Können KI-Programme als Theorien betrachtet werden?

Klaus Manhart

Überarbeitete Fassung aus meiner Dissertation

„KI-Modelle in den Sozialwissenschaften“,
Oldenbourg-Verlag, München, 1995

www.klaus-manhart.de
mail@klaus-manhart.de

München, September 2007

1. Einleitung

Grundsätzlich können bei der Computermodellierung zwei Vorgehensweisen unterschieden werden: entweder man geht von (empirischen oder fikten) Daten aus oder von einer bereits existierenden, umgangssprachlich oder mathematisch formulierten Theorie.

Im letzten Fall würde das Computerprogramm die vorliegende Theorie „irgendwie" repräsentieren. Es soll nun die in der Literatur vielfach vertretene Auffassung untersucht werden, wonach das Programm direkt die Theorie *ist* oder die Theorie *verkörpert* - auch ohne ein verbales oder mathematisches Gegenstück.

Diese Position findet sich vor allem in der KI-orientierten und manchmal in der nicht-KI-orientierten Simulationsliteratur (KI = Künstliche Intelligenz). Wir möchten zunächst einige Autoren zitieren, welche diese Vorstellung mehr oder weniger explizit vertreten.

(Z1) Ostrom (1988) betrachtet in „Computer Simulation: The Third Symbol System" Computermodelle neben der natürlichen Sprache und der Mathematik als drittes Symbolsystem, in dem sich Theorien repräsentieren lassen. Ostrom behauptet, dass *jede* Theorie, die sich in einem der beiden anderen Systeme ausdrücken läßt, sich auch als Computerprogramm formalisieren läßt.

(Z2) Frijda (1967: 610) propagiert Computerprogramme als „unambigious formulations of a theory" und begründet dies mit der Präzision von Programmiersprachen.

(Z3) Die KI-Pioniere Simon und Newell (1956) argumentieren in einem frühen Artikel dafür, Theorien statt in verbaler oder mathematischer Form direkt als Computerprogramme zu formulieren.

(Z4) Weizenbaum (1978: 205-206) behauptet, dass „Modelle in Form von Computerprogrammen ebenfalls Theorien darstellen (zumindest verdienen einige Programme die Bezeichnung)".

(Z5) In einem der weitverbreitetsten KI-Lehrbücher findet man die folgende Bemerkung: „Occasionally after seeing what a program can do someone will ask for the theory behind it. Often the correct response is that the program *is* the theory" (Winston 1984: 12, Hervorhebung Winston).

(Z6) Ähnliche Anmerkungen finden sich gehäuft in Boden (1987), z.B. über ein psychologisches Neurosenmodell: „... the program represents a psychological theory of neurosis" (Boden 1987: 6).

(Z7) An anderer Stelle wird der dynamische Charakter von computerisierten Theorien betont: „A functioning program is a theory that is intended more as a movie of mind than a portrait of it..." (Boden 1987: 34).

(Z8) Newell und Simon sagen über das Potential einer Informationsverarbeitungs-Theorie menschlicher Problemlösung: „... the theory performs the tasks it explains. That is, a good information processing theory of a good human chess player can play good chess; a good theory of how humans create novels will create novels, a good theory of how children read will likewise read and understand“ (Newell/Simon 1972: 10-11).

(Z9) Ähnlich äußert sich Wessels (1984: 371): Eine Simulation ist „in dem Ausmaß eine angemessene psychologische Theorie, indem die Simulation menschliches Verhalten exakt abbildet“.

In der Regel werden diese Thesen ohne eine Begründung oder weitere Diskussion in den Raum gestellt. Explizit wird dieses Thema behandelt von Kobsa (1982) „On Regarding AI Programs as Theories“, Kobsa (1984) „What is Explainded by AI-Models“, Simon (1979) „Philosophical Objections to Programs as Theories“ und Manhart (1989) „Können AI-Programme als Theorien betrachtet werden?“.

Wenn Programme mit Theorien gleichgesetzt werden (können), drängen sich einige Fragen auf, z.B.:

1. Lässt sich, wie Ostrom ohne weitere Begründung behauptet, *jede* Theorie in ein Programm übersetzen (Z1)?
2. Wenn ja, ist das *ganze* Programm (Z5) oder sind nur Programmteile mit der Theorie *identisch*?
3. Sind Programmiersprachen überhaupt *geeignete Sprachmittel* zur Repräsentation von Theorien?
4. Was ist die Beziehung zwischen verbaler/mathematischer Theorie, Computermodell und Computertheorie (z.B. Z4)?
5. Ist I/O-Äquivalenz von empirischem System und Programm hinreichend für den theoretischen Status eines Programms (Z8, Z9)?

2. Von der Theorie zum Programm: Theoriegesteuertes Vorgehen

Wir unterscheiden im folgenden zwei grundsätzlich verschiedene Vorgehensweisen, die man als Top-Down- und Bottom-up-Ansatz bezeichnen kann. Im Top-Down-Ansatz geht man von einer vorliegenden Theorie aus und versucht, diese in ein lauffähiges Computerprogramm zu bringen. Beim Bottom-up-Ansatz schreibt man zunächst experimentelle Programme, die empirische Phänomene beschreiben, verfeinert diese, bis das Programm „funktioniert“ und kann unter Umständen das Ergebnis - so die Behauptung von zum Beispiel Z7, Z9 - als Theorie der abgebildeten Phänomene auffassen. Diese Dichotomie korrespondiert in etwa der Unterscheidung von Langley et al (1987) in „Data-Driven and Theory-Driven Science“. Das Ziel beider Vorgehensweisen ist die wissenschaftliche Entdeckung: bei der datengesteuerten Methode die Entdeckung einer Menge von

Regularitäten oder einer Theorie (Langley et al 1987: 23), beim theoriegesteuerten Vorgehen neue, überraschende Ableitungen aus der Theorie.

2.1 Maschinelle Theoriepräsentation

Die allgemein akzeptierte Turing-These behauptet, dass man jedes intuitiv effektive Verfahren mit einer Turing-Maschine realisieren kann und umgekehrt, dass alles, was auf der Turing-Maschine ausführbar ist, algorithmisch ist. Da im Prinzip jeder moderne Computer eine Turing-Maschine ist (Weizenbaum 1978: 93), folgt daraus, dass man eine Theorie dann auf einer Maschine ausführen kann, wenn es gelingt, diese in effektiver Form, also algorithmisch, zu formalisieren. Lindenberg (1971: 85) deutet die Turing-These salopp in der Form, dass uns der Computer zu Hilfe kommt, wenn wir statt in Propositionen in Algorithmen denken.

Die Antwort auf die Frage, ob eine Theorie in ein Computerprogramm übersetzbar ist, wäre also: ja, wenn es gelingt, einen Algorithmus für eine Theorie zu finden (die Theorie zu „algorithmisieren“) - wobei wir weiter unten diese Behauptung etwas differenzieren. Das Problem, das sich stellt, ist, ob es zu einer gegebenen - mathematischen oder verbalen - empirischen Theorie eine algorithmische Form *geben kann*. Da Theorien im Sinn des Standardtheorienkonzepts als Satzsysteme mit nomologischen Aussagen betrachtet werden, reduziert sich die Frage im wesentlichen darauf, ob Gesetze in algorithmischer Form dargestellt werden können und Programmteile entsprechend diese Gesetze repräsentieren.

Wie an anderer Stelle gezeigt, kann es für bestimmte Probleme keinen Algorithmus geben. Somit kann es auch Theorien geben, für die es nicht möglich ist, eine algorithmische Form anzugeben. Eine algorithmische Theorie, die irgendwelche Aussagen über natürliche Zahlen entgegennimmt und mit wahr oder falsch antwortet, kann nach dem Gödel-Theorem nicht existieren. In den Naturwissenschaften und in der Mathematik dürfte es viele Beispiele für Theorien geben, die nicht in Programme transformiert werden können. Die Aussage von Ostrom (Z1), nach der *jede* Theorie in ein Programm übersetzt werden kann, ist damit eindeutig falsch.

Beschränkt man sich auf human- und sozialwissenschaftliche Theorien, dürften die Theoreme der Berechenbarkeitstheorie weniger relevant sein. Statt dessen werden die zu bearbeitenden Probleme oft nicht verstanden und haben intuitiven Charakter. Ist das Verständnis jedoch hinreichend, müsste die Übersetzung gelingen. Die folgenden Projekte zeigen, dass derart in eine Programmiersprache transformierte Theorien *unmittelbar* als programmiersprachliche Darstellung der Theorie betrachtet werden.

