähnt, unklar.

Bei der Frage, wie unsere sprachlichen Ausdrücke zu ihrer Bedeutung kommen, und damit zusammenhängend, welche Rolle das Konzept der Semantik für die aufwandsarme Herstellung von Interoperabilität technischer Systeme im Zeitalter der umfassenden Vernetzung spielt, wird es wirklich bunt.

Die Folgen sind nicht nur innerhalb der Informatik dramatisch. Dinge, die im Alltag jedem Menschen verständlich erschienen, etwa die Unterscheidung

zwischen dem Datenblatt einer Heizung, der Heizung selbst, sowie den Geschäftsprozessen, in die eine Heizung eingebunden ist, scheinen mit dem Auftauchen der Informatik plötzlich rätselhaft verwirrend (Beispiel aus [25]). Und die Folgen sind teuer: So kristallisierte sich etwa in Standardisierungsdiskussionen, die im Rahmen der Industrie 4.0-Initiative der Bundesregierung seit 2012 stattfinden, erst nach Jahren langsam heraus, dass mit neu eingeführten Termen wie "Verwaltungsschale" gleichzeitig über Datenmodelle (entspricht dem Datenblatt), Komponentenmodelle (entspricht der Heizung) und Prozessmodelle (entspricht den Geschäftsprozessen) geredet wurde [25] (mittlerweile als "Verwaltungsschale" vom Typ 1, 2 und 3 bezeichnet [3]).

Informatik ist keine Kunst, oder gar Zauberei, sondern eine Wissenschaft zur Gestaltung unserer Welt. Und die meisten ihrer Konzepte, wie Dokumente, Prozesse, Systeme etc. können meiner Ansicht nach jeder gut ausgebildete Ingenieur mit seinem Alltagsverständnis gut erfassen. Dennoch gibt es einiges, das die Informatik in der Tat besonders und in gewissem Sinne auch auf den ersten Blick verwirrend macht. Mit dieser Reihe unternehme ich daher auch den Versuch aufzuzeigen, wo dieses Alltagsverständnis eine gute Basis der gegenseitigen Verständigung mit Informatikern darstellt und an welchen Stellen uns tatsächlich genuin informatische Konzepte begegnen, die dann aber auch jeder, der sich mit einer Informatikerin verständigen will, wenigstens in ihren Grundzügen kennen sollte. Es ist in diesem Rahmen sicherlich nicht möglich alle diese Konzepte im Detail auszubreiten. Dazu verweise ich auf [21, 23, 24].

Darüber hinaus ist mir wichtig zu vermitteln, dass eine wohlverstandene Informatik tatsächlich menschlicher wird. Mit ihr können wir aufzeigen, dass eben nicht die Berechenbarkeit die Welt mit ihrem Determinismus dominiert, sondern wir permanent große Vorteile aus ihrer Unberechenbarkeit, aus ihrer Unbestimmtheit ziehen. Tatsächlich leistet eine solche Informatik einen essentiell wichtigen Beitrag für unser Verständnis solch wichtiger alltagssprachlicher Begriffe wie der unserer persönlichen Freiheit – was vor den aktuellen gesellschaftlichen, wesentlich durch die Informatik möglich gemachten Entwicklungen der Megainteraktionsnetzwerke, auch zu erwarten wäre und meiner Ansicht nach auch dringend erforderlich ist.

Zum weiteren Vorgehen in diesem Artikel: Den inhaltlichen Teil beginne ich mit dem nach meinem Verständnis zentralen Begriff des Systems. Zum einen unterstreicht das den Charakter der Informatik als Ingenieursdisziplin. Zum anderen brauchen wir ihn, um das eigentliche Konzept der Informatik zu definieren, das sie unverwechselbar macht: die Information. In Abschnitt 3 beschreibe ich, wie Information durch den Fokus auf das Unterscheidbare entsteht. Der Rest ist eigentlich die Anwendung mathematisch formulierbarer Konzepte. Dies zeigt sich wesentlich in dem Konzept der Komposition, das meiner Ansicht nach in verschiedener Gestalt die Informatik dominiert. Zum einen in der Informationsverarbeitung im Sinne der Theorie der Berechenbarkeit, die ich in Abschnitt 4 vorstelle und auf die sich der Datenbegriff in Abschnitt 5 wesentlich stützt. Zum anderen in der Beschreibung der Interaktion von Systemen in ihren verschiedenen Spielarten, auf die ich im zweiten Artikel dieser Reihe dann eingehen werde. Insofern schließe ich mich Stavros Tripakis an, der in [30] meint, dass *"Compo-*

sition and compositionality should probably be the most important concepts in modern system thinking".

2 Der Systembegriff

Naturwissenschaftlich beschreiben wir die Welt mittels Zustandsfunktionen, die die Zeit auf Zustandswerte abbilden (s. Abb. 1). Andere häufig verwendete Bezeichner für diese Funktionen sind auch "Zustandsgröße", "Variable", "Signal" oder einfach nur "Zustand", wobei letzterer oft auch den Wert bezeichnet, den eine Zustandsfunktion zu einem Zeitpunkt annimmt. Je nach Zeitskala und Wertebereich kann es sehr unterschiedliche Zustandsfunktionen geben. Für die klassische Informatik genügen diskrete Zeitskalen und endliche Wertebereiche. Das bedeutet insbesondere, dass wir im Weiteren davon ausgehen, dass eine Zustandsfunktion zu einem Zeitpunkt nur Einzelwerte und keine Mengen repräsentieren kann.

