Serjoscha Wiemer

Strategiespiele und das Medium-Werden des Computers. Computerschach als Faktor der Rekonzeptionalisierung des Computers als programmierbare Maschine zwischen 1945 und 1960

2014

https://doi.org/10.25969/mediarep/15016

Veröffentlichungsversion / published version
Sammelbandbeitrag / collection article

Empfohlene Zitierung / Suggested Citation:
Wiemer, Serjoscha: Strategiespiele und das Medium-Werden des Computers. Computerschach als Faktor der Rekonzeptionalisierung des Computers als programmierbare Maschine zwischen 1945 und 1960. In: Stefan Böhme, Rolf F. Nohr, Serjoscha Wiemer (Hg.): *Diskurse des strategischen Spiels. Medialität, Gouvernementalität, Topografie.* Münster: LIT 2014, S. 83–111. DOI: https://doi.org/10.25969/mediarep/15016.

Erstmalig hier erschienen / Initial publication here:
http://nuetzliche-bilder.de/bilder/wp-content/uploads/2020/10/B%C3%B6hme_Nohr_Wiemer_Strategie_Onlineausgabe.pdf

Serjoscha Wiemer

Strategiespiele und das Medium-Werden des Computers Computerschach als Faktor der Rekonzeptionalisierung des Computers als programmierbare Maschine zwischen 1945 und 1960

»Unser Problem besteht nun darin, wie die Maschinen für das Spiel zu programmieren sind.«
Alan Turing, *Computing Machinery and Intelligence*, 1950

Strategiespiele sind auf vielfältige Weise mit der Entwicklung des Computers verbunden. Während heute paradigmatische Strategiespiele als populäre Unterhaltungsform innerhalb der Computerspielindustrie aufgefasst werden können, zeigt sich in der Frühzeit der Computerentwicklung ein ganz anderes Bild. Als nach dem Zweiten Weltkrieg die Geschichte des Computers in eine neue Phase eintritt, gekennzeichnet durch eine Vielzahl an neuen Entwürfen für Computerdesign, durch erste kommerzielle Rechenanlagen und durch neue Anwendungsgebiete, treten die Rechenmaschinen aus dem engen Anwendungsfeld militärischer Nutzung (insbesondere in der Kryptografie) allmählich heraus, in dessen Anziehungskraft sie während des Zweiten Weltkriegs auf unterschiedliche Weise gestanden hatten. Genau zu jener Zeit lässt sich innerhalb der frühen Computerforschung ein zunehmendes Interesse am Schachspiel feststellen.
Ich werde im Folgenden anhand früher Texte von Alan Turing und Claude Shannon darstellen, in welchen Zusammenhängen das paradigmatische Strategiespiel Schach in den frühen Jahren der Computerrevolution an Bedeutung gewann und versuchen zu skizzieren, welche Zielvorstellungen, Ideen und Wunschvorstellungen dabei erkennbar werden. Wenn in diesem Text durchgehend vom ›Computer‹ die Rede ist, dann ist dieser Begriff nicht im heutigen Sinne zu verstehen. Der Begriff des Computers ist historisch nicht stabil, sondern in seiner Bedeutung abhängig von dem jeweiligen Stand der Entwicklung der Computertechnologie und ihrer sozialen und kulturellen Verwendung. Was jeweils als ›Computer‹ benannt und gedacht wird, ist in ständigem Fluss und dies trifft besonders auf jene Frühzeit der Computerentwicklung in den ersten Jahrzehnten nach dem Zweiten Weltkrieg zu, als noch keine geteil-

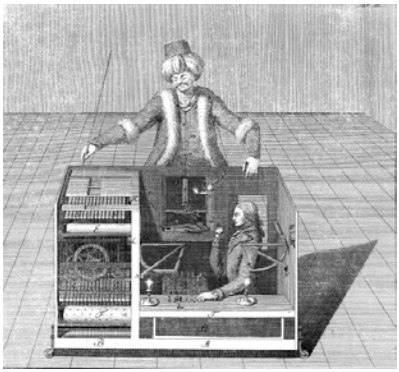

Abb.1: Das Innenleben des ›Schachtürken‹

ten Standards für den Aufbau der technischen Bestandteile, für die internen Funktionsweisen, für die Kopplung mit Ein- und Ausgabegeräten und für die Programmierung der Rechenmaschinen existierten.

Die Fragestellung, die damit verfolgt wird, ließe sich im Schnittfeld von Software Studies, Game Studies und Mediengeschichte/Medienarchäologie situieren. Sie geht von der Voraussetzung aus, dass Computer nicht von Beginn an als Medium anzusehen sind, sondern erst zum Medium ›gemacht‹ werden. Dies geschieht in einem langwierigen Prozess durch eine Reihe fortgesetzter Variationen, Erweiterungen und Re-Konzeptionalisierungen (vgl. Casey 2011).

Zu einer Zeit, als weltweit nur eine handvoll funktionsfähiger elektronischer Rechenmaschinen existierten, war es weder ausgemacht, ›was‹ ein Computer ist, noch welche seiner Eigenschaften für zukünftige Entwicklungen und Nutzungen bedeutsam sind. Es stand damals auf der Agenda, die ›Rechenknechte‹, die im Zweiten Weltkrieg ihren Dienst aufgenommen hatten, für neue Entwicklungsmöglichkeiten, Forschungsprojekte und Anwendungsfelder zu öffnen.

Hier bietet das Schachspiel nicht nur eine dezidierte Problemstellung für Ansätze der frühen Software-Entwicklung, sondern es ist zugleich in Argumentationen eingebettet, in denen es als idealisiertes Beispiel zur Formulierung bestimmter Problemstellungen prototypische Funktion erhält und mit Wunschvorstellungen aufgeladen wird, bezogen auf die projektierte ›Zukunft‹ des Computers. Die Perspektive der hier verfolgten Fragestellung richtet sich entsprechend also nicht allein auf eine Geschichte der Apparate und ihrer Software, sondern ebenso auf die sie begleitenden imaginären Konstellationen.

Schachspiel und Computergeschichte

Die Geschichte der Schachprogrammierung beginnt in vielen Darstellungen mit der Geschichte historischer Schachautomaten. Sie reicht damit mindestens zurück bis ins 18. Jahrhundert, als der ungarische Erfinder und Architekt Wolfgang von Kempelen einen schachspielenden Automaten konstruierte, der als ›Schachtürke‹ beziehungsweise ›mechanischer Türke‹ Berühmtheit erlangte.