Ein frühes Beispiel ist EPAM (Elementary Perceiver and Memorizer) von Feigenbaum (1963). EPAM modelliert die kognitiven Prozesse, die beim Lernen von Unsinnssilben vor sich gehen. Das Modell geht davon aus, dass das Lernen von Unsinnssilben ein komplexer Prozess der Symbolmanipulation ist, der in Form noch elementarerer

Symbolmanipulationen beschrieben und verstanden werden kann. Das Modell erklärt z.B., wie es möglich ist, dass wir längere Zeit etwas völlig vergessen und uns trotzdem später wieder daran erinnern können (Weizenbaum 1978: 218).

Diese Erklärung erfolgt mit Hilfe nomologisch-theoretischer – zum Teil statistisch formulierter - Prinzipien etwa folgender Art (Feigenbaum 1963: 299):
„Je ähnlicher Reizsilben einander sind, um so mehr Versuche sind zum Erlernen erforderlich“
oder:
„Wenn Assoziationen über eine Anzahl von Versuchen hinweg korrekt wiedergegeben werden, dann werden sie manchmal vergessen, um wieder aufzutreten und erneut zu verschwinden“.

Diese Prinzipien oder Gesetzmäßigkeiten finden sich im Programmcode und das Programm kann damit als eine Theorie darüber betrachtet werden, wie Menschen Nonsens-Silben lernen: „Wenn man es beispielsweise einem Psychologen gibt, der mit der Programmiersprache vertraut ist, in der sie geschrieben ist, so darf man erwarten, dass er sie verstehen wird. Was es jedoch zur Theorie macht, ist der Umstand, dass es bestimmte Prinzipien aufstellt, aus denen Konsequenzen gezogen werden können. Diese Prinzipien liegen in Form eines Computerprogramms vor, und ihre Konsequenzen lassen sich am Verhalten des Programms ablesen, das heißt, an der Art und Weise, wie der Computer das Programm liest“ (Weizenbaum 1978: 235-236).

Was das Programm „zur Theorie macht“, ist also das Vorliegen bestimmter Prinzipien oder *nomologischer Aussagen in Form des Programmcodes*. Colby (1973) bezieht sich in einem Modell, das einen paranoiden Akteur simuliert, direkt auf das nomologische Erklärungsschema der analytischen Wissenschaftstheorie:
„This model is considered a theoretical model because its inner structure embodies an explanatory account of the complex phenomena of paranoid communicative behavior. An explanatory account contains statements of lawlike generalisations. In order to explain concrete individual events, it also contains singular statements of individual initial conditions. In order to run and test the model, the general lawlike principles must be combined with the singular conditions to generate the I-O behavior of this individual hypothetical case“ (Colby 1973: 265-266).

Die von Colby postulierten Parallelen zwischen dem D-N-Erklärungsschema und dem Input-Output-Schema der Computersimulation lassen sich genauer wie folgt darstellen:

Theorie (D-N-Schema)	*Computermodell (I-O-Schema)*
Randbedingung	Input
Gesetz	Programm
----------L	==========A
Ereignis	Output

Den Randbedingungen entsprechen die Inputdaten, den Gesetzen das Computerprogramm und den aus Randbedingungen und Gesetzen folgenden Ereignissen die Outputdaten des Computers. Die zwei wesentlichen Aufgaben des D-N-Schemas sind Erklärung und Prognose. Im D-N-Schema gilt ein Ereignis dann als ***erklärt***, wenn es aus gesuchten Anfangsbedingungen und Gesetzen logisch deduziert werden kann. Dem entspricht auf Computerseite die „Erklärung" des Outputs durch „Herleitung" aus dem Programm und den Inputdaten. Umgekehrt - und für Computersimulation wichtiger - können im D-N-Schema Ereignisse aus gegebenen Randbedingungen und Gesetzen ***prognostiziert*** werden. Dies korrespondiert beim Computermodell der Erzeugung von Outputdaten aus Inputdaten und dem Programm, so dass das Computerprogramm als Vorhersageinstrument verwendet werden kann.

Dennoch sind D-N- und I-O-Schema gewöhnlich nicht völlig äquivalent. Während im D-N-Schema der Übergang vom Explanans zum Explanandum auf einem logischen Schluss basiert (symbolisiert durch ein L an der Übergangslinie), unterliegt dieser Überführung beim Computermodell normalerweise keine logische Deduktion, sondern sie ***ergibt sich aus dem Algorithmus des Programms*** (symbolisiert durch ein A an der Übergangslinie). Das einschränkende „normalerweise" ist deshalb nötig, weil es logische Programmiersprachen gibt, bei denen der Output unmittelbar als logische Deduktion aufgefasst werden kann.

Die Erzeugung des zu erklärenden oder prognostizierenden Ereignisses/Outputs findet aber bei beiden Schemata gewöhnlich auf unterschiedliche Weise statt. Diese Divergenz sollte man nicht aus den Augen verlieren, wenn man Computerprogramme als Theorien betrachtet.

Im metatheoretischen Rahmen des hier unterlegten Standardtheorienkonzepts macht die Aussage, dass ein Programm eine Theorie ist oder verkörpert, also Sinn, und wir verwenden für derartige Programme im folgenden auch die Bezeichnung „Computational Theories".

Weizenbaum (1978: 196ff.) sieht in der computerisierten Darstellung von Theorien eine völlig neue Beziehung zwischen Theorie und Modell: Theorien in Form von Computerprogrammen sind unter dem Gesichtspunkt der Sprache - abgesehen von einigen Besonderheiten - völlig gewöhnliche Theorien. Ein Physiker kann zum Beispiel seine Theorie über das Pendel entweder als Menge mathematischer Gleichungen oder als Computerprogramm formulieren. Das Programm hat aber den Vorteil, dass es nicht nur

von jedem verstanden werden kann, der in der Sprache ausgebildet ist, sondern darüber hinaus, dass es auch einen Computer durchlaufen kann. Da ein Modell den Verhaltensgesetzen einer Theorie genügt, *ist eine Theorie in Form eines Programms damit sowohl eine Theorie als auch - in einen Computer eingegeben und von diesem bearbeitet - ein Modell, auf das die Theorie Anwendung findet.*

In diesem Sinn interpretiert Weizenbaum die Aussage von Newell und Simon über ein Programm, das menschliches Problemlösungsverhalten nachbildet: „Die Theorie löst die Aufgaben, die sie erklärt“ (Z8). Eine Theorie kann nichts lösen, wohl aber ein Modell. In Computational Theories wird die Theorie - in der Interpretation von Weizenbaum - damit zum Modell und umgekehrt, so dass Theorie und Modell identisch werden.

2.2 Probleme maschineller Theorierepräsentation

Mit der Tatsache, dass die Theorie an die Bedingung der Ausführbarkeit auf Computern angepasst werden muss, sind einige pragmatische Besonderheiten und Probleme verknüpft.

Lindenberg (1971) weist darauf hin, dass in der Praxis Theorien selten unter dem Gesichtspunkt der Äquivalenz übersetzt werden.

- Erstens arbeitet man meist nicht mit der ganzen Originaltheorie, sondern begnügt sich nur mit bestimmten Ableitungen, deren Resultat für Testkomponenten anderer Sätze gebraucht wird. Auf diese Weise werden Ableitungen einzelner Sätze zu einem Prozessablauf zusammengefügt und die (Teil-) Theorie wird „prozessionalisiert“.

- Beispiele zeigen, dass zweitens oft zusätzliche Annahmen in das Computerprogramm eingeführt werden, die in der Originaltheorie nicht enthalten sind.

Damit haben wir die Situation, dass in der programmierten Theorie Elemente der Originaltheorie nicht enthalten sind und Elemente in Form von Zusatzannahmen hinzukommen. Dies lässt sich in folgendem Mengendiagramm veranschaulichen.

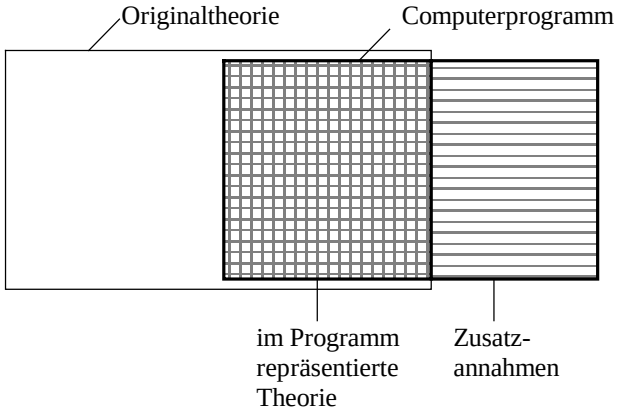

Abb. 1: Beziehung zwischen Originaltheorie und (in ein Computerprogramm) übersetzter Theorie

Die Umsetzung einer Theorie in ein Programm und insbesondere das Hinzufügen von Zusatzannahmen sollen an einem konkreten Beispiel verdeutlicht werden.