Systeme entstehen, wenn wir annehmen, dass diese Zustandswerte nicht unabhängig voneinander sind, sondern sich aufeinander in der Zeit beziehen. Die in gewisser Weise einfachste Beziehung ist die einer Funktion, weil ihre Richtung, basierend auf ihrer Eindeutigkeit eine klare Abgrenzung zwischen "Eingabe" und "Ausgabe" ohne weitere Zusatzannahmen ermöglicht. Ein solches System grenzt dann in natürlicher Weise ein "Inneres" vom Rest der Welt, der Umgebung ab, in dem vermittels der Systemfunktion Eingabe- von Inneren- und Ausgabezustandsfunktionen unterschieden werden können.

Damit erhalten wir ein sehr dynamisches Bild von der Welt, weil nun Systeme beständig entstehen und vergehen, je nachdem ob sich ein funktionaler Zusammenhang zwischen den Werten verschiedener Zustandsfunktionen in der Zeit ausbildet oder auch verschwindet.

Dieses Systemmodell illustriere ich in Abb. 1. Für eine Diskussion alternativer Systemmodelle verweise ich auf [24]. Dort wird gezeigt, wie die Systemfunktion f einen Eingabezustandswert $in(t)$ und inneren Zustandswert $q(t)$ zum Zeitpunkt t in einem Zeitschritt $t \to t' = t + 1$ auf einen Ausgabezustandswert $out(t')$ und einen neuen inneren Zustandswert $q(t')$ abbildet.

Da sich in unserem Systemmodell das Ergbnis der Systemfunktion immer auf den nächsten Zeitpunkt $t' = t + 1$ bezieht, können wir aus unserer Verhaltensbeschreibung die explizite Abhängigkeit von der Zeit eliminieren. Dazu gehen wir zu einem I/O-Transitionssystem (I/O-TS)[1] über (z.B. [20, 1]), dessen Übergänge durch einen 4er-Tupel (i, o, p, q) aus Eingabezeichen i, Ausgabezeichen o, Startzustand p und Zielzustand q beschrieben werden. Dabei beziehen sich i und p auf den Zeitpunkt t und o und q auf den Zeitpunkt t'.

Systeme, die immer ihre gesamte Eingabe abbilden, nenne ich einfach. Um auch das Verhalten von Systemen darstellen zu können, die nur auf Teile ihrer Eingaben reagieren, führe ich zusätzlich das leere Zeichen ϵ ein und definiere für ein Alphabet $A^\epsilon = A \cup \{\epsilon\}$. Besitzt ein Eingabezeichen in einer Komponente das

[1] Diese I/O-TS werden in der Literatur auch Transducer und im deterministischen Fall auch Mealy-Automat genannt [26].

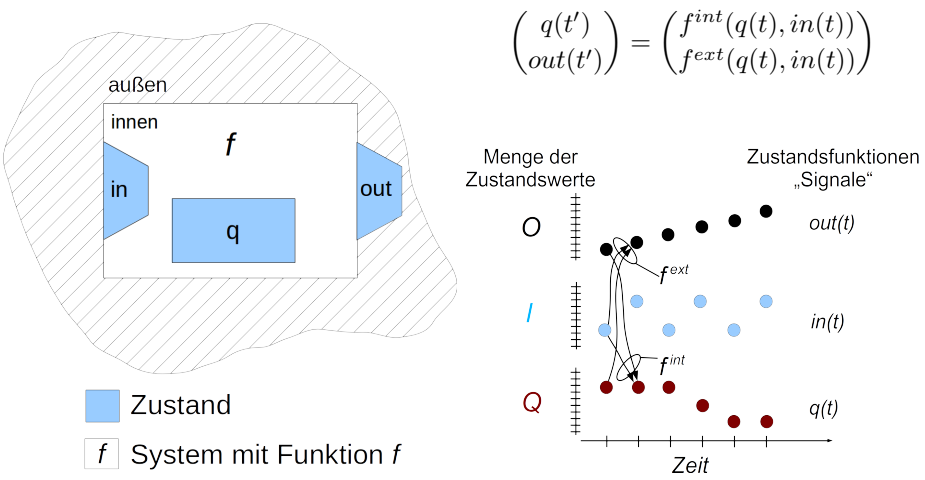

Abbildung 1: Ein diskretes System mit seinen drei, ggfs. vektorwertigen Zustandsfunktionen $(in, out, q) : T \to I \times O \times Q$, die die Systemzeit T auf ihre jeweilige Wertemengen I, O, Q abbilden. Der Zusammenhang dieser Werte zu den Zeitpunkten (t, t') wird durch die Systemfunktion $f = (f^{int}, f^{ext})$ hergestellt, wie rechts dargestellt.

leere Zeichen ϵ, dann spielt diese Komponente für die Abbildung dieses Zeichens keine Rolle. Solche Systeme nenne ich "Multiinputsysteme".