Der Schachtürke war allerdings kein Gerät, das sinnvolle Schachzüge tätigen konnte, sondern ein Trickautomat: Mithilfe ausgeklügelter Mechanik konnte ein im Gerät verborgener menschlicher Schachspieler die Schachzüge einer türkisch gekleideten Puppe steuern. Dieser Apparat inspirierte nicht nur zahlreiche Nachahmungen, sondern hat vermutlich auch Charles Babbage, den Erfinder der Analytical Engine, zum Entwurf eines eigenen Schachautomaten angeregt.◂1 Als Erfinder des ersten tatsächlich funktionsfähigen Schachautomat gilt jedoch nicht Babbage, sondern der Spanier Torrès y Quévedo, der 1912 ein Gerät konstruierte, das in der Lage war, ein Endspiel mit Turm und Königen zu spielen.◂2

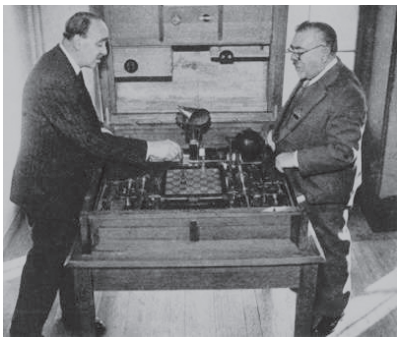

Abb.2: Gonzalo Torres y Quevedo führt Norbert Wiener den Schachautomaten El Ajedrecista vor (1951)

Die Programmierung von Rechnern für Schachanwendungen beginnt jedoch erst in den 1940er Jahren. Der erste Versuch eines Schachprogramms wird Konrad Zuse zugeschrieben. Zuse hat Schach unter anderem als eine Art ›Leistungstest‹ für die von ihm entwickelte Programmiersprache, das Plankalkül, aufgefasst und formulierte in dieser Sprache ein entsprechendes Programm.◂3 Alan Turings hat vermutlich bereits ab 1941 damit begonnen, Schachprogramme zu entwerfen.◂4 Der erste wissenschaftliche Artikel zur Programmierung eines Computers, um Schach zu spielen, stammt jedoch nicht von Turing, sondern von Claude Shannon. 1950 erschien sein grundlegender Aufsatz *Programming a Computer for playing chess* in der März-Ausgabe des Philosophical Magazine. Im gleichen Jahr veröffentlichte der britische Computerpionier Donald Davies in Penguin Science News *A Theory of Chess and Noughts and Crosses*, einen Artikel, der viele weitere Arbeiten zur Spielprogrammierung inspirierte.

Zwar waren auch andere Spiele für die Öffentlichkeit und für Mathematiker, die Zugriff auf Computer hatten, von Interesse, wie frühe Spielprogramme für Dame, Nim oder Noughts-and-Crosses (ein Spiel ähnlich Tic-Tac-Toe) belegen.◂5 Unter den verschiedenen frühen Spielprogrammen nimmt das Strategiespiel Schach jedoch eine Sonderstellung ein, ihm kommt im Computerdiskurs jener Zeit mit Abstand die größte Aufmerksamkeit zu. Zahlreiche der angesehensten Computerforscher widmen sich im Laufe der 1950er Jahre Problemen der Schachprogrammierung, setzen Experimente auf den damals fortschrittlichsten Geräten in Gang und veröffentlichen dutzende Forschungsberichte und Aufsätze zu Computerschach.◂6 Unter ihnen neben Turing und Shannon

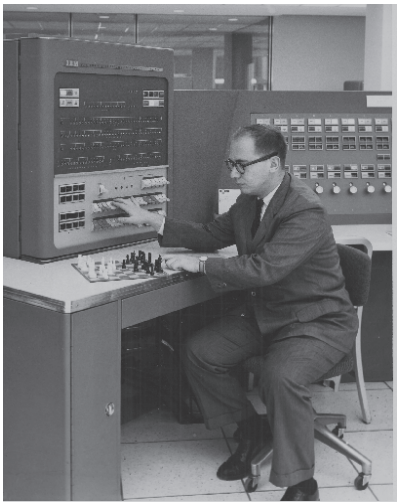

Abb.3: Alex Bernstein an der Konsole des IBM 704 demonstriert sein Schachprogramm (1958)

auch John von Neumann, Stanislaw Ulam sowie Allen Newell und Herbert Simon. Zu den bekanntesten frühen Spielprogrammen zählen Mate-in-Two von Dietrich Prinz, das 1951 auf einem Manchester Ferranti Mark I Schachprobleme vom Typ Matt in zwei Zügen berechnete, das Los Alamos Chess, von Stanislaw Ulam, Paul Stein, Mark Wells, J. Kister, W. Walden und John Pasta für den MANIAC I von 1956 sowie das 1957 von Alex Bernstein und Kollegen am MIT für einen IBM 704 programmierte Bernstein-Schach, das als das erste vollständige und funktionsfähige Schachprogramm angesehen wird.◂**7** Ab Mitte der 1960er Jahre hatten Schachprogramme dann eine Spielstärke erreicht, die sie auch gegen menschliche Spieler unter Turnierbedingungen Erfolge zeigen ließ.◂**8**

Was verbindet Schach und Computer?

Warum ist Schach jedoch innerhalb der frühen Computerforschung überhaupt von solchem Interesse? Warum geben sich unterschiedliche Forscher die Mühe, den Computer zum Spielen zu programmieren? Welchen Einfluss haben diese Experimente auf die Entwicklung des Computers und welche Dynamiken ergeben sich aus der Beschäftigung mit Schachprogrammierung? Diese Fragen sind gerade vor dem Hintergrund der frühen Computergeschichte bedeutsam, denn so wie zeitgenössische digitale Medien in erheblichem Maße durch eine hierarchische Zugriffsordnung gekennzeichnet sind, so war auch, wenngleich in viel größerer Rigidität, in den 1940er und 1950er Jahren der Zugriff auf Rechenanlagen durch strenge Hierarchien geregelt. Der Zugang zu Computern war ein seltenes Privileg, und umso bemerkenswerter ist es, dass dieses Privileg so früh dafür genutzt wurde, dem Computer das Spielen beizubringen. Zu fragen ist insbesondere auch nach den Gründen für die privilegierte Position, die dabei dem Schachspiel zugesprochen wird. Es muss davon ausgegangen werden, dass im Strategiespiel Schach sowohl praktische als auch theoretische Vorteile gesehen wurden, die bestimmten Vorstellungen und Bedürfnissen der Computerentwickler entgegenkamen. Neben forschungslogischen Vorausset-

zungen sind hierfür auch die kulturelle Stellung des Spiels sowie bestimmte mediale Eigenschaften von Schach relevant.