Homans (1961) stellt in „Social Behavior: Its Elementary Forms“ eine Theorie sozialen Verhaltens vor, welche Prinzipien der klassischen Ökonomie und behavioristischen Psychologie vereint. Soziales Handeln wird auf der Grundlage von Lerntheorie und Nutzenaustausch erklärt. Der Theorie und dem Modell unterliegen zwei zentrale Annahmen:

(1) Belohntes Verhalten wird häufiger gezeigt als nicht belohntes;
(2) Jede soziale Interaktion ist durch einen Austausch von Belohnungen gekennzeichnet.

Homans formuliert seine Theorie in fünf Hypothesen, und Gullahorn/Gullahorn (1963) haben die Propositionen aus Homans (1961) in ein Modell (HOMUNCULUS) übersetzt und - in der Terminologie Lindenbergs - „prozessionalisiert“.

Das Modell realisiert die Theorie mit einer einfachen Interaktionsfolge zwischen zwei Akteuren, die gemäß den Homansschen Thesen handeln. Ein Akteur wird dabei als hypothesentestender und informationsverarbeitender Organismus betrachtet, der Informationen empfangen, analysieren, rekonstruieren und speichern kann. Der Ablauf der Simulation gibt eine Folge von Interaktionen wieder, in denen ein Akteur A einen Akteur B um Hilfe bittet, Hilfe erhält, dafür Beifall und Respekt als Belohnung zollt, was die Interaktion verstärkt, bis der Helfende sie eventuell abbricht, weil sie ihn mehr kostet, als sie ihm an Belohnung einbringt.

Lindenberg (1971) zeigt, wie in das Modell Annahmen verschiedener Art eingeführt werden, die in der Originaltheorie nicht enthalten sind. Diese Annahmen, die wir im folgenden kurz betrachten wollen, sind bestimmte Randbedingungen, Operationalisierungen und Spezifizierungen.

Da Homans sich an der behavioristischen Lerntheorie orientiert, wurden *alle kognitiven Aspekte in die Randbedingungen geschoben.* Im Modell werden diese Aspekte aber explizit gemacht. Homans erste These lautet:

„Wenn das Verhalten in einer früheren Stimulussituation belohnt worden ist, dann wird diese Person jetzt dies oder ein ähnliches Verhalten um so wahrscheinlicher an den Tag legen, je mehr die frühere Stimulussituation der jetzigen ähnelt“ (Homans 1961: 53).

Gullahorn/Gullahorn (1963) fügen nun die Hypothese hinzu, dass die Person sowohl Stimuli als auch Reaktionen *generalisieren kann.* Diese Annahme ist nicht explizit in der These enthalten, sondern nur implizit („dies oder ein ähnliches Verhalten“, „je mehr die frühere Stimulussituation der jetzigen ähnelt“). In das Modell wird damit eine Gedächtnisstruktur eingeführt, die in der Originaltheorie nicht enthalten ist.

Ein zweiter Typ von Annahmen, die Gullahorn/Gullahorn machen, betrifft *Operationalisierungen*. Homans zweite These lautet:

„Je häufiger innerhalb einer gegebenen Zeitperiode jemand durch eine Handlung das Handeln eines anderen belohnt hat, um so öfter wird der andere die Handlung ausführen“ (Homans 1961: 54).

Eine einfache Operationalisierung bestünde im Zählen der belohnten Reaktionen, Gullahorn/Gullahorn lehnen dies aber als der menschlichen Informationsverarbeitung inadäquat ab. Sie postulieren einen viel gröberen Maßstab in Form einer Ordinalskala mit fünf Punkten von „fast immer belohnt“ zu „fast nie belohnt“. Im Modell wird also eine *Ad-hoc-Annahme* verwendet, die zwar plausibel erscheint, aber in der Theorie nicht vorkommt.

Die dritte Art von Annahme, die die Modellbauer einführen, ist *Spezifizierung*, was Lindenberg an der dritten Hypothese erläutert:

„Je mehr Nutzen jemand durch eine Handlung eines andern hat, um so häufiger wird er so handeln, dass er von anderen mit dieser Handlung belohnt wird“ (Homans 1961: 55).

Im Modell wird spezifiziert, *was* einer Person Nutzen bringt und *welche Verhaltenseinheiten* für ihn mehr Nutzen bringen als andere. Es werden Parameter geschaffen, die bei der Simulation mit eindeutigen Werten besetzt werden.

Lindenberg (1971) diskutiert die Beispiele so, wie wenn die Einführung dieser Annahmen eine *zwangsläufige* Folge der Computerimplementierung wäre. Die Einschleusung der Zusatzannahmen wird durchweg in dem Sinn gedeutet, dass *jede* in ein Computerprogramm prozessionalisierte Theorie viel mehr enthalten muss als das verbal oder mathematisch formulierte Gegenstück und in der Regel erheblich komplizierter wird als das Original. Dies ist jedoch nur bei einem bestimmten Implementierungstyp der Fall, den Gullahorn/Gullahorn gewählt haben und bei dem versucht wird, die Theorie möglichst weit in empirischen Daten zu verankern.

Unserer Ansicht nach ergibt sich aus den Beispielen eine ganz andere Folgerung. Sie belegen, dass sehr abstrakt formulierte Theorien um so schwieriger und aufwendiger zu implementieren sind, je mehr man versucht, sie empirisch zu verankern. Wenn man hingegen „theoretische“ Daten zulässt, gibt es die angesprochenen Probleme nicht, das heißt, es müssen weder Randbedingungen, Operationalisierungen noch Spezifizierungen eingeführt werden.

Lässt man als theoretisches Datum zum Beispiel zu:
Person x hat von Handlung y der Person z einen Nutzen u (formal: u(x,y,z)),

dann braucht nicht spezifiziert zu werden, was einer Person Nutzen bringt und Daten des Typs
u(x,y,z1) > u(x,y,z2)
können direkt im Bedingungsteil der oben genannten Hypothese verwendet werden. Je mehr man aber die Theorie empirisch verankern will, um so mehr Aufwand ist bei der Umsetzung in ein Programm nötig und um so mehr Zusatzannahmen müssen eingeführt werden.

Die Konsequenz dieses praktischen Beispiels ist, dass bei der Umsetzung einer Theorie in ein Programm verschiedene Implementierungsebenen unterscheidbar sind: Je abstrakter und höher die „theoretische Ebene", um so weniger werden die von Lindenberg angesprochenen Probleme relevant. Je konkreter und „empirischer" aber die Ebene, um so mehr treten diese Schwierigkeiten auf.

Ein zweites Problem maschineller Theoriepräsentation ist die unterschiedliche Wissensorganisation von Theorien und Programmen. Konventionelle Computerprogramme haben grundsätzlich einen *anderen* formalen Wissensaufbau als (qualitative) Theorien. Der hierarchische Aufbau von Computerprogrammen erzwingt, dass eine qualitative Theorie in eine algorithmische, abgestuft strukturierte Sequenz von Befehlen und Abfragen übersetzt werden muss.

Der Programmierer muss ferner einen schrittweisen Ablauf von vorzunehmenden Operationen festlegen (vgl. Puppe 1988: 3, Schnupp/Nguyen Huu 1987: 17). Dies steht in krassem Gegensatz zum Standardtheorienkonzept, nach dem eine Theorie im wesentlichen eine Menge von nomologischen Aussagen oder Propositionen ist und statisches, deklaratives Wissen enthält.

Während Algorithmen also prozedurales, algorithmisches Wissen enthalten, die spezifizieren, *wie etwas* in welcher Reihenfolge *getan* wird, enthalten Theorien propositionales Wissen, die Sachverhalte in Aussagesätzen codieren und ausdrücken, *was gelten* soll (vgl. die Homans-Hypothesen). Wenn Theorien in Algorithmen überführt werden müssen, wird der Modellierer damit gezwungen, das „Was" durch ein „Wie", die Proposition durch eine Prozedur auszudrücken.

In konventionellen Programmen kann diese „semantische Lücke" zwischen der verbal oder mathematisch formulierten Theorie und dem Computerprogramm eine erhebliche Behinderung sein. Der Ausdruck „semantische Lücke" bezieht sich dabei auf den Unterschied zwischen der ursprünglichen Repräsentation der Theorie und der programmierten Version (vgl. Merritt 1989: 3-4). Die semantische Lücke zwischen einem quantitativen Gesetz und der Repräsentation in einer numerischen Sprache wie FORTRAN ist gering. Eine Differentialgleichung kann zum Beispiel unmittelbar ohne wesentliche Änderung in die numerische Sprache abgebildet werden. Hingegen ist die semantische Lücke zwischen den Propositionen von Homans und einer konventionellen Programmiersprache erheblich.