Beschreiben wir mit einem I/O-TS $\mathcal{A}$ das ganze System $\mathcal{S}$, dann ist die Transitionsrelation $\Delta_{\mathcal{A}}$ der Graph der Systemfunktion $f_{\mathcal{S}}$ und entsprechend deterministisch. Für die Transitionsrelation $\Delta_{\mathcal{A}} \subseteq I^{\epsilon} \times O^{\epsilon} \times Q \times Q$ des I/O-TS $\mathcal{A}$ gilt dann für alle Zeiten $(t, t' = t+1)$ seiner Existenz, dass $(in_{\mathcal{S}}(t), out_{\mathcal{S}}(t'), q_{\mathcal{S}}(t), q_{\mathcal{S}}(t')) \in \Delta_{\mathcal{A}}$ und dass $\Delta_{\mathcal{A}}$ darüber hinaus keine weiteren Elemente enthält.

Mit einem I/O-TS lassen sich jedoch ebenso nur Teile des Verhaltens eines Systems beschreiben, etwa im Sinne einer Projektion. Das wird v.a. für die Beschreibung des Verhaltens von interaktiven Systemen wichtig. Dann ist die Transitionsrelation ggfs. auch nichtdeterministisch.

Die Interaktion solcher Systeme basiert auf gemeinsamen Zustandsfunktionen die für die "sendenden" Systeme die Ausgabe und für die "empfangenden" System die Eingabe repräsentieren. In der Welt der I/O-TSe entspricht dieser Systemkopplung die Kopplung über identische I/O-Zeichen: Die Ausgabezeichen des "sendenden" TS sind die Eingabezeichen des "empfangenden" TS. Die Transitionsrelation des Produkt-TS der interagierenden TSe ergibt sich als entsprechende systematische Restriktion der unbeschränkten Produkt-Transitionsrelation gemäß der Anforderung, dass eingehende Zeichen zu verarbeiten sind (s. [21]).

3 Information

Ralph V. L. Hartley [10], Claude E. Shannon [27, 28] und andere hatten die bahnbrechende Idee, die Zustände der Natur nicht als Orte, Geschwindigkeiten oder Spannungen zu bemessen, sondern sich auf das Unterscheidbare zu fokussieren. Um weiter über die Zustandswerte reden zu können, mussten sie nun für jeden einzelnen Wert ein neues Symbol einführen und fassten alle diese neu eingeführten Symbole zu einem "Alphabet" zusammen. Die Information war "erfunden". Also etwa "0" oder "1" im Falle einer binären Unterscheidbarkeit, wobei die Gestalt der Zeichen unerheblich ist. Hauptsache, sie sind ihrerseits unterscheidbar und alle haben sich darauf geeinigt welche unterscheidbaren Zustandswerte in welchen Systemen mit welchen Zeichen benannt werden sollen.

Gehen wir zusätzlich davon aus, dass die beteiligten interagierenden Systeme sich durch die Interaktion strukturell nicht verändern, können wir mit dieser Information nun genau zwei Dinge anstellen: wir können sie erstens "übertragen" und zweitens können wir sie "verarbeiten".

Information zu übertragen bedeutet, den Wert einer Ausgabezustandsfunktion eines Systems in einer Eingabezustandsfunktion eines weiteren Systems so zu reproduzieren, dass man ihn sowohl hier wie da mit demselben Symbol benennen darf. Informationsübertragung geschieht demnach per Definitionem immer *zwischen* Systemen. Sinnvoll über "Informationsübertragung" zu sprechen setzt somit die Einigung über die Bezeichnung der unterscheidbaren Zustandswerte in den verschiedenen Systemen strikt voraus. Im einfachsten Fall ist die Ausgabezustandsfunktion des "Senders" und die Eingabezustandsfunktion des "Empfängers" schlicht identisch. In der Welt der I/O-TSe entspricht das exakt dem beschriebenen Kopplungsmechanismus.

Damit erhalten wir auch die nachträgliche Rechtfertigung, warum wir uns im Rahmen der Informatik auf die in Abschnitt 2 beschriebenen diskreten Systeme beschränken können und finden in den Alphabeten I, O und Q die Mengen unterscheidbarer Zeichen.

Informationen zu verarbeiten heißt dann, die Zeichen über die einfache Identitätsbeziehung hinaus aufeinander zu beziehen, was per Definitionem *innerhalb* der Systeme geschieht. Systeme, deren Funktion die Informationsübertragung selbst ist, nennen wir "Kanal".

4 Komposition und Berechenbarkeit

4.1 Komposition

Komposition bedeutet in der Sprache der Mathematik aus zwei oder mehr mathematischen Entitäten mithilfe einer mathematischen Abbildung eine weitere zu bilden. Sind $\mathcal{S}_1, \ldots, \mathcal{S}_n$ unsere betrachteten mathematischen Entitäten und $C_{\mathcal{S}}$ als partielle Funktion[2] der Kompositionsoperator mit entsprechenden

[2] Partiell bedeutet, dass diese Funktion nicht für alle möglichen Entitäten erklärt ist, d.h. nicht jede Entität für jede Komposition geeignet ist.