Claus Pias hat auf das Argument verwiesen, dass das Strategiespiel, namentlich das Schachspiel, »immer schon die Entstehungsgeschichte des Computers« (Pias 2000, 159) durchzogen habe. Er nimmt dabei Bezug auf einen Gedankengang Sybille Krämers, demzufolge der Gebrauch des Computers »weniger in der Perspektive instrumenteller Nutzung und mehr am Modell des spielerischen Umgangs zu konzipieren sei« (Krämer 1995, 226). Pias zufolge ist die enge Verbindung von Schach und Computergeschichte dadurch zu erklären, dass das Schachspiel als ein Mittel oder eine Hilfe angesehen wird, »den Computer selbst erst zu denken« (Pias 2000, 159).◂**9** Als Bekräftigung für diese These kann der Hinweis dienen, dass bereits Babbage überzeugt war, dass Schach auf seiner Analytical Engine implementierbar wäre, und dass sich schließlich im 20. Jahrhundert die prominentesten Figuren der Computergeschichte mit dem Strategiespiel Schach befassten.

Abb.4: Begeisterung für ›Elektronengehirne‹ in den 1950er Jahren in Deutschland

Die Bedeutung des Schachspiels stellt sich aus dieser Perspektive so dar, dass es historisch als Modell und als »Denkwerkzeug« für die Entwicklung des Computers und seiner Möglichkeiten diente. Eine solche Funktion des Schachspiels formuliert auch Konrad Zuse in seinen Erinnerungen, wenn er schreibt, er habe »nur deshalb Schach spielen gelernt«, weil er das Schachspiel für ein Modell hielt, am dem das Kalkül für seine Rechenmaschine entwickelt und erprobt werden könnte (Zuse 2010, 51).

Das Bemerkenswerte an Pias' These ist, dass entgegen der herkömmlichen Auffassung, derzufolge das Schachspiel als eine der möglichen *Anwendungen* des Computers aufgefasst wird, die nachträglich für die Maschine programmiert werden (wie jede andere Software oder jedes andere Computerspiel), das Schachspiel dem vorangestellt als Element in der *Konzeption* dessen gelten kann, was überhaupt ein Computer ist und sein kann. Das Strategiespiel Schach käme dann nicht als Implement nachträglich zum Computer hinzu, sondern wäre als Mittel anzusehen, den Computer überhaupt (als Medium) vorstellbar, denkbar oder modellierbar zu machen. Es wäre beinhahe so etwas wie ein ›Geburtshelfer‹ des Mediums Computer.

Mediale und kulturhistorische Voraussetzungen des Computerschachdiskurses

Auffallend ist, dass die meisten der frühen Forschungen zu Computerschach ihre Auswahl von Schach nicht mit dessen kultureller Tradition und Wertigkeit, sondern allein mit dessen internen Eigenschaften zu rechtfertigen versuchen: Schach sei ideal für Computerprogrammierung geeignet, da es ein einfach zu implementierendes Spiel sei, mit eindeutigen und leicht zu formalisierenden Regeln (Ensmenger 2011, 17). Dass ein Schachbrett als einfaches Feld von 8x8 Positionen dargestellt werden kann, gilt als Beleg für seine diskrete Natur, derzufolge es besonders kompatibel mit der »digitalen Natur« moderner elektronischer Rechner sei (ebd.,10). Dies trifft allerdings auch auf andere Spiele wie Nim und Dame zu, die sich ebenfalls auf relative einfache Weise intern von Computern repräsentieren lassen. Nathan Ensmenger hat darauf hingewiesen, dass die inhärenten strukturellen Merkmale von Schach allein nicht ausreichen, um den Vorzug gegenüber anderen Spielen zu erklären.◂10 Er argumentiert, dass es die spezifischen sozialen und materiellen Praxen sind, die kulturhistorisch mit Schach verknüpft wurden und sich um Schach herum ausgebildet haben, die für den Erfolg von Schach als Forschungsobjekt seit den 1950er Jahren verantwortlich sind (ebd., 17f.).

Das Schachspiel ist ein kulturelles Artefakt, das über Jahrhunderte mit unterschiedlichen Zuschreibungen und Bewertungen versehen worden ist. Die »Vorstellungen, die wir im zwanzigsten Jahrhundert mit dem Schachspiel verbinden – Notwendigkeit, Regelmäßigkeit und Berechnung«, sind keineswegs natürlich dem Spiel immanent, sondern müssen als diskursive Setzungen aufgefasst werden (Petschar 1986, 18). Wie Hans Petschar in seiner kultursemiotischen Studie zum Schachspiel gezeigt hat, entsteht die Vorstellung vom Schach als Spiel der Vernunft erst durch eine allmähliche ›Reinigung‹ des Schachspiels vom Zufälligen, Kontingenten, Akzidentellen. Ein Prozess, der sich kulturhistorisch über mehrere Jahrhunderte erstreckt. Gerade die kulturhistorische Variabilität des Schachspiels und seiner Zuschreibungen fordert dazu auf, auch die Einbettung der spielenden Subjekte – und Programmierer – in einem »größeren kulturellen Feld des Wissens und der Imagination« zu berücksichtigen (ebd., 46).