Ein drittes Problem von Computational Theories ist die Trennung theoretisch relevanter von theoretisch nicht relevanten Prozeduren. Während eine Theorie, die in einer Programmiersprache ausgedrückt wird, theoretische Prozeduren enthalten muss, ist eine Vielzahl von Prozeduren eines Programms *irrelevant* für die Theorie. Da die Theorie auf der Maschine ausgeführt wird, werden Prozeduren für „Buchführungszwecke“ (Lindenberg 1971: 92) benötigt, auf die - bedingt durch die maschinelle Ausführung - nicht verzichtet werden kann.

In realen Programmen werden ganze Teilprozeduren für Zwecke gebraucht, die in keiner Relation zur Theorie stehen: „The low order subroutines and a number of technical necessities are determined by the particularities of the programming language, the mode of operation of the particular computer, and the special limitations inherent in serially operating digital machines“ (Frijda 1967: 611). Weiter muss es Prozeduren geben, die die Kommunikation mit der Außenwelt regeln, die Benutzerschnittstelle definieren oder Daten einlesen.

Es ist klar, dass diese Module nichts mit einer Theorie zu tun haben. Fasst man das ganze Computerprogramm als Modell auf, so ist die Abundanzmenge, anders als bei Nicht-Computermodellen, hier besonders groß. Da es theoretisch relevante und nicht relevante Programmteile gibt - die unter Umständen sogar vermischt werden - macht es wenig Sinn, die ganze Summe von Prozeduren als Theorie anzusehen und das Programm mit der Theorie gleichzusetzen. Das Programm *ist* also nicht die Theorie (vgl. Z5), vielmehr sollte das Programm besser als ***Repräsentation*** einer Theorie betrachtet werden (Frijda 1967: 611).

Ein Programm, das eine Theorie repräsentiert, enthält damit Teile der Originaltheorie plus eine (eventuell leere) Menge von Zusatzannahmen plus Teile, die nichts mit der Theorie zu tun haben. Das oben angegebene Mengendiagramm kann danach wie in Abb. 2 erweitert werden.

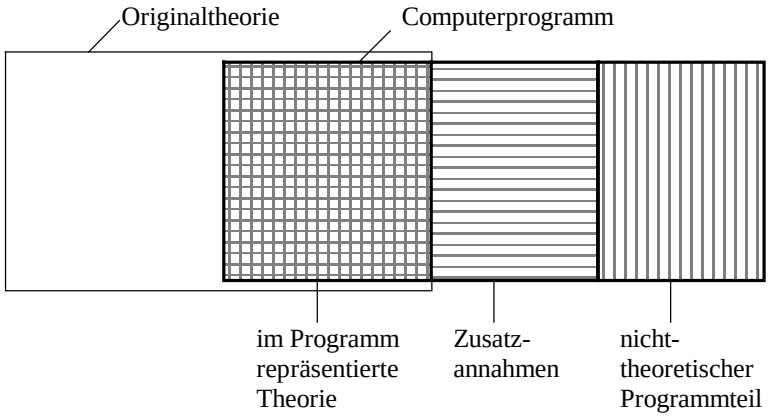

Abb. 2: Die Beziehung zwischen Originaltheorie und als Computerprogramm repräsentierter Theorie unter Berücksichtigung nicht-theoretischer Programmteile

Frijda (1967) fordert die strikte Trennung theorierelevanter von nicht theorierelevanten Prozeduren: „The theory-relevant routines must, so to speak, be isolated from theoretically irrelevant auxiliary operations and must be as independent of technical realizations as possible“ (Frida 1967: 612). Unglücklicherweise ist die Abschottung theorierelevanter Prozeduren von theoretisch irrelevanten Bearbeitungsprozeduren in konventionellen Sprachen nicht streng durchzuführen, da ***keine*** „klare Schnittstelle“ zwischen den theoretischen Programmteilen und den Problemlösungsstrategien, die die theoretischen Teile bearbeiten, existiert (vgl. Puppe 1988: 2-3).

Ungeachtet dieses Problems ist damit vieles im Programm keine Theorie und nichts deutet an, wann etwas zur Theorie gehört und wann nicht (Frijda 1967). Es liegt somit am Programmautor klar zu machen, welche Programmteile die Theorie verkörpern und wie und unter welchen Bedingungen diese funktionieren.

Damit kommen wir zu einer vierten und letzten Streitfrage bei Computational Theories: dem Kommunikationsproblem. Die meisten zeitgenössischen Autoren stehen auf dem Standpunkt, dass es grundsätzlich sinnvoll und nützlich ist, das Computerprogramm in der Wissenschaftlergemeinschaft zu kommunizieren und nicht nur die vom Modell produzierten Ergebnisse.

Die bloße informelle Programmbeschreibung war die in der Vergangenheit bevorzugte Kommunikationsform. In der Simulationsliteratur der sechziger und siebziger Jahre findet sich kaum Programmcode. Gullahorn/Gullahorn (1963 und 1965) zum Beispiel stellen ihr Programm informell vor anhand von Struktogrammen und verbalen Beschreibungen. Diese Präsentation unterscheidet sich in nichts von verbalen Theorien und der Vorteil der Präzisierung geht dadurch eindeutig verloren.

In der Vergangenheit gab es wichtige Gründe, auf die Publizierung von Programmtext zu verzichten: teure, inkompatible Maschinen an den Universitäten; nicht standardisierte und hardwarenahe Programmiersprachen; schwer lesbare, lange Programme. Viele Gründe für die Nichtveröffentlichung von Programmcode haben sich historisch überholt durch die Entwicklung von Hardware-Standards, strukturierter und normierter Programmiersprachen sowie nicht zuletzt veränderter Wissenschaftler-Sozialisation. Mehr und mehr Fachwissenschaftler beherrschen zumindest in Grundzügen eine Programmiersprache, Programme sind einfacher und leichter lesbar geworden, und Sprachwerkzeuge und Hardware sind allgemein verfügbar, standardisiert und billig.

Die Veröffentlichung des Programmtextes hat zwei entscheidende Vorteile:

- Der Programmcode kann gelesen werden und die Übersetzung der Theorie kann geprüft werden.
- Andere Wissenschaftler verwenden und modifizieren das Programm für Simulationen und Experimente, was die Weiterentwicklung der Theorien fördert.

Unter dem Aspekt der Lesbarkeit und Modifizierbarkeit sollten die Programme aber neben der Trennung theoretisch relevanter von theoretisch nicht relevanten Prozeduren möglichst *gut dokumentiert* und *informell beschrieben* sein. Ein bleibender Nachteil ist sicher die oft erhebliche Länge der Programmtexte von mehreren 1.000, 10.000 oder gar 100.000 Zeilen. Im Zeitalter des Internet ist die Übermittlung aber kein Problem mehr.

3. Vom Programm zur Theorie: Datengesteuertes Vorgehen

Wir sind bislang davon ausgegangen, dass eine *bereits vorliegende* - umgangssprachlich oder mathematisch formulierte - Theorie in ein Computerprogramm übersetzt wird.

Aus der Literatur lässt sich nun ein zweiter Ansatz rekonstruieren, der genau umgekehrt vorgeht: ausgehend von den Daten bzw. dem Input/Output(I/O)-Verhalten des betrachteten empirischen Systems werden Programme gebaut, die dieses möglichst gut nachahmen. Der Modellierer beschreibt sein Modell dabei idealerweise direkt in der Sprache des Simulationssystems (Möhring 1990: 23), das heißt, Modelle werden ohne - explizit - vorliegende Theorie gleich im Rahmen von Generate-and-Test-Programmen erstellt.

Mit „Generate-and-Test“ ist gemeint, dass Programmcode geschrieben wird, der versucht, das I/O-Verhalten des zu modellierenden Systems - zunächst grob - zu generieren. Mit dem generierten Code wird das I/O-Verhalten getestet und solange verbessert, bis das Verhalten dem empirischen System entspricht. Dieses Verfahren korrespondiert in etwa dem „rapid prototyping“ der Informatik: der Entwurf einer Lösung und ihr rasches Umsetzen in ein Programm (Lischka/Diederich 1987: 28).

Für bestimmte Problemstellungen wird also nicht explizit ein verbales oder mathematisches Modell formuliert, sondern diese werden gleich im Rahmen von Simulationsprogrammen angegangen. Das Ziel ist, durch das Erstellen der Programme theoretische Einsichten in die Funktionsweise des modellierten Gegenstands zu bekommen oder Gesetzmäßigkeiten zu entdecken. In Anlehnung an Ziegler (1972: 285) könnten wir dieses Vorgehen als den „Entwurf einer Theorie im Prozess der Formalisierung“ bezeichnen.[1]

Langley et al (1987) beschreiben dieses datengesteuerte Vorgehen in ihrem Buch „Scientific Discovery“ anhand von Problemlöseprozessen wie folgt:

„We can write in an information processing language - a symbolic rather than numerical language - a computer program that describes, say, the processes that intelligent adults are hypothesized to use in solving some class of problems. We can then present the program

[1] Die von uns vorgenommene Trennung - von der Theorie zum Programm und umgekehrt - ist natürlich etwas künstlich und findet in diesem Idealtypus gewöhnlich nicht statt. Beispielsweise können in Generate-and-Test-Modelle bestimmte Aspekte schon vorliegender Theorien integriert werden und umgekehrt können in Programme, welche Theorien repräsentieren, eigene Ideen des Modellierers einfließen.

and some human subjects with identical problems and compare their behaviors. From the computer we will obtain a trace, from the human subjects a verbal and written protocol of their behavior while they were solving the problems. We can then test the program - our theory of the behavior - by comparing the trace with the protocol, just as any theory is tested by comparing the system path it predicts with data showing the actual path“ (Langley et al 1987: 33).