Definitions- und Wertemenge, dann ergibt sich die zusammengesetzten Entität als:

$$\mathcal{S}_{ges} = C_{\mathcal{S}}(\mathcal{S}_1, \ldots, \mathcal{S}_n). \tag{1}$$

So bildet etwa die Addition das Kreuzprodukt der natürlichen Zahlen auf die Menge der natürlichen Zahlen ab oder der Negationsoperator der Aussagenlogik die Menge aller Aussagen auf sich selbst ab. Sind $\mathcal{S}_1, \ldots, \mathcal{S}_n$ und $\mathcal{S}_{ges}$ aus derselben Wertemenge, spreche ich auch von einer "homogenen", ansonsten von einer "inhomogenen" Komposition. Beispiele homogener Kompositionen sind etwa die im nächsten Abschnitt vorzustellende Komposition berechenbarer Funktionen aus berechenbaren Funktionen oder auch die im nächsten Artikel vorgestellte Komposition von Systemen zu Supersystemen.

4.2 Informationsverarbeitung mit berechenbaren Funktionen

Die Beschreibung der Informationsverarbeitung wurde durch drei auf den ersten Blick recht unterschiedliche, aber bei näherer Betrachtung äquivalente Konzepte der rekursiven Funktionen [14], der Turingmaschine [31] sowie dem λ-Kalkül [5] begründet – vielleicht die Keimzelle des Lamportschen "Whorfian"-Syndroms.

Das Konzept der rekursiven, oder auch berechenbaren Funktionen hat den Vorteil, dass wir für seine Formulierung keine Konzepte einführen müssen, die an sich schon einen gewissen "informatischen" Charakter haben, wie etwa Automaten oder formale Kalküle. Es illustriert meiner Ansicht nach am besten den konstruktiven Charakter der Informatik als Ingenieursdisziplin und v.a. auch ihre Beziehung zum Systembegriff.

Konkret konnte Stephen Kleene in seiner bahnbrechenden Arbeit [14], basierend auf Überlegungen von Kurt Gödel, zeigen, dass ausgehend von gegebenen Elementaroperationen (Nachfolger, Konstante und Identität) sich alle weiteren berechenbaren Operationen durch die folgenden 3 Regeln konstruieren lassen (F_n sei die Menge aller Funktionen auf den natürlichen Zahlen mit Arität n):

1. *Comp:* Sei $g_1, \ldots, g_n \in F_m$ und $h \in F_n$ berechenbar, dann ist $f = h(g_1, \ldots, g_n)$ wiederum berechenbar.

2. *PrimRec:* Sind $g \in F_n$ und $h \in F_{n+2}$ berechenbar und $a \in \mathbb{N}^n$, $b \in \mathbb{N}$ dann ist die Funktion $f \in F_{n+1}$, gegeben durch $f(a,0) = g(a)$ und $f(a, b+1) = h(a, b, f(a,b))$ wiederum berechenbar.

3. *μ-Rec:* Sei $g \in F_{n+1}$ berechenbar und $\forall a \exists b$ so dass $g(a,b) = 0$ und die μ-Operation $\mu_b[g(a,b) = 0]$ ist definiert als das kleinste b mit $g(a,b) = 0$. Dann ist $f(a) = \mu_b[g(a,b) = 0]$ wiederum berechenbar.

Eine Funktion wird also berechenbar, wenn sie sich aus anderen, wiederum berechenbaren Funktionen zusammensetzt, bis hinab zu "irgendwie" gegebenen Elementarfunktionen. Berechenbarkeit ist damit ein sehr einfaches (homogen) kompositionales Konzept, mit gerade einmal 3 Kompositionsregeln: Man kann

Funktionen hintereinander oder parallel anwenden (*Comp*), man man sie eine vorgegebene Anzahl auf Zwischenergebnissen anwenden (*PrimRec*), oder man kann sie so oft auf Zwischenergebnissen anwenden, bis ein (wiederum berechenbares Problem) damit gelöst wird, etwa eine Nullstelle gefunden wurde (μ-*Rec*).

Da wir gesagt hatten, dass die konkrete Wahl unseres Alphabets irrelevant ist, lässt sich dieser Berechenbarkeitsansatz unmittelbar von den natürlichen Zahlen auf alle diskreten Alphabete übertragen. Darüber hinaus sind, vergleichbar mit Konstanten, auch Zufallsvariablen als Elementarfunktionen möglich. Mit diesen lassen sich auch nichtdeterministische Berechnungen konstruieren.

5 Daten

Alonso Church hat das (Daten-) Typkonzept in die Informatik eingeführt, um die Wohlgeformtheit seines λ-Kalküls sicherzustellen [6]. Tatsächlich ist der Datenbegriff wegen der Äquivalenz der drei Ansätze zur Berechenbarkeit nicht auf ein Kalkül oder gar eine Programmiersprache angewiesen.

Anschaulich können wir Daten als Informationen verstehen, von denen wir grundsätzlich wissen, wie wir sie zu verarbeiten haben [22, 2]. Was meine ich damit konkret? Zunächst verstehen die meisten Informatiker unter Datentypen die Verbindung von einem Alphabet und einer Menge von Funktionen, die dieses Alphabet als Wertebereich haben [16, 17, 19]. Es gibt jedoch auch abweichende Vorstellungen, etwa das Alphabet alleine schon als Typ zu betrachten (z.B. [4]) oder Typen als Äquivalenzklasse von Variablen (z.B. [18]).