Erst nachdem es vom Zufall gereinigt war, konnte das Schachspiel allmählich in seine privilegierte Position als ›Spiel der Vernunft‹ einrücken. Dabei wird es charakteristisch auch von den Veränderungen erfasst, die den Begriff der Rationalität spätestens seit dem Ende des 18. Jahrhunderts mit Ideen von Eindeutigkeit, Berechenbarkeit und technischer Operationalität aufladen.◂11

Als eine frühe und entscheidende medienhistorische Voraussetzungen für die Transformationsgeschichte des Schachspiels kann die Entwicklung der Schachnotation im 13. Jahrhundert angesehen werden. Die Möglichkeit, Partien aufzuzeichnen, zu wiederholen und zu analysieren begünstigt die Standardisierung und Verbreitung der Spielregeln sowie die Aufwertung des Schachspiels innerhalb der mittelalterlichen Schriftkultur. Die Verschriftlichung des Schachspiels in der Schachnotation arbeitet zudem einer spezifischen Abstraktion des Spiels und des Spielverlaufs zu. Dass Schach aufschreibbar wird, bedeutet zugleich, dass es in eine symbolische Notation übertragen werden kann, abgelöst von konkreten Spielfeldern und Figuren – und von menschlichen Spielern.◂12 Neben seinem Nimbus als ›rationalem‹ Spiel und der Möglichkeit der ›Übersetzung‹ von Spiel in Schrift mittels der Partiennotation gibt es eine Reihe weiterer kultureller, struktureller und medialer Eigenschaften des Schachspiels, die seine Privilegierung gegenüber anderen Spielen im frühen Computerdiskurs zu verstehen helfen. Anders als beispielsweise Go war Schach ein in der europäischen und angloamerikanischen Kultur sehr populäres und verbreitetes Spiel. Weiterhin wurde es als ein Spiel attribuiert, das strategisches Denken, intellektuelle Kreativität und Planung voraussetzte und galt als besondere intellektuelle Herausforderung. Das Schachspiel hatte darum nicht nur praktische Vorzüge aufzuweisen, die es für die Implementierung als Computerprogramm geeignet sein schienen, sondern hatte auch einen erheblichen ›symbolischen‹ Wert.

Historisch gab es in den 1950ern bereits eine enge Verbindung zwischen Schach, Mathematik und Rechenmaschinen. Viele Mathematiker, die an Computerschach arbeiteten, waren selbst aktive Schachspieler wie Shannon, Turing oder Richard Greenblatt. Für die Computerindustrie in den USA der 1950er und 1960er Jahre waren Schachkenntnisse Ensmenger zufolge sogar ein Faktor bei der Personalauswahl: Schachkenntnisse wurden nicht nur auf Personalbögen verzeichnet, sondern waren mitunter auch direkt Bestandteil von Eignungstests. Grundlage hierfür war eine enge Assoziation von Schachkenntnissen mit Programmierfähigkeiten.◂13

Auch die für die Umsetzung als Spiel auf einem Rechenautomaten notwendigen Formalisierungen konnten von der kulturellen Tradition des Schachspiels profitieren. So gab es bereits eine umfangreiche Literatur zur Theorie und Geschichte von Schach, die seine Aneignung im Computerdiskurs erleichterte. Es existierten weitgehend vereinheitlichte und untereinander vergleichbare symbolische Notationen für Schachpartien sowie ein System für die Bewertung von Spielstärken. In den 1950er Jahren war u.a. das Harkness-System gebräuchlich, das später durch das Elo-System abgelöst wurde.◂14 Spielstärke war damit be-

reits ›quantifizierbar‹. Die Bewertung von Zügen, eine wichtige Voraussetzungen für entsprechende Evaluations-Funktionen in Computerschach, musste für die algorithmische Bearbeitung nicht von grundauf neu erfunden werden, sondern fand Vorläufer in einer bestehenden Schachpraxis, die ihrerseits über Methoden verfügte, um Positionen zu bewerten und Stellungen zu beurteilen (Ensmenger 2011, 18f.). Auch wenn diese Methoden, die etwa für Schachanalysen und Spieltraining zum Einsatz kamen, nicht vollständig quantifiziert waren, sind sie ein weiteres Beispiel dafür, wie die Kulturgeschichte des Schachspiels Voraussetzungen dafür schafft, dass Schach als »ideale experimentelle Technologie« (ebd.) im Computerdiskurs populär und erfolgreich werden konnte.
Darüber hinaus kann Schach innerhalb des Computerdiskurses nicht nur als Idealbild für die Verbindung von Rationalität und Spiel angesehen werden, sondern kann, im Sinne einer operationalisierten Wissensformation, insbesondere als Verbindungsstelle zwischen unterschiedlichen Diskursen und Diskurstypen fungieren:◂15 Nicht zuletzt war die Beschäftigung mit Computerschach auch ein geeignetes Mittel, um öffentliche Aufmerksamkeit für Ideen der Computerforschung zu wecken. Schach war ein Vehikel, um in der Öffentlichkeit Interesse für Computer und insbesondere für die Fortschritte der Computertechnologie und ihrer Programmierung zu gewinnen.◂16 Die kulturhistorische Sicht auf Schach ergibt ein differenziertes Bild unterschiedlicher, sich überlagernder Faktoren, die zur privilegierten Position von Schach innerhalb des Computerdiskurses beitragen. Wenn Pias als Erklärung für die beobachtbare Affinität zwischen Schach und dem Medium des Computers anführt, dass Schach »immer schon« mit der Entstehungsgeschichte des Computers verbunden sei, dass Schach ein Mittel sei, um den Computer »selbst erst zu denken« (Pias 2000, 159), sollte dies *nicht als Folge von internen, strukturellen, mathematischen oder philosophischen Eigenschaften des Schachspiels aufgefasst werden*. Vielmehr gilt es, die konkreten *historischen und diskursiven Voraussetzungen* zu bedenken, die das Schachspiel prägen und schließlich seine Attraktivität als »experimentelle Technologie« (Ensmenger 2012,6) im Schach-Computer-Diskurs stützen.

Klassische Positionen des Schach-Computer-Diskurses: Turing und Shannon

Welche Verwendungen Schach im Computerdiskurs erfährt und wie Schach dabei im Diskurs positioniert wird, um bestimmte (Re-)Konzeptionalisierungen des Computers zu forcieren, soll im Folgenden an zwei Beispielen erörtert werden. Gerade die frühen Arbeiten von Alan Turing und Claude Shannon

zu Computerschach scheinen dafür besonders geeignet, da sie als gleichsam ›klassische‹ Positionen angesehen werden können, die den Schach-Computer-Diskurs der 50er Jahre nachhaltig prägten. In ihrem jeweiligen Zugriff auf das Schachspiel zeigen Turing und Shannon durchaus Gemeinsamkeiten, es treten aber auch charakteristische Unterschiede zutage.
Es wird also darum gehen, auf welche Weise Schach bei Turing und Shannon für den Computerdiskurs aufgegriffen wird und welche spezifischen Ansätze und Leitvorstellungen im Hinblick auf das ›Medium‹ Computer dabei erkennbar werden. Die Begründungen dafür, warum man dem Computer Schach beibringen will, sind hier mindestens ebenso von Interesse wie die bloße Tatsache, dass ein Rechner zum Spielen genutzt werden soll. Die angeführten Begründungen sind dabei vor allem deshalb bedenkenswert, weil damit jeweils Aussagen innerhalb eines Computerdiskurses getroffen werden – Aussagen, die im Kontext der rasanten Innovationsdiskurse jener Jahre auch als Positionsbestimmungen darüber zu verstehen sind, wie diese neuen Rechenmaschinen zu konzeptionalisieren und zu verwenden sind.