Dieses Vorgehen ist typisch in der KI-basierten Modellierung. Ein Beispiel für ein solches Programm mit sozialwissenschaftlichem Bezug ist TEAMWORK (Doran 1985).[2] Das Modell untersucht, wie kooperative Aktivitäten von mehreren Akteuren koordiniert werden und simuliert hierzu ein Multi-Akteur-System, dessen Individuen eine gemeinsame Aufgabe bearbeiten müssen. Modelliert werden Planung und Ausführung von Aufgaben, sowie die Mechanismen von Kommunikation und Kooperation zwischen Akteuren. Explizites Ziel dieses Modells ist, ein tieferes Verständnis von realen Systemen zu bekommen „... both those that might be constructed and those human systems that are in existence around us“ (Doran 1985: 160).

Diese Fragestellungen würden Substanzwissenschaftler normalerweise durch Beobachtung und Experiment mit „natürlichen“ Akteuren untersuchen. In der KI wird einfach ein Computerprogramm gebaut, das die Ideen der Modellierer ausdrückt, und es wird geprüft, ob sich das Programm wie erwartet verhält (Gilbert/Heath 1985: 1). Tut es dies nicht, wird das Modell solange verbessert, bis es den Erwartungen der Modellierer genügt.

Programme, die erfolgreich arbeiten, werden dabei oftmals als Theorien und Erklärungen der modellierten Phänomene betrachtet, wie die Zitate Z5, Z7 und Z9 und die Aussage von Langley et al. („das Programm ist unsere Verhaltenstheorie“) belegen. Wir wollen uns nun mit dieser Sichtweise befassen, die insbesondere typisch für die Cognitive Science ist, einer Disziplin, die im wesentlichen ein Zusammenschluss von KI und kognitiver Psychologie ist.

3.1 Computational Theories in der Cognitive Science

Cognitive Science (Kognitionswissenschaft) ist ein interdisziplinäres Gebiet, dessen Ziel die Erklärung technischer ***und*** menschlicher intelligenter Systeme ist (Simon 1981). Die Wurzeln dieser Disziplin ruhen in der Erkenntnis, dass sich Psychologen, Linguisten, Informatiker und Neurologen ähnliche Fragen über den menschlichen Geist aus unterschiedlicher Sichtweise stellen. In der Regel versteht sich die Cognitive Science als theoretische Psychologie, in der der Computer als Instrument dient, theoretische Vorstellungen zu realisieren und das Verhalten dieser Realisationen wiederum als Phänomen zu untersuchen. Das Erstellen von Computermodellen ist dabei nicht Forschungsziel, sondern Forschungsmethode.

---

[2] Eine zusammenfassende Beschreibung des Modells findet sich in Manhart (1991).

Vorrangiger Untersuchungsgegenstand der Cognitive Science sind menschliche Mentalstrukturen, die auf einer mentalen Ebene erklärt werden ohne Rückgriff auf neuronale Zustände oder beobachtbares Verhalten. Bei der Analyse mentaler Prozesse bedient sich die Cognitive Science Konzepten der Computerwissenschaft: Der Mensch wird in Analogie zum Computer als Informationsverarbeitungssystem verstanden, der Informationen aufnimmt, verarbeitet und abgibt.

Die „Computermetapher" besagt, dass der Computer als Vorstellungshilfe bei der Beschreibung psychologischer Phänomene dienen kann, da die elementaren Prozesse, die der menschlichen Informationsverabeitung zugrunde liegen, denen bei Computern entsprechen. Mensch und Computer werden als Instanzen einer abstrakten Gattung symbolverarbeitender Systeme („physikalischer Symbolsysteme") angesehen, in denen sich dieselben theoretischen Strukturen realisieren. Postuliert wird eine fundamentale Gemeinsamkeit von Berechnung und Denken: „Wenn wir ein Buch lesen, nehmen wir visuelle Informationen - die gedruckten Buchstaben - auf und encodieren sie nach ihrer Bedeutung. Wir erinnern uns an viele Informationen, die wir aufgenommen haben, und können selbst über die Zeit hinweg gespeicherte Informationen betrachten. Beim Erinnern rufen wir vorher gelernte Informationen ab, und Vergessen kann als fehlerhaftes Abrufen aufgefasst werden. Man kann Menschen wie auch Computer als Systeme verstehen, die symbolische Informationen verarbeiten" (Wessels 1984: 39).

Die Cognitive Sciene geht aber weiter, als den Computer lediglich als Metapher für menschliche kognitive Prozesse zu benutzen. Ihr unterliegt die starke Annahme, dass ***jedes psychologische Phänomen durch ein effektives Verfahren erzeugbar*** (Boden 1988: 5)[3] und damit auf einem Computer ausführbar ist. Die Annahme ist insofern stark, als damit behauptet wird, dass alle psychischen Vorgänge zur Klasse der berechenbaren und - noch stärker - sogar zur Klasse der berechen- ***und*** durchführbaren Probleme gehören.

Das Ziel der Cognitive Science ist es, diese Algorithmen zu identifizieren und zu studieren. Sie versucht deshalb, Computerprogramme in Form von symbolverarbeitenden Algorithmen zu erstellen, die ein analoges Input-Output-Verhalten generieren wie die zu modellierenden menschlichen Phänomene. Um Einblick in die Funktionsweise menschlichen Denkens zu bekommen, wird versucht, dessen Funktionalität künstlich herzustellen. Kann menschliches Intelligenzverhalten Stück für Stück so simuliert werden, dass simuliertes und simulierendes Verhalten ununterscheidbar sind, so ist ein Beweis geglückt, dass Ergebnisse dieser Klasse von einem Computer berechnet werden können (Boden 1988: 7). Die Generierungsstrukturen der Modelle werden dabei nicht mehr als psychologisch irrelevant betrachtet, vielmehr lässt sich aus abstrakten Software-Strukturen von erfolgreichen Programmen unter Umständen auf Mental- und Verhaltensstrukturen bei Menschen schließen (Kobsa 1986: 113-114). In dieser Interpretation unterscheidet sich die

---

[3] Boden (1988) spricht nicht von Cognitive Science, sondern von Computational Psychology. Gemeint ist aber im wesentlichen dasselbe. In dem Buch von Boden sind Computational Theories der Cognitive Science detailliert beschrieben.

Cognitive Science nicht grundlegend von dem Modellierungsziel, wie es in anderen empirischen Wissenschaften formuliert ist.

In ihrer radikalsten Variante geht die Cognitive Science aber noch weiter. Die Behauptung der „starken Variante“ der Cognitive Science ist, dass die Programme ***nicht nur ein wirksames Werkzeug*** bei der Erforschung des menschlichen Geistes sind, sondern die menschlichen kognitiven Fähigkeiten ***vollständig erklären***. Der Erklärungsbegriff der Cognitive Science unterscheidet sich dabei ***fundamental*** vom deduktiv-nomologischen Erklärungsbegriff der analytischen Wissenschaftstheorie.

Während Erklärungen im Sinn des deduktiv-nomologischen Erklärungsschemas unter Rückgriff auf ***Gesetze*** erfolgen, beziehen sich Kognitionswissenschaftler mit ihrem Erklärungsbegriff auf das Wissen, ***wie*** ein Ding funktioniert, somit explizit auf das Zustandekommen von Verhalten und die ***Struktur*** der Entität, welches das zu erklärende Verhalten erzeugt. Sie verwerfen das Subsumtions-Modell der Erklärung von Hempel-Oppenheim als unbefriedigend und ersetzen es durch ein ***kausal-mechanistisches Erklärungsmodell***. Wir wollen diesen Erklärungsbegriff kurz erläutern.