Allerdings gibt es bei der Menge der Funktionen Klärungsbedarf. Beschränken wir uns auf eventuell gegebene Elementarfunktionen, dann wären etwa die beiden Datentypen $\mathcal{T}_1 = (\mathbb{N}, \{incr\})$ mit $incr(x) = x+1$ und der erweiterte Datentyp $\mathcal{T}_1' = (\mathbb{N}, \{incr, incr2\})$ mir $incr2(x) = incr(incr(x))$ verschieden. Das entspricht aber nicht unserer Intuition. Dies können wir berücksichtigen, wenn wir die Menge der betrachteten Funktionen auf alle Funktionen, die sich aus den gegebenen Elementarfunktionen unter den 3 Berechenbarkeitsregeln bilden lassen, erweitern. Dann wären beide Datentypen tatsächlich in diesem Sinne gleich.

Das entspricht unserer Intution, dass sich etwa in einer Programmiersprache ein Datentyp nicht ändert, nur weil wir mit seiner Hilfe eine neue Operation definiert haben.

Mit "Daten als Informationen verstehen, von denen wir grundsätzlich wissen, wie wir sie zu verarbeiten haben" meine ich daher konkret, dass wir zu einem Alphabet eine zugeordnete Menge von Elementaroperationen kennen und so mit den 3 Berechenbarkeitsregeln grundsätzlich alle damit konstruierbaren Funktionen konstruieren könnten.

Ein einfaches Beispiel ist der Datentyp einer 64-Bit Gleitkommazahl gemäß IEEE Standard. Hier sind die Elementaroperationen +, -, * und /, die heutzutage alle von den modernen Rechenzentraleinheiten zur Verfügung gestellt werden. Sie alle gehen von einer bestimmten Struktur dieser 64 Bits aus, etwa dass die ersten 56 Bits für die Mantisse und die restlichen 8 Bits für den Exponenten

zu einer zuvor vereinbarten Basis vorgesehen sind. Genau genommen sind dies Elementarfunktionen, die auf Paaren von Gleitkommazahlen operieren. Die genaue theoretische Behandlung solcher Datentupel kann unterschiedlich erfolgen und führt über diesen Artikel hinaus. Ebenso die Möglichkeit, Operationen als Datümer zu betrachten (z.B. [16]).

Zwei begriffliche Abgrenzungen sind zum Verständnis von Daten im vorgestellten Sinne wichtig. Zum einen die Abgrenzung gegenüber sogenannten "Abstrakten Datenypen" oder den mit ihnen verwandten "Objekten" im Sinne von Instanzen von Klassen der Objektorientierung (OO). Solche Objekte stellen eine Verbindung von einer konkret bestimmten, endlichen Menge berechenbarer Funktionen mit einer konkreten Zustandsfunktion her. Von deren Zustandswerten wissen wir dann nicht nur "grundsätzlich" wie sie verarbeitet werden "könnten", sondern wir wissen, wie sie tatsächlich verarbeitet werden. Objekte, bzw. Instanzen von Klassen im Sinne der OO sind also eher Systeme (s. Abschnitt 6.1) und keine Datentypen im vorgestellten Sinn. Entsprechend benötigen sie eine Initialisierung ("Konstruktor"). Diesen Bezug haben Datentypen nicht.

Zum anderen ist die Abgrenzung zwischen einem Datentyp, der sich auf eine Wertemenge bezieht, und dem Konzept, das er ggfs. repräsentieren soll, wesentlich. Unterhalten sich zwei Menschen etwa über Temperatur in einem physikalischen Sinn, dann ist es sehr unwahrscheinlich und auch gar nicht notwendig, dass sie beide genau dasselbe Verständnis von Temperatur haben. Vielleicht ist der eine Mensch eine Physikerin mit einer Promotion im Bereich thermodynamischer Beschreibung von Quantenvielteilchensysteme und der andere Mensch ihr 8-jähriger Sohn, der ihr erzählt, dass er heute, bei 30°C in der Schule hitzefrei bekommen hat. Reden hingegen zwei Menschen über einen Datentyp "Temperatur", dann ist es möglich und auch zu hoffen, dass beide über dasselbe reden, also beide dieselben Elementaroperationen kennen und das Konzept der Berechenbarkeit verstanden haben.

6 Abstraktion durch Äquivalenzklassenbildung

Bisher haben wir uns in unserer Darstellung immer auf einzelne Zeichen und Zustandsübergänge beschränkt. Das wird für komplexere Anwendungsfälle schnell unübersichtlich. Komplexere Anwendungsfälle lassen sich deutlich einfacher darstellen, wenn es uns gelingt, uns auf das "Wesentliche" zu fokussieren. In Bezug auf unsere I/O-TSe bedeutet das, dass wir bestimmte Transitionen bezügliche eines Kriteriums als äquivalent ansehen.

Mathematisch gesprochen wollen wir mit Hilfe einer Relation die bisher betrachteten Transitionsrelationen in Äquivalenzklassen aufteilen unter Erhaltung einer uns wesentlichen Eigenschaft.

Eine so genannte Äquivalenzrelation $\sim$ auf einer Menge R ist eine Relation, die die drei folgenden Eigenschaften für drei Elemente $a, b, c \in R$ besitzt:

1. *Reflexivität*: $a \sim a$. D.h. jedes Element ist zu sich selbst äquivalent.

2. *Symmetrie*: $a \sim b \Rightarrow b \sim a$: D.h. Äquivalenz gilt immer in beide Richtungen.

3. *Transitivität*: $a \sim b$ und $b \sim c \Rightarrow a \sim c$. D.h. sind jeweils a zu b und b zu c äquivalent, so ist auch a zu c äquivalent.