Turing (1948): Zukünftige Maschinen, Probleme der Programmierbarkeit und Schach als Intelligenztest

Für Alan Turing ist die Idee des Schach spielenden Computers eng mit der Frage nach den Möglichkeiten intelligenter Maschinen verbunden.◂17 Die Frage nach ›Maschinenintelligenz‹ war in der Computerforschung der 50er und 60er Jahre Gegenstand zahlloser Veröffentlichungen. Angestoßen wurde die Diskussion dazu jedoch maßgeblich durch Turing, der die Frage »denkender Maschinen« immer wieder thematisierte. Auch wenn man, Hilary Putnam folgend, davon ausgeht, dass »die Idee des Geistes als eine Art Rechenmaschine zurück bis ins 17. Jahrhundert« reicht, so ist es doch Turing, durch den diese Idee eine entscheidende neue Form erhalten hat (Putnam 1996, 257). Nach Dotzler und Kittler setzt die Frage danach, ob eine Maschine denken kann, genau an dem »entscheidende[n] Unterschied« an, den Turings Entwurf einer Rechenmaschine von früheren Maschinen wie Leibniz *Lebendige Rechenbanck* oder Babbages *Analytical Engine* abhebt: der »Programmierbarkeit« (Kittler/Dotzler 1987, 226). Das Problem intelligenter Maschinen hatte Turing spätestens seit 1941 beschäftigt. Als er allerdings 1948 in seinem Text *Intelligent Machinery* die Frage ins Zentrum rückt, ob es möglich sei, dass Maschinen intelligentes Verhalten zeigen, nimmt er an entscheidender Stelle auf das Schachspiel Bezug. Bedeutsam ist zudem, dass Schach in *Intelligent Machinery* in einem

Kontext diskutiert wird, der von der Frage nach den *zukünftigen* Arbeitsbereichen der Rechenmaschinen bestimmt wird. So listet Turing, bevor er schließlich den Schachtest erläutert, verschiedene Aufgabenstellungen auf, die geeignet sein könnten, die Intelligenz der Maschine zu zeigen: »Wir stehen dann dem Problem gegenüber, angemessene Denkarbeiten [branches of thought] für die Maschine zu finden, in denen sie ihre Fähigkeiten ausüben kann«, so Turing (1948, 97). Er nennt die folgenden fünf Bereiche, die er für vielversprechend hält, damit Computer ihre ›Intelligenz‹ zeigen könnten: I) Spiele, II) Das Erlernen von Sprachen, III) Das Übersetzen von Sprachen, IV) Kryptografie, V) Mathematik. Auffällig ist bei dieser Auflistung, dass ausgerechnet Spiele an erster Stelle genannt werden. Obgleich Turing gleich mehrere Spiele beispielhaft auflistet, neben Schach sind dies Noughts-and-crosses, Bridge und Pokern, ist Schach das einzige Spiel, das in diesem Aufsatz weitere Beachtung erfährt und schließlich den Schlusspunkt von Turings Überlegungen bildet. Bemerkenswert ist dabei, auf welche Weise Turing das Schachspiel in seiner Argumentation verwendet: Es zeigt sich nämlich, dass in diesem frühen, von der medienwissenschaftlichen Forschung bisher wenig berücksichtigten Text, das Schachspiel als Vorwegnahme des später berühmt gewordenen Turing-Tests fungiert.

Die weitaus bekanntere, ›kanonische‹ Variante des Turing-Tests findet sich in Turings berühmt gewordenem Aufsatz *Computing Machinery and Intelligence*, der erstmals 1950 in MIND erschien. In diesem Text, der mit der Frage »Can a machine think?«, »Können Maschinen denken?« einsetzt, beschreibt Turing ein Imitationsspiel, das dazu dienen soll, die Intelligenz eines Computers in einem Spiel auf die Probe zu stellen. Diese kanonische Fassung des Turing-Tests ist als ein gender-orientiertes Frage- und Antwortspiel konzipiert: Ein Beobachter/Fragesteller (C) soll entscheiden, welche von zwei in einem anderen Raum befindlichen Personen (A) und (B) ein Mann bzw. eine Frau ist. Der Intelligenztest für die Maschine besteht in dieser Version darin, dass sie die Rolle von A übernehmen soll und ihre ›Intelligenz‹ dadurch unter Beweis stellen kann, wie gut sie das Antwortverhalten eines Menschen ›imitieren‹ kann.◂**18** Gelingt es dem Fragesteller nicht, die Maschine zu erkennen, ist damit deren ›intelligentes Verhalten‹ bewiesen.

1948 dagegen stellt Turing eine fast identische Versuchsanordnung vor. Der entscheidende Unterschied besteht darin, dass der Computer im Entwurf von *Intelligent Machinery* 1948 seine Intelligenz nicht dadurch beweist, dass er das ›Sprachspiel‹ beherrscht, und überzeugend ein menschliches Antwortverhalten imitiert, sondern dadurch, *dass er überzeugend Schach spielt*. Der Testaufbau ist dabei weitgehend analog zur späteren Variante des Turing-Tests (1948, 112f.): Gegeben seien in diesem Experiment drei Personen A, B, C, die gegenei-

nander Schach spielen sollen. Die Spieler befinden sich in zwei getrennten Räumen, um Sichtkontakt auszuschließen und die Schachzüge werden auf geeignete Weise zwischen den getrennten Räumen übermittelt. Spieler A und C sind Schachspieler auf niedrigem Niveau, Spieler B dagegen ist ein Computer, genauer gesagt eine Papiermaschine, d.h. ein Mensch in Computerfunktion, der exakt nur solche Anweisungen ausführt, die auch eine Rechenmaschine bewerkstelligen könnte. Der Testfall besteht nun darin, für Spieler C zu entscheiden, ob er in dem anderen Raum gegen einen Menschen (Person A) oder gegen die Papiermaschine spielt.