Computerprogramme sind durch eine hierarchische Organisation gekennzeichnet, bei der komplexe Aufgaben auf immer einfachere Teilaufgaben reduziert werden. Das Verhalten eines („intelligenten“) Programms kann durch Reduktion komplexer Aufgaben in einzelne („dümmere“) Teilaufgaben vollständig verstanden und damit - im Verständnis der Kognitionswissenschaftler - auch ***erklärt*** werden: „We posit subprocesses to explain the actions of processes, subsubprocesses to explain the actions of subprocesses, and so on, until we reach the level of elementary information processes. Though the action of the whole program may appear to require brilliance ... the processes into which it decomposes at the first level (the 'main' subroutines) require only moderate brightness. As we descend through the levels of decomposition in the explanation, the spark of intelligence required for the processes at each level gradually dims, until we reach the machine language instruction, which are easy to implement mechanically. The ghost is exorcized by gradually reducing it to simple formal operations as we elaborate the explanation“ (Stillings et al 1987: 313).

Das Verhalten eines Programms wird also dadurch erklärt, dass man sich auf die Teile, die dieses Phänomen generieren und das Zusammenwirken dieser Teile bezieht. Das Ganze wird damit sukzessive in einfache, fundamentale Funktionsteile zerlegt, die für sich keine intelligenten Verfahren durchführen. Erst das Zusammenwirken dieser Funktionsteile erzeugt intelligentes Verhalten.

Kognitionswissenschaftler fassen den menschlichen Geist als ein analoges Konstrukt wie Computerprogramme auf: der menschliche Geist kann erklärt werden durch die Auffassung einer Person als eines intelligenten Systems, das mit einer Reihe von jeweils weniger intelligenten Subsystemen ausgestattet ist. Jedes dieser Subsysteme nimmt bestimmte Funktionen wahr und deren Ausführung wird wiederum an weitere, noch

weniger intelligente Subsubsysteme usw. delegiert (Heyer 1988: 38). Da Prinzipien von Computersystemen mit denen des menschlichen Geistes vergleichbar sind, kann das, was das Verhalten des Computerprogramms erklärt, genauso zur Erklärung kognitiver Phänomene herangezogen werden. Die Bedingung für eine gültige Erklärung lautet nur, dass I/O-Verhalten und simulierte und simulierende Strukturen identisch sind:[4]
„... for an adequat simulation to be an adequate explanation it must be the case ***both*** that the behaviours available to the machine correspond to the behaviours available to the organism ***and*** that the processes whereby the machines produces behaviour simulate the processes whereby the organism does" (Fodor 1968: 136, Hervorhebung Fodor).

Die Erklärung kognitiver Vorgänge ist damit äquivalent zur Generierung von Programmen, die ähnliches Verhalten künstlich erzeugen: „Thinking is to be explained by writing a program for a thinking process" (Newell/Simon 1971: 152). In diesem Sinn sind die Zitate (Z8) und (Z9) zu verstehen. Die Cognitive Science behauptet damit, dass es für die wissenschaftliche Erklärung intelligenten Verhaltens gleichgültig ist, was für ein Ding es ist, das sich intelligent verhält - Mensch oder Computer. Es gibt nur eine - uns derzeit noch unbekannte - Grundtheorie intelligenten Verhaltens, aus der sich für jeden Einzelfall eine Erklärung ableiten lässt. Jedem Intelligenzverhalten liegt eine fundamentale Struktur zugrunde, die mit einer Einheitstheorie der Intelligenz erklärt werden kann (Loeck 1986).

Damit erhebt die Cognitive Science den Anspruch, eine empirische Theorie menschlicher Intelligenz auf der Basis von Computerprogrammen zu liefern. Da die Programme, welche kognitive Funktionen nachbilden, diese gleichzeitig erklären, kann in der Auffassung der starken Variante der Cognitive Science jedes dieser Programme als Theorie über die modellierten Phänomene betrachtet werden. Newell und Simon argumentieren zum Beispiel, dass solche Programme in dem gleichen Sinn eine Theorie enthalten wie die Gleichungen von Newtons Dynamik eine Bewegungstheorie des Sonnensystems enthalten. Während die Differentialgleichungen von Newtons Theorie bestimmen, was als nächstes geschehen wird - in Abhängigkeit des exakten Systemzustands zu Anfang eines Intervalls - bestimmt das Programm, was als nächstes geschehen wird, in Abhängigkeit von seinem Zustand - der von der Geschichte und der Umgebung abhängt (Newell/Simon 1971).

Haugeland (1981: 2) fordert, dass eine Theorie der „natürlichen" Intelligenz die gleiche Basisform haben sollte wie eine Theorie, die intelligente Computersysteme erklärt. Der normalen Psychologie würde dann lediglich die Aufgabe zufallen, die von der Cognitive Science akzeptierten Konstrukte auf ihre empirische Brauchbarkeit hin zu prüfen (Kobsa 1986: 116). Dies geht sogar soweit, dass nicht als Computerprogramm formalisierte Theorien von den Theoretikern der Cognitive Science als „folk psychology" abgelehnt werden (Boden 1988). Fodor (1980) behauptet, dass die Computational Psychology im Sinn der Cognitive Science die einzige theoretische Psychologie ist, die wir jemals hoffen können zu erreichen.

---

4 Bei anderen Autoren genügt es bereits, wenn nur das I/O-Verhalten identisch ist. Vgl. (Z9).

Die meisten Kognitionswissenschaftler verstehen die Cognitive Science damit als eine Wissenschaft, die den Wissenschaftsbegriff der analytischen Wissenschaftstheorie sprengt. Kognitionswissenschaftler suchen nach wissenschaftlichen Erklärungen, die nicht-nomologische, kausal-mechanistische Erklärungen sein sollen und wollen auch Explananda, die eindeutig Gegenstand der Naturwissenschaft sind, auf eine andere, nicht-nomologische Weise erklären (Loeck 1986: 16). Stegmüller (1983: 482) hegt die Vermutung, dass es sich bei der Cognitive Science möglicherweise um eine Wissenschaft *sui generis* handle, also einer Wissenschaft eigener, neuer Sorte, die sich dem nomologischen Erklärungsschema der analytischen Wissenschaftstheorie entzieht. Die Diskussion um den erkenntnis- und wissenschaftstheoretischen Status der Cognitive Science ist seit ihrem Entstehen in vollem Gang und nimmt in der - eher kognitionswissenschaftlichen als wissenschaftsphilosophischen - Fachliteratur einen breiten Raum ein.[5]

3.2 Einwände gegen die Computational Theories der Cognitive Science

Ohne dass hier tiefer auf wissenschaftsphilosophische Fragen der Cognitive Science eingegangen oder die Frage nach dem wissenschaftstheoretischen Status der Disziplin gar entschieden werden kann, sollen zwei Einwände gegen die Theorieauffassung der Cognitive Science angerissen werden.

In ihrem Aufsatz „Ist Simulation Erklärung? Cognitive Science wissenschaftstheoretisch betrachtet“ untersucht Loeck (1986) die Frage, ob die Annahme der Cognitive Science gerechtfertigt ist, nach der die Theorie der natürlichen Intelligenz des Menschen grundsätzlich mit der Theorie identisch sei, die das intelligente Verhalten der technisch produzierten Computersysteme erklärt. Etwas präziser und allgemeiner lässt sich diese Behauptung folgendermaßen formulieren (Loeck 1986: 18):

Wenn ein Verhalten V_i eines Dings A durch ein Verhalten V_i* eines Dings B ununterscheidbar (=perfekt) simuliert wird, dann impliziert die Theorie T_i*, die das simulierende Verhalten V_i* unter Bezug auf B erklärt, diejenige Theorie T_i, die das simulierte Verhalten V_i unter Bezug auf A erklärt.

Am Beispiel der Theorie des natürlichen Ferro- und künstlichen Elektro-Magnetismus weist Loeck nach, dass beide Phänomene *nicht durch eine einheitliche Theorie erklärt* werden können. Wie in der Cognitive Science zeigen beide physikalische Phänomene ununterscheidbares Verhalten: jeder natürliche Magnet kann durch einen Elektromagneten simuliert werden. Die Theorie des natürlichen Magnetismus muss jedoch Begriffsentitäten verwenden, die in der Theorie des künstlichen Magnetismus nicht enthalten sind, so dass zwar die Theorie des künstlichen Magnetismus von der Theorie des natürlichen Magnetismus impliziert wird, aber nicht umgekehrt.

---

[5] Vgl. z.B. Dennett (1978), Pylyshyn (1980) und Zeitschriften wie „The Behavioral and Brain Sciences“, „Cognitive Science“ und „Cognitive Psychology“.