Eine Äquivalenzrelation hat die gewünschte Eigenschaft, ihre Bezugsmenge immer vollständig in paarweise disjunkte Teilmengen, die sogenannten Äquivalenzklassen aufzuteilen und dabei gewisse Eigenschaften zu konservieren. Damit lässt sich die mathematische Äquivalenzklassenbildung als Abstraktion vom Einzelfall hin zu einer "Verallgemeinerung" verstehen.

Ich stelle zwei unterschiedliche Äquivalenzklassenkonstruktionen für die Transitionsrelation der I/O-TSe vor, die unterschiedliche Aspekte hervorheben. Eine für den deterministischen und eine für den nichtdeterministischen Fall.

6.1 Objekte im Sinne der Objektorientierung

Im Fall eines deterministischen I/O-TSs können wir die Transitionsrelation Δ in Subtransitionsrelationen Δ_l zerlegen, die ihrerseits deterministisch sind, soweit es uns opportun erscheint. Damit definiert jede von ihnen eine Funktion δ_l mit $(o, q) = \delta_l(i, p)$ für $(p, q, i, o) \in \Delta_l$. Tatsächlich besteht δ_l damit aus zwei Teilen, wobei der erste Teil für die Objekt-orientierte Art [7, 13] steht, eine Zustandsänderung eines sogenannten "Objektes" darzustellen, indem die Zustandsfunktion mit einem Namen bezeichnet wird, dem "Objektnamen". Dieser Zustandsfunktion werden dann die Operationen zugeordnet, etwa so: outputParameters = objectName.operation(inputParameters). Die zweite Gleichungsrelation für die Entwicklung des Wertes des inneren Zustands ist in der Welt der Objektorientierung nur implizit gegeben.

Wir sehen, dass "Objekte" im Sinne der Objektorientierung eigentlich Systeme in unserem Sinne sind, weswegen man eigentlich besser "System"- statt "Objekt"-Orientierung sagen sollte.

6.2 Dokumente und Hauptzustände

Gewöhnlich beschreiben wir Interaktionen, etwa im Geschäftsleben nicht mit dem Bezug auf konkrete, bis ins Detail ausgefüllte Dokumente. Stattdessen abstrahieren wir und sagen etwa, "Ich bestelle ein Ware, indem ich eine Bestellung versende." oder "Ich überweise einen Geldbetrag, indem ich eine Überweisung ausfülle und abschicke.". Und vom Empfänger eines solchen Dokuments sagen wir vielleicht, dass er von einem Wartezustand mit Erhalt des Dokuments zusammen mit seiner Entscheidung, unserem Ansinnen nachzukommen, in einen Zustand entsprechender Geschäftigkeit wechselt.

D.h. wir abstrahieren von vielen Details, was bestellt wird, wie viel bestellt wird, die Identitäts- und Adressdaten des Kunden usw., und betrachtet alle diese Übergänge in Bezug auf diese Darstellung als gleichwertig. Diese Fokussierung führt offensichtlich dazu, die Zeichenmengen in "Dokumentenklassen"

und die Zustandswertemenge in "wichtige" und "weniger wichtige" Zustände zu unterteilen.

Die Grundidee dieser Äquivalenzklassenkonstruktion besteht darin, diese Partitionierung nicht direkt von i, o, p und q abhängig zu machen, sondern nur von der wichtigen Komponente des Zustands sowie der Dokumentenklasse des Eingabedokuments und der Erfüllung einer zusätzlichen Bedingung.

Ich führe daher zunächst eine Funktion $parse$ ein mit $(docCls, param) = parse(a)$, die jedem Zeichen a eines I/O-TS eine Dokumentenklasse $docCls(a)$ und einen Wert einer Parametermenge $param(a)$ zuordnet. Wir können also jedes Zeichen als eine Instanz eines Dokuments einer gegebenen Dokumentenklasse ansehen, das bestimmte Parameterwerte repräsentiert.

Zweitens teile ich den Zustand p des I/O-TS in eine Modus-Komponente und einen Rest auf: $p = (mode(p), rest(p))$.

Und drittens führe ich zu jedem Modus $mode(p)$ und jeder Dokumentenklasse $docCls(i)$ der Eingangsdokumente eine Testfunktion $cond_{mode(p),docCls(i)}$ ein, die testet, ob zusätzlich die Werte des Rest-Zustandes und der Parameter der Eingabe-Dokumente eine zu bestimmende Bedingung erfüllen oder nicht.

Damit lässt sich jede Transition eindeutig einer Äquivalenzklasse zuordnen, die durch 5 Angaben beschrieben wird: der Moduswert des Start- und Zielzustands, die Dokumentenklasse von Ein- und Ausgabe sowie eine Bedingung. In Anlehnung an die Notation $p \xrightarrow{i/o} q$ als Ersatz für $(i, o, p, q) \in \Delta$ schreiben wir die Äquivalenzklasse Δ_l auch als:

$$mode(p) \xrightarrow{docCls(i), cond_{mode(p),docCls(i)}(p_{rest}, param_i)/docCls(o)} mode(q) \qquad (2)$$

Auf diese Darstellung werde ich im Laufe der Artikelserie bei der Darstellung von Interaktionen zurückkommen.