Turing berichtet, er selbst habe diese idealisierte Experiment durchgeführt und stellt als Effekt dieser Versuchsanordnung nicht nur heraus, dass es für C schwierig sein kann, zwischen Mensch und Maschine zu unterscheiden, sondern betont zudem deren emotionale Wirkung: Man erlebe das Gefühl, gegen etwas oder jemanden ›Lebendiges‹ zu spielen.◂**19** Mit der Erfahrung von intelligentem Verhalten kann also die eher ›intuitive‹ Erfahrung von Lebendigkeit einhergehen, wenn man Turings Schilderung an diesem Punkt folgt.

Fasst man die Rolle des Strategiespiels Schach in *Intelligent Machinery* zusammen, lassen sich mindestens drei unterschiedliche Funktionen erkennen:

1. Das Schachspiel ist zentraler Bestandteil einer Argumentation, die sich mit der Frage nach intelligenten Maschinen und mit der Zukunft des Computers befasst.

2. Spiele, und privilegiert Schach, werden als geeigneter Anwendungsbereich gesetzt, im dem Computer ihre Denkfähigkeit erkennbar ausüben können. Obwohl bestehende Maschinen noch nicht in der Lage sind, geeignete Schachprogramme auszuführen, kann ein solches Programm für Papiermaschinen entworfen und getestet werden. Die *Programmierung* der Maschine setzt sich dabei von der existierenden ›Hardware‹ ab.

3. Schach wird als Testfall oder Prüfanwendung für einen Intelligenztest respektive ein Imitationsspiel genommen. Die mögliche *Ununterscheidbarkeit* zwischen menschlichem Gegenspieler und Papiermaschine wird hier zum Prüfstein für Maschinenintelligenz – und weitergehend für die gefühlte ›Lebendigkeit‹ im Austausch mit einer (Papier-)Maschine.

Mechanischer Geist und reale Hardware

Dass Turing sich so intensiv mit dem Motiv der ›denkenden Maschine‹ auseinandersetzte, wird insbesondere vor dem Hintergrund seiner bahnbrechenden theoretischen Arbeiten zur ›Universalmaschine‹ plausibel. Deren prinzipielle Funktionsweise beruht darauf, Berechnungen in einfachste Schritte zu zerlegen und damit geistige Arbeit zu ›mechanisieren‹. Mit Bettina Heintz lässt sich Turings Konzept einer algorithmischen Maschine im »Rahmen der Rationalisierungsbewegung« des frühen 20. Jahrhunderts verstehen:

»Turing hat auf mentale Prozesse übertragen, was Taylor und Ford noch auf körperliche Bewegungen beschränkt hatten. […] Komplexe mentale Prozesse werden in einfache Grundoperationen zerlegt, in mentale Handgriffe sozusagen und dann gemäß der algorithmischen Vorschrift sequentiell aneinandergereiht. Das Fließband im Gehirn. […] Turing hat den Taylorismus zu Ende gedacht – alles läßt sich aufspalten und anschließend mechanisieren. Nicht bloß Bewegungen, auch das Denken.« (Heintz 1993, 172-174).

Computer konnten gegen Ende des Zweiten Weltkriegs Operationen ausführen, die zuvor nur durch menschliche Mathematiker(innen) geleistet werden konnten. Was ein Rechenautomat tat, war insbesondere in seiner Verwendung für numerische Aufgaben wie die Berechnung von Tabellen für die Ballistik direkt vergleichbar der Tätigkeit der »Computer«, jenen zumeist weiblichen Mathematiker(innen), die für wichtige Berechnungen zuständig waren und dabei Tischrechner, Tabellen und Stift und Papier zuhilfe nahmen. Insbesondere für die umfangreichen Berechnungen der Luftwaffe wurden solche Mathematiker(innen) in großer Zahl beschäftigt (vgl. Hagen 1994, 139f.). Was ein Rechenautomat tat, war beschreibbar als eine Abarbeitung einzelner Arbeitsschritte, die wohldefiniert waren und darum auch insofern ›geistig‹, als sie ein Äquivalent zur geistigen Arbeit der menschlichen ›Computer‹ darstellten.

Wenn Intelligenz zum »Service« wird, »der von Intellektuellen auf Maschinen übergeht«, wie Bernhard Dotzler und Friedrich Kittler zugespitzt formulieren (Kittler/Dotzler 1987, 211), wirft dies in der Folge die Frage auf, ob Maschinen nun »denken« können. Unterscheidet sich das Denken der Maschinen dann noch von dem menschlichen Denken? Wie lässt sich die Reichweite des maschinellen Denkens bestimmen und wo findet es eine Grenze? Turing greift das Schachspiel heraus, um im Rahmen einer Liste zukünftiger Forschungsfelder genau diese Fragestellung und damit die Möglichkeiten maschineller Intelligenz zu diskutieren. Schach wird zum Bestandteil einer Testanordnung, in der

die *Unterscheidbarkeit* von menschlicher und maschineller Intelligenz zur Disposition gestellt wird.
Als Turing 1947 seine Überlegungen zu denkenden Maschinen niederschrieb, gab es noch keine ernsthaft funktionsfähigen Schachprogramme. Das Computerschach stand noch am Anfang, sowohl Unzulänglichkeiten der Hardware, vor allem fehlende Speicherkapazitäten, als auch ungelöste Probleme der Programmierung setzten dem Projekt noch enge Grenzen. Einerseits waren die nötigen Berechnungen noch zu langsam in der maschinellen Ausführung, so dass die Suchtiefe, um Stellungen zu analysieren und geeignete Züge zu finden nicht ausreicht, andererseits war die Programmierung des Computers zu umständlich, wenn sie überhaupt möglich war. Turing konzipierte sein anvisiertes Schachprogramm darum auch nicht für damals existierende Rechenmaschinen, sondern für eine ›Papiermaschine‹, d.h. für die Berechnung mit Stift und Papier. *Als Problem der Programmierung war Schach den damals existierenden Maschinen um Jahre voraus*. Genau deshalb schien es aber auch geeignet, die *zukünftigen* Möglichkeiten und Herausforderungen des Computers auszuloten.