Die Simulation liefert damit nicht die Erklärung des simulierten Systems. Das simulierte Verhalten V und das perfekt simulierende Verhalten V* können nicht durch dieselbe Theorie (der Magnetfelderzeugung) erklärt werden. Ein Beispiel ist zwar kein Beweis und keine unmittelbare Widerlegung der Simulation-ist-Erklärung-Annahme, es spricht aber zumindest ein Präzedenzfall aus der Physik dagegen, in dem diese Annahme nachweisbar falsch ist. Loeck sieht deshalb die Behauptung der Cognitive Science als eine unhaltbare Spekulation an.[6]

Kobsa (1982 und 1984) wendet sich gegen die Auffassung, Programme im Umfeld der Cognitive Science als Theorien anzusehen und verweist auf den unterschiedlichen formalen Aufbau von Theorien und Computerprogrammen. Computerwissenschaftler und Kognitionswissenschaftler beziehen sich bei der Erklärung der Fähigkeiten von Programmen auf die Funktionen dieser Module und bedienen sich eines funktionalanalytischen Vokabulars. Ein Computermodell ist eine funktionale Dekomposition: es kann in verschiedene Teile mit spezifischen Funktionen aufgeteilt werden, deren Zusammenwirken das Resultat erzeugt. Die einzelnen Teile können durch funktional äquivalente Teile ausgetauscht werden, so dass es verschiedene (unter Umständen unendlich viele), funktional äquivalente Rekonstruktionsmöglichkeiten gibt.

Ein Computerprogramm wäre dann lediglich eine Instantiierung irgendeiner solchen funktionalen Dekomposition und ist lediglich ein Beweis, dass es möglich ist, ein bestimmtes Verhalten durch effektive Verfahren zu rekonstruieren. Damit hat man aber nur ein künstliches Individuum konstruiert, ein Modell, nicht aber die Theorie selbst. Was in den Aussagen der Kognitionswissenschaftler verwechselt wird, sind die Begriffe „Theorie“ und „Modell“ (Kobsa 1982: 933). KI-Programme sind nicht Theorien *über* Strukturen, Mechanismen und Prozesse, sondern *sind* diese Strukturen, Mechanismen, Prozesse (Kobsa 1982: 934).

Kobsa stützt seine Argumentation durch Gegenbeispiele wie den mittelalterlichen Uhrwerkmacher, der durch das Bauen von immer komplizierteren Uhrwerkmechanismen niemals die zugrundeliegenden Newtonschen Gesetze entdecken würde. Ähnlich argumentiert T.W. Simon (1979: 238): „It is easy to envisage someone constructing a device, such as a model automobile, which corresponds in every relevant respect to the available behaviours and internal processes of an actual automobile and which nonetheless, cannot provide any universal laws governing the operation of the automobile. This construction might be good technology but not good science“. Die Autoren wollen damit illustrieren, dass durch das Bauen immer komplexerer Modelle niemals die Theorie entdeckt wird, welche diese Modelle beschreibt.[7]

---

[6] Man darf aber nicht übersehen, dass Loeck sich in ihrer Argumentation auf den traditionellen Erklärungsbegriff bezieht. Ändert man den Begriff der Erklärung, so kann die Behauptung „Simulation ist Erklärung“ trivial werden.

[7] Allerdings sind beide Beispiele schlecht gewählt, da es in keinem Fall eine naturwissenschaftliche Theorie gibt, die solche Systeme behandelt.

In unserem Verständnis sind die Einwände von Kobsa und Simon in Abhängigkeit von der wissenschaftsphilosophischen Position relativ einfach - zumindest prinzipiell - zu entscheiden:

- Angenommen, die Cognitive Science sei eine Wissenschaft sui generis im Stegmüllerschen Sinn - was sich allerdings erst noch erweisen müsste: Dann würden die Einwände nicht zutreffen, da die Cognitive Science nicht nach Gesetzen und Erklärungen im traditionellen Sinn sucht.

- Angenommen, die Cognitive Science hat nicht den Status einer Wissenschaft sui generis mit eigenem Erklärungscharakter und man ordnet Theorien der Cognitive Science dem traditionellen Standardtheorienkonzept zu. Dann sind die Einwände zumindest z.T. berechtigt, da die Programme der Cognitive Science nicht per se als Erklärungen und Theorien fungieren können. Ein Programm ist nicht zwangsläufig eine Theorie, nur weil es I/O-äquivalente Ausgaben erzeugt.

Aber auch in diesem Fall - wenn man als Kriterium das Vorliegen gesetzesartiger Regularitäten akzeptiert - ist die von Kobsa angesprochene Funktionssichtweise ***kein grundsätzlicher*** Einwand dagegen, dass Programme Theorien repräsentieren können.

Interpretiert im Sinn des Aussagenkonzepts ist vielmehr entscheidend, ob sich Teile als gesetzesartige Regularitäten identifizieren lassen, die in einer relevanten Beziehung zum empirischen System stehen und aus denen sich eine empirische Theorie rekonstruieren läßt. Theoretische Strukturen in Form nomologischer Aussagen können sich experimentell aus, mit und in den Programmen mit fortschreitender Entwicklung realisieren. Da es keinen ernsthaften Einwand gibt, aus einer bestehenden Theorie ein Computerprogramm zu machen und die Gesetze im Programmcode zu repräsentieren, muss es auch im umgekehrten Fall möglich sein - wenn keine explizite Ausgangstheorie vorliegt -, dass sich hier nomologische Prinzipien finden lassen, die ein Fachwissenschaftler als theoretische Aussagen zu den im Modell abgebildeten Phänomenen akzeptieren würde.

Unabhängig von der Frage, ob mit dem Programm wirklich eine Theorie vorliegt, können Programme zweifellos durch ständiges Experimentieren, Verbessern und Verfeinern theoretische Einsichten liefern und die Bildung von Theorien fördern, so dass auf der Basis von Computerprogrammen zumindest Theorien konstruiert werden können. In der kognitiven Psychologie wird die Theoriebildung beispielsweise entscheidend von Computermodellen angeregt. Theorien über Schemata oder propositionelle Repräsentationen verdanken ihr Entstehen im wesentlichen Computerprogrammen (vgl. Wessels 1984). Dieses Vorgehen ist sicherlich auch außerhalb der kognitiven Disziplinen anwendbar, wie das oben erwähnte Teamwork-Modell belegt.

3.3 Induktives Entdecken von Gesetzen

Wir möchten abschließend noch auf eine andere Programmklasse hinweisen, die diesem Bottum-Up- oder datenbasierten Ansatz zugerechnet werden kann. Wir haben eben gesagt, dass sich theoretische Strukturen in und aus den Programmen entwickeln können. Diese bewusst vage Sprechweise hat in der KI ein präzises Äquivalent. Unter dem Label „Maschinelles Lernen“ gibt es Programme, die (natur-)wissenschaftliche Gesetze induktiv aus Daten ableiten können (z.B. Langley et al 1987, ein Überblick findet sich in Slezak 1989). Gibt man einem solchen Programm z.B. die von Robert Boyle um 1660 benutzten Rohdaten als Input, dann findet das Programm das entsprechende Gesetz von Boyle bezüglich Gasdruck und Volumen als Output (Slezak 1989).

Eine bekannte Familie von solchen Entdeckungsmodellen sind die BACON-Programme, deren Ergebnisse unter Titeln wie „Rediscovering Chemistry with the BACON System“ oder „Rediscovering Physics with BACON.3“ veröffentlicht wurden. In letzterem beschreibt Langley (1979) eine Programmversion, die empirische Gesetze der Physik unter Nutzung einiger weniger Heuristiken entdeckt: „These rules detect constancies and trends in data, and lead to the formulation of hypothesis and the definition of theoretical terms. BACON.3 represents data at varying levels of description, where the lowest have been directly observed and the highest correspond to hypotheses that explain everything so far observed“ (Langley 1979: 505). BACON.3 hat Versionen folgender Gesetze wiederentdeckt: das ideale Gasgesetz, Keplers drittes Gesetz, das Gesetz von Coulomb und Ohm und Galileis Pendelgesetz. Inzwischen gibt es - beginnend mit BACON.1 - eine ganze Reihe unterschiedlich leistungsfähiger Versionen.

Summarisch lassen sich diese Entdeckungsprogramme wie folgt beschreiben: sie nehmen als Input eine Menge von Beobachtungsdaten und versuchen eine Menge von allgemeinen Gesetzen zu formulieren, die diese Daten bündeln. Zur Entdeckung von Invarianten in empirischen Daten oder mathematischen Konzepten werden bestimmte Heuristiken und Problemlösetechniken der KI verwendet. All diesen Programmen unterliegt die Annahme, dass wissenschaftliches Entdecken eine Spezialform von Problemlösen ist (Langley et al 1987: 13).

Die Entdeckungsmechanismen sind weder beschränkt auf quantitative noch auf empirische Gesetze. Beispielsweise entdeckt GLAUBER qualitative Gesetze aus der Chemie über Säuren und Basen, wie z.B.: „Säuren reagieren mit Basen zu Salz“ (Langley et al. 1987). Ein anderes Programm, AM (Lenat, vgl. Slezak 1989), hat wichtige Konzepte der Zahlentheorie wieder gefunden, wie z.B. bestimmte Eigenschaften der natürlichen Zahlen, die vier arithmetischen Grundoperationen oder die Goldbachsche Vermutung, nach der jede gerade Zahl als Summe zweier Primzahlen dargestellt werden kann.