7 Diskussion

Unser Verständnis der Informatik entwickelt sich in dem Spannungsfeld aus unserer Alltagsintuition und genuin informatischen Konzepten. Im günstigen Fall trägt die Informatik mit ihrem Fokus auf die Begrenztheit der Unterscheidbarkeit ihren Teil zur Weiterentwicklung unseres Denkens bei – im ungünstigen Fall vestellt sie uns den Blick auf das Wesentliche.

Die Leistungsfähigkeit einer Informatik, die sich auf das beschriebene Fundament der Systemmodelle, der Information, und der Komposition stützt, muss sich letztlich an der Frage messen, wie sie unser Verständnis unserer Welt verbessert.

Meiner Ansicht nach ergeben sich aus der in diesem Artikel entwickelten Sicht auf die Informatik in dieser Hinsicht drei wichtige unmittelbare Konsequenzen. Erstens gibt es keine zwei verschiedenen Welten der Informatik und der Physik, die auf geheimnisvolle Weise "wechselwirken" (entgegen etwa dem Modell in [8]). Sondern: Es gibt nur eine Welt, die wir unterschiedlich beschreiben. Das

ist insbesondere im Bereich der sogenannten cyberphysikalischen Systeme, aber auch für unsere Vorstellung der Mensch-Maschine-Interaktion zunehmend von ganz praktischer Bedeutung.

Zweitens lässt sich nun mit Klärung der Frage, was denn aus Sicht der Informatik eigentlich zwischen Systemen, die über Signale miteinander kommunizieren, "transportiert" wird – nämlich die Information – auch feststellen, was insbesondere *nicht* transportiert wird – nämlich irgendeine Bedeutung. Tatsächlich formuliert Claude E. Shannon in [27] explizit, dass eventuelle Bedeutungen der übertragenen Signale für seine Betrachtungen des Transports völlig "irrelevant" seien. Mit ihrem Konzept der Information, das im selben Atemzug auch deren Transport und ihre Verarbeitung unterscheiden lässt, legt uns die Informatik den Schlüssel zum Erfassen eines sehr leistungsfähigen Konzepts der Bedeutung ausgetauschter Zeichen in den Schoß.

Wenn wir im Rahmen der Informatik über die Bedeutung transportierter Informationen für die beteiligten Systeme sprechen wollen, etwa in dem Sinne, dass nicht alles was gleich unterscheidbar ist (also eine Information ist) auch gleich bedeutsam ist (also gleich relevant ist) kann diese Bedeutung ausschließlich durch ihre Verarbeitung entstehen. Damit können wir sagen, dass Bedeutung in einem informatischen Sinne "erteilt" wird. Für diese Vorstellung spricht auch, dass wohl jeder der Feststellung zustimmen würde, dass Information, die nicht verarbeitet wird, ganz sicher bedeutungslos ist, in dem Sinne, dass sie keine Wirkung entfalten wird. Mehr dazu dann im vierten Artikel dieser Serie.

Und drittens ergibt sich ein gewisser "virtueller" Charakter rein informatisch beschreibbarer Systeme. In dieser Beschreibung hatten wir ja gerade von der konkreten Realisierung ihrer Zustandswerte abstrahiert und für deren Bezeichnung stattdessen die informatischen Alphabete eingeführt. Damit lassen sich auch ganze informatische Systeme entsprechend einfach transportieren. Aber genauso endet ihr "virtueller" Charakter eben dort, wo die Abstraktion, dass die Struktur von Systemen in der Interaktion invariant bleibt und es allein auf die Unterscheidbarkeit ankommt, ihre Gültigkeit verliert. Wenn ich etwa auf die Nase falle, kann man die unterscheidbaren Zustände "Nase heile" und "Nase blutet" als eine Information ansehen, die man auch etwa mit einer Münze kodieren könnte (Kopf versus Zahl). Aber dieser Aspekt ist für mich dann (wahrscheinlich) nicht der wesentliche.

Hätten die Evangelisten der Objektorientierung (OO) in den 1990er Jahren Recht gehabt (z.B. [13]), wäre die Entstehung der Data Economy unverständlich – immerhin beruht der aktuelle Börsenwert (Jan. 2022) der 5 größten US-amerikanischen IT-Konzerne Google (Alphabet), Amazon, Facebook (Meta Platforms), Apple und Microsoft von ca. 9,5 Billionen US$ weitgehend auf dem wirtschaftlichen Wert von Daten – und nicht von Objekten im Sinne der OO. Tatsächlich ist die Festlegung der Verarbeitung von Informationen wie sie mit der Objektorientierung geschieht, unserem Modell der Semantik folgend, nichts anderes, als ihre Semantik festzulegen. Genau das vermeidet das vorgestellte Datenkonzept, in dem es einen Datentyp als eine Festlegung begreift, die sich durch die Konstruktuion einer weiteren berechenbaren Operation i.S. der

Berechenbarkeit nicht verändert.

Und hätte der von mir hochgeschätzte Leslie Lamport mit seiner Feststellung, "Computer science is largely about computation" [15] in jeder Hinsicht recht, dann könnte ich mir die weiteren 3 Artikel dieser Serie wahrscheinlich sparen und wäre stattdessen mit einem kurzen weiteren Abschnitt über Systemkomposition, die sich auf die Komposition berechenbarer Funktionen stützt, schon fertig. Aber: Informatik ist im Wesentlichen eine Theorie der Interaktion. Und ihr Konzept der Berechenbarkeit kommt nur dort zum Tragen, wo wir Systemfunktionen zusammensetzen. Wie ich im nächsten Teil dieser Reihe zeigen werde, ist das aber für eine große Klasse der Interaktionen nicht der Fall, nämlich für alle die Interaktionen, die nichtdeterministisch und zustandsbehaftet sind und die wir am besten mit Protokollen beschreiben. Mehr dazu im nächsten Artikel.