Der schachspielende Rechner bei Shannon

Nur wenige Jahre nach Turings Überlegungen zur Zukunft intelligenter Maschinen veröffentlicht Claude Shannon im März 1950 seinen später berühmten Artikel *Programming a computer for playing chess*. Es ist die erste theoretische Arbeit, in der dezidiert die Problematik der Schachspielprogrammierung ins Zentrum gestellt wird.
Shannon war nicht nur ein herausragender Mathematiker und Informationstheoretiker, dem zugeschrieben wird, »die erste Computerspielkonsole und die ersten portablen Computerspiele« entwickelt zu haben (Roch 2009, 24),◂**20** sondern auch ein passionierter Schachspieler, der 1949 mit Caissac einen der ersten Schachrechner überhaupt konstruierte. Shannons intensive Beschäftigung mit unterschiedlichen Spielmaschinen weist darauf hin, dass für ihn dem Spielen selbst ein eigenständiger Wert zukommt. Seine Spielapparate sind nicht nur als bloße Demonstrationsobjekte für technische Machbarkeit zu verstehen, sondern eröffnen alternative Zugänge und einen spielerischen Umgang mit Technik. In seiner grundlegenden Arbeit zu Computerschach geht es allerdings nicht um Spielen als ›play‹, sondern, streng zweckorientiert, um die Etablierung von Schach als Modellfall für die Programmierbarkeit von Computern.◂**21** Mit Blick auf die Möglichkeiten der modernen »general purpose

computer« formuliert Shannon das Ziel, neue Anwendungsmöglichkeiten des Computers zu erschließen, die über die »gewöhnliche Nutzung« für numerische Berechnungen in mehrfacher Hinsicht hinausgehen. (»extension over the ordinary use of numerical computers in several ways«) (Shannon 1950, 637). Seine wegweisende Idee besteht darin, das Schachspiel als Ausgangspunkt für weitere Grundlagenforschungen zu setzen. Er nennt mehrere Gründe, warum Schach ein »idealer« Ausgangspunkt sei, um die neuen Herausforderungen programmierbarer Computer anzugehen:

»1) Das Problem ist deutlich definiert, hinsichtlich der erlaubten Operationen (Züge) und im Ziel (Schachmatt); 2) Die Problemstellung ist weder zu einfach, um trivial zu sein, noch zu schwierig, um eine zufriedenstellende Lösung finden zu können; 3) Schach wird gemeinhin als ein Spiel aufgefasst, das ›Denken‹ erfordert, um gekonnt zu spielen; eine Lösung der Problemstellung zwingt uns entweder dazu, die Möglichkeit ›mechanisierten Denkens‹ anzuerkennen oder unser Konzept von ›Denken‹ weiter einzuschränken; 4) Die diskrete Struktur des Schachspiels passt hervorragend zur digitalen Verfasstheit (›digital nature‹) moderner Computer.« (ebd., 638, dt. d. Verf.).

Obwohl Schachprogrammen kein direkt praktischer Nutzen zu attestieren sei, können die in ihnen enthaltenen Aufgabenstellungen Shannon zufolge als modellhaft für allgemeinere logische Probleme, für den Entwurf von unterschiedlichen Algorithmen oder für neue Programmiertechniken gelten (ebd. 637). Schach wird von ihm in dieser Hinsicht konsequent als Programmierproblem begriffen. Als solches erfordert Schach, eine *Strategie* zu finden und diese in Algorithmen zu übersetzten. Dabei wird die Idee einer ›Strategie‹ in zweifacher Weise von Shannon in seiner Argumentation verwendet. Für die erste Bedeutung steht folgende Formulierung: »A strategy for chess may be described as a process for choosing a move in any given situation.« (ebd., 641). Dieser Begriff von Strategie ähnelt demjenigen, den von Neumann und Morgenstern 1944 in ihrer mathematisch-ökonomischen Spieltheorie einführten:

»Imagine now that each player [...] instead of making each decision as the necessity for it arises, makes up his mind in advance for all possible contingencies; i.e. that the player begins to play with a complete plan: a plan which specifies what choices he will make in every possible situation, for every possible actual information which he may possess at that moment in conformity with the pattern of information which the rules of the game provide him for that case. We call such a plan strategy« (von Neumann/Morgenstern 1990, 79).

Trotz der Ähnlichkeit zu diesem spieltheoretischen Strategiekonzept ist bei Shannon eine wesentliche Abwandlung erkennbar: Während die mathematische Spieltheorie Strategie als einen vorab kalkulierten »vollständigen Plan«

definiert, der jede mögliche Situation bereits umfasst, verschiebt Shannon den Schwerpunkt auf den *Prozess der Auswahl* eines Zuges in jeder gegebenen Situation. Statt eines vorab feststehenden Plans wird die Strategie an Situationen gebunden und Entscheidungen werden prozessual konzipiert.
Weil der Auswahlprozess geeigneter Spielzüge durch einen Computer getätigt werden soll, impliziert Shannons Begriff von Strategie zugleich Bedingungen der Programmierbarkeit, d.h. die algorithmische Formulierung dieses Auswahl- respektive Entscheidungsprozesses sowie dessen Übersetzung in programmierbare Funktionen. Entsprechend unterteilt Shannon die Aufgabe der Schachprogrammierung in drei Elemente:

»The problem of setting up a computer for playing chess can be divided into three parts: first, a code must be chosen, so that chess positions and the chess pieces can be represented as numbers; second, a strategy must be found for choosing the moves to be made; and third, this strategy must be translated into a sequence of elementary computer orders, or a programm.« (Shannon 1950b, 659).

Was Shannon hier beschreibt, klingt aus heutiger Sicht selbstverständlich und scheint kaum der Erwähnung wert. Vor dem Hintergrund der frühen 1950er Jahre geht es hier aber genau darum, ein neues Paradigma zu formulieren, das konsequent auf Programmierung abzielt. Darum führt Shannon mit der Re-Formulierung der Aufgabenstellung als *Programmier*aufgabe zugleich auch einen zweiten, komplementären Begriff von Strategie ein: Die Schachprogrammierung erfordert strategische Entscheidungen für das Programmdesign, weil festgelegt werden muss, auf welche Weise der Computer eine Auswahl zwischen möglichen Zügen im Spielverlauf treffen soll. Diese *Lösungsstrategien* zu wählen und auszuarbeiten obliegt dem Programmierer / der Programmiererin.