Die Möglichkeit der Implementierung solcher Programme zeigt, dass die Entdeckungsheuristiken nicht von spezifischen Eigenschaften des Problembereichs abhängen, sondern allgemein auf viele unterschiedliche Domänen anwendbar sind (Slezak 1989:

569). Die Programme deuten somit darauf hin, dass es formale Entdeckungsregeln und ein rationales Induktionsprinzip *gibt* (Slezak 1989: 567). Da der Prozess des Entdeckens von Theorien genauso rational zu rechtfertigen sei wie die Falsifikation, setzen sich Langley et al (1987) für die Möglichkeit einer psychologischen und normativen Theorie der Entdeckung ein. Dies kontrastiert zur Auffassung von Popper, nach der das Humesche Induktionsprinzip und damit die *Entdeckung* von Gesetzen und Theorien rational nicht zu rechtfertigen sei. Es bleibt abzuwarten, ob diese Programme Ausgangsbasis einer „Theorie der induktiven Entdeckung“ sein werden, welche die Ideen von Bacon und Mill präzisieren. In jedem Fall bestätigen die Programme die Nützlichkeit von datengesteuerten induktiven Entdeckungsmethoden..

LITERATUR

Boden, M. (1987). *Artificial Intelligence and Natural Man* (2.Aufl.). New York: Basic Books.

Boden, M. (1987). *Artificial Intelligence and Natural Man* (2.Aufl.). New York: Basic Books.

Colby, K.M. (1973). Simulations of Belief Systems. In R.Schank/K.M.Colby (eds.), *Computer Models of Thought and Language*. San Francisco: Freeman, S. 251-286.

Dennett, D.C. (1978). *Brainstorms. Philosophical Essays on Mind and Psychology*. Hassocks: Harvester.

Doran, J. (1985). The Computational Approach to Knowledge, Communication and Structure in Multi-Actor-Systems. In G.N.Gilbert/C.Heath (eds.), S.160-171.

Feigenbaum, E.A. (1963). The Simulation of Verbal Learning Behavior. In E.A.Feigenbaum/ J.Feldman (eds.), *Computers and Thought*. New York: McGraw-Hill, S.299-325.

Fodor, J.A. (1968). *Psychological Explanation*. New York: Random House.

Fodor, J.A. (1980). Methodological Solipsism Considered as a Research Strategy in Cognitive Psychology. *The Behavioral and Brain Sciences*, 3, S.63-110.

Frijda, N.H. (1967), Problems of Computer Simulation. In J.M.Dutton/W.H.Starbuck (eds.), *Computer Simulation of Human Behavior*. New York: Wiley, S.610-618.

Gilbert, G.N./Heath, C. (eds.) (1985). *Social Action and Artificial Intelligence*. Aldershot: Gower.

Gullahorn, J.T./Gullahorn, J.E. (1963). A Computer Model of Elementary Social Behavior. *Behavioral Science*, 8, S.354-362 (deutsch in: R.Mayntz (ed.), *Formalisierte Modelle in der Soziologie*. Neuwied/Berlin: Luchterhand, S. 233-248).

Haugeland, J. (ed.) (1981). *Mind Design. Philosophy, Psychology, Artificial Intelligence*. Cambridge, Mass: MIT Press.

Heyer, G. (1988). Geist, Verstehen und Verantwortung. Philosophische Grundlagen der Künstlichen Intelligenz Teil 1. *KI*, 1, S.36-40.

Homans, G.C. (1961). *Social Behaviour: Its Elementary Forms*. New York: Harcourt.

Kobsa, A. (1982). On Regarding AI Programs as Theories. In R.Trappl (ed.), *Cybernetics and Systems Research*. Amsterdam: North-Holland, S.933-935.

Kobsa, A. (1984). What is Explained by AI Models. *Communication & Cognition*, 17, 2/3, S.49-65.

Kobsa, A. (1986). Artificial Intelligence und Kognitive Psychologie. In J.Retti et al (eds.), *Artificial Intelligence. Eine Einführung* (2.Aufl.). Stuttgart: Teubner, S.105-130.

Langley, P. (1979). Rediscovering Physics with BACON.3. *Proceedings of the Sixth International Joint Conference on Artificial Intelligence* Vol.1, Tokyo, S.505-507.

Langley, P./Simon, H.A./Bradshaw, G.L./Zytkow, J.M. (1987). *Scientific Discovery: Computational Explorations of the Creative Processes*. Cambridge, Mass: MIT Press.

Langley, P./Simon, H.A./Bradshaw, G.L./Zytkow, J.M. (1987). *Scientific Discovery: Computational Explorations of the Creative Processes*. Cambridge, Mass: MIT Press.

Lindenberg, S. (1971). Simulation und Theoriebildung. In H.Albert (ed.), *Sozialtheorie und soziale Praxis*. Meisenheim: Hain, S.78-113.

Lischka, C./Diederich, J. (1987). Gegenstand und Methode der Kognitionswissenschaft. *GMD-Spiegel*, 2/3, S.21-32.

Loeck, G. (1986). Ist Simulation Erklärung? Cognitive Science - wissenschaftstheoretisch betrachtet. *Zeitschrift für allgemeine Wissenschaftstheorie*, 17, S.14-39.

Manhart, K. (1989). Können AI-Programme als Theorien betrachtet werden? Ein wissenschaftsphilosophischer Beitrag. In J.Retti/K.Leidlmeier (eds.), *Proceedings 5. Österreichische AI-Tagung*. Berlin: Springer, S.246-358.

Manhart, K. (1991). KI-Modellierung in den Sozialwissenschaften. *KI*, 2, S.32-40.

Merritt, D. (1989). *Building Expert Systems in Prolog*. Berlin: Springer.

Möhring, M. (1990). *Mimose. Eine funktionale Sprache zur Beschreibung und Simulation individuellen Verhaltens in interagierenden Populationen*. Dissertation, Universität Koblenz-Landau.

Newell, A./Simon, H.A. (1971). Simulation of Human Thought. In J.Dutton/W.H.Starbuck (eds.), *Computer Simulation of Human Behavior*. New York: Wiley, S.150-169.

Newell, A./Simon, H.A. (1972). *Human Problem Solving*. Englewood Cliffs, N.J.: Prentice-Hall.

Ostrom, T.M. (1988). Computer Simulation: The Third Symbol System. *Journal of Experimental Social Psychology*, 24, S.381-392.

Puppe, F. (1988). *Einführung in Expertensysteme*. Berlin: Springer.

Pylyshyn, Z.W. (1980). Computation and Cognition: Issues in the Foundations of Cognitive Science. *The Behavioral and Brain Sciences*, 3, S.111-139.

Schnupp, P./Nguyen Huu, C.T. (1987). *Expertensystem-Praktikum*. Berlin: Springer.

Simon, H.A. (1981). *The Sciences of the Artificial* (2.Aufl.). Cambridge, Mass: MIT Press.

Simon, H.A./Newell, A. (1956). Models: Their Uses and Limitations. In L.D.White (ed.), *The State of the Social Sciences*. Chicago: University Press. S.66-83.

Simon, T.W. (1979). Philosophical Objections to Programs as Theories. In M.Ringle (ed.), *Philosophical Perspectives in Artificial Intelligence*. Hassocks: Harvester, S.225-242.

Simon, T.W. (1979). Philosophical Objections to Programs as Theories. In M.Ringle (ed.), *Philosophical Perspectives in Artificial Intelligence*. Hassocks: Harvester, S.225-242.

Slezak, P. (1989). Scientific Discovery by Computer as Empirical Refutation of the Strong Programme. *Social Studies of Science*, 19, S.563-600.

Stegmüller, W. (1983). *Erklärung - Begründung - Kausalität* (Probleme und Resultate der Wissenschaftstheorie und Analytischen Philosophie, Band I. 2.Aufl.). Berlin: Springer.

Stillings, N.A. et al. (1987). *Cognitive Science. An Introduction*. Cambridge, Mass: MIT Press.

Weizenbaum, J. (1978). *Die Macht der Computer und die Ohnmacht der Vernunft*. Frankfurt a.M.: Suhrkamp (zuerst 1976: Computer Power and Human Reason. From Judgement to Calculation. San Francisco: Freeman).

Wessels, M.G. (1984). *Kognitive Psychologie*. New York: Harper & Row.

Winston, P.H. (1984). *Artificial Intelligence* (2.Aufl.). Reading, Mass: Addison-Wesley.

Ziegler, R. (1972). *Theorie und Modell. Der Beitrag der Formalisierung zur soziologischen Theoriebildung*. München: Oldenbourg.