Literatur

[1] R. Alur. *Principles of Cyber-Physical Systems.* MIT Press, 2015.

[2] Bitkom. *Vorschlag zur systematischen Klassifikation von Interaktionen in Industrie 4.0 Systemen – Hinführung zu einem Referenzmodell für semantische Interoperabilität.* White paper, 2020.

[3] *Verwaltungsschale in der Praxis: Wie definiere ich Teilmodelle, beispielhafte Teilmodelle und Interaktion zwischen Verwaltungsschalen (Version 1.0).* Bundesministerium für Wirtschaft und Energie (BMWI), 2020.

[4] L. Cardelli and P. Wegner. On Understanding Types, Data Abstraction, and Polymorphism. *ACM Comput. Surv.*, 17(4):471–523, Dec. 1985.

[5] A. Church. An unsolvable problem of elementary number theory. *American Journal of Mathematics*, 58:345 – 363, 1936.

[6] A. Church. A formulation of a simple theory of types. *Journal of Symbolic Logic*, 5:56–68, 1940.

[7] O.-J. Dahl and K. Nygaard. Simula - an algol-based simulation language. *Commun. ACM*, 9(9):671–678, 1966.

[8] DIN. SPEC 91345:2016-04 Referenzarchitekturmodell Industrie 4.0 (RAMI4.0), 2016.

[9] G. Gabriel, H. Hermes, F. Kambartel, C. Thiel, A. Veraart, et al. Gottlob Frege - Wissenschaftlicher Briefwechsel. *Hamburg: Meiner Verlag*, 1976.

[10] R. V. Hartley. Transmission of information. *Bell Labs Technical Journal*, 7(3):535–563, 1928.

[11] J. Hey. The data, information, knowledge, wisdom chain: the metaphorical link. *Intergovernmental Oceanographic Commission*, 26:1–18, 2004.

[12] ITU-T. X.200 Information Technology - Open Systems Interconnection – Basic Reference Model, 1994.

[13] I. Jacobson. *Object Oriented Software Engineering: A Use Case Driven Approach*. Addison-Wesley Professional, 1 edition, 1992.

[14] S. Kleene. General recursive functions of natural numbers. *Mathematische Annalen*, 112(5):727–742, 1936.

[15] L. Lamport. Computer science and state machines. In *Concurrency, Compositionality, and Correctness*, pages 60–65. Springer, 2010.

[16] B. H. Liskov and J. M. Wing. A behavioral notion of subtyping. *ACM Trans. Program. Lang. Syst.*, 16(6):1811–1841, 1994.

[17] J. C. Mitchell. *Foundations for Programming Languages*. MIT Press, Cambridge, Massachusetts, 3 edition, 2000.

[18] D. L. Parnas, J. E. Shore, and D. Weiss. Abstract types defined as classes of variables. *ACM SIGPLAN Notices*, 11(SI):149–154, 1976.

[19] B. C. Pierce. *Types and programming languages*. MIT Press, Cambridge, MA, USA, 2002.

[20] J. Reich. Finite system composition and interaction. In K.-P. Fähnrich and B. Franczyk, editors, *40. GI Jahrestagung (2)*, volume 176 of *LNI*, page 603. GI, 2010.

[21] J. Reich. Composition, cooperation, and coordination of computational systems. *preprint arXiv:1602.07065*, 2016/2020/2021.

[22] J. Reich. Data. *preprint arXiv:1801.04992*, 2018.

[23] J. Reich. A theory of interaction semantics. *preprint arXiv:2007.06258*, 2020/2022.

[24] J. Reich. Komposition und Interoperabilität von IT-Systeme. *Informatik Spektrum*, 40:339–346, 2021.

[25] J. Reich, L. Zentarra, and J. Langer. Industrie 4.0 und das Konzept der Verwaltungsschale – eine kritische Auseinandersetzung. *HMD Praxis der Wirtschaftsinformatik*, pages 1–15, 2020.

[26] J. Sakarovitch. *Elements of Automata Theory*. Cambridge University Press, 2009.

[27] C. E. Shannon. A Mathematical Theory of Information. *Bell System Technical Journal*, 27:379–423, 623–656, 1948.

[28] C. E. Shannon. Communication in the Presence of Noise. *Proceedings of the IRE*, 37(1):10–21, 1949.

[29] A. Tolk, C. D. Turnitsa, S. Y. Diallo, and L. S. Winters. Composable M&S web services for net-centric applications. *The Journal of Defense Modeling and Simulation*, 3(1):27–44, 2006.

[30] S. Tripakis. Compositionality in the Science of System Design. *Proceeding of the IEEE*, 104:960 – 972, 2016.

[31] A. M. Turing. On Computable Numbers, With an Application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society, Series 2*, 42:230–265, 1936.

[32] C. Zins. Conceptual approaches for defining data, information, and knowledge. *Journal of the American Society for Information Science and Technology*, 58(4):479–493, 2007.