Programmierstrategien Typ A / Typ B

Die Einsicht in die Unmöglichkeit, alle möglichen Kombinationen oder theoretischen Zugfolgen in allen Varianten zu berechnen, bildet dabei den Ausgangspunkt für Shannons Analyse von Programmierstrategien: Geht man von einem durchschnittlichen Spiel mit etwa 40 Zügen aus, und legt dem eine durchschnittliche Zahl von rund 10^3 möglichen Zugvarianten (legalen Zügen) pro Zug zu Grunde, ergibt sich eine Zahl von 10^{120} möglichen Spielzügen pro Partie. Shannon führt diese Rechnung explizit vor, um zu unterstreichen, dass sich mit dem bloßen Durchrechnen von Zugvarianten kein funktionierendes Schachprogramm schreiben lässt. »Eine Maschine«, so Shannon, »die mit ei-

ner Geschwindigkeit von einer Mikrosekunde pro Zugvariante rechnen würde, bräuchte mehr als 10^{90} Jahre, um den ersten Zug zu berechnen!« (Shannon 1950, 641, dt. d. Verf.). Schachprogramme benötigen darum notwendig eigenständige Routinen zur Auswahl und Bewertung von Zügen. Alle möglichen legalen Züge vorab zu berechnen ist schlicht unmöglich.

Shannon zeigt schließlich in seiner Untersuchung zwei gegenläufige Typen von Strategien auf, die er schlicht als »Typ A« und »Typ B« bezeichnet. Eine Typ A Strategie zielt darauf, alle möglichen Zugfolgen bis zu einer bestimmten Suchtiefe zu berechnen und mittels einer Formel zur Bewertung der Züge zu evaluieren. Sie ähnelt einem Brute-Force-Ansatz, der zentral auf die Rechengeschwindigkeit des Computers setzt, um geeignete Züge zu finden. Der dazugehörige Algorithmus wird als Minimax-Algorithmus bezeichnet und bildet in verschiedenen Varianten bis heute den Kern von spielenden Programmen, insbesondere Computerschach (Ensmenger 2011). Als entscheidende Schwäche des Minimax-Algorithmus und der Typ A Strategie erkennt Shannon, dass es extrem ineffektiv wäre, alle möglichen Zugvarianten bis zu einer festgelegten Suchtiefe zu berechnen und zu bewerten, ohne zwischen vielversprechenden und sinnlosen Varianten zu unterscheiden. Eine Typ B Strategie soll demgegenüber zwei wesentliche Verbesserungen anstreben. Aus der Spielpraxis von menschlichen Schachmeistern sei bekannt, dass sie Zugvarianten nur extrem selektiv berücksichtigen. Entsprechend sollte eine Typ B Strategie einen Auswahlmechanismus beinhalten, der starke Varianten von schwächeren unterscheiden kann und nur geeignete Zugvarianten weiterverfolgt. Dadurch wird für vorteilversprechende Situationen zugleich eine größere Suchtiefe ermöglicht. Zudem müsste ein Prozess (Algorithmus) gefunden werden, um quasistabile Positionen auf dem Brett zu erkennen, und gezielt nach starken Fortsetzungen zu suchen. Statt alle Züge wahllos zu evaluieren, müssten taktische *Analysen* einer Position durchgeführt und der Maschine ein stärker selektives Verhalten implementiert werden.◂**22** Obwohl Shannon selbst recht deutlich zu erkennen gibt, dass er die Typ B Strategie für vielversprechender hält, ist der Minimax-Algorithmus historisch ungleich einflussreicher für die Entwicklung von Schachprogrammen geworden. Nathan Ensmenger führt dies darauf zurück, dass der »mimetische« Ansatz der Typ B Strategie (Ensmenger, 11), der sich an einer menschlichen Spielweise orientiert, größere Probleme in der Formalisierung aufwarf. Nötig wären komplexe Modelle menschlicher Entscheidungsprozesse, auf deren Grundlage dann überhaupt erst programmiertechnische Nachbildungen dazugehöriger kognitiver Prozesse erarbeitet werden könnten. Demgegenüber hat die Typ A Strategie den Vorteil, dass sie sich vergleichsweise einfach formalisieren lässt und dass sich darüber hinaus die Zugberech-

nungen und -evaluationen proportional zur Rechengeschwindigkeit des Computers steigern lassen. Ein Zuwachs an Spielstärke kann relativ einfach bereits dadurch erreicht werden, dass der Minimax-Algorithmus auf schnellerer Hardware läuft (vgl. Ensmenger, 12).◂23

Zwischen Geist und Software

Betrachtet man die frühen Ansätze von Turing und Shannon im Vergleich, zeigt sich, dass bei beiden das Strategiespiel Schach dazu dient, zukünftige Möglichkeiten und Anwendungsfelder des Computers zu thematisieren. Gerade in den ersten Jahren nach dem Zweiten Weltkrieg wird die Differenz zwischen den denkbaren Möglichkeiten und tatsächlichen Realisierungen eines ›general purpose‹ Computers sehr deutlich gesehen und gerade deshalb zum Forschungsprojekt erhoben.

Turing und Shannon waren entscheidend an der Entwicklung der ersten einsatzfähigen Rechenautomaten im Zweiten Weltkrieg beteiligt und unmittelbar mit deren Möglichkeiten und Limitierungen vertraut. Während des Krieges waren die Aufgabenfelder der Rechner und auch ihr grundlegendes Design auf militärische Anwendungen beschränkt. Shannon wie Turing waren jedoch überzeugt, dass Computer für andere Aufgaben als Kryptoanalyse oder die Berechnung ballistischer Tabellen zu verwenden seien. Ihre jeweiligen Beiträge zu intelligenten Maschinen und zur Programmierung von Computerschach sind darum auch Bestandteil eines Innovationsdiskurses und Teil der Transformationsgeschichte des Computers von militärischen zu zivilen Nutzungen, vom Rechenautomaten zum ›Universalrechner‹ und schließlich, weiter gefasst, zum Computer als ›Medium‹. In seiner theoretischen Konzeption als universelle Turingmaschine ist der Computer eine programmierbare Maschine, die in der Lage ist, alle denkbaren Maschinen als ›Programm‹ zu implementieren oder zu simulieren. Auch alle denkbaren Schachprogramme sind somit zumindest theoretisch in der Turingmaschine bereits angelegt. Diese theoretische Idee ist jedoch in der Praxis mit zahlreichen Schwierigkeiten und Unzulänglichkeiten konfrontiert, die der Programmierbarkeit Grenzen setzen. Nicht ohne Grund mussten Turing und David Champernowne ihre Entwürfe für Schachprogramme Ende der 1940er Jahre von Hand berechnen und Turing selbst dabei als »Papiermaschine« fungieren.

Turing wie Shannon geht es in den vorgestellten Arbeiten darum, Perspektiven für den Fortschritt der Computerentwicklung aufzuzeigen. Bemerkenswert ist, dass dabei einem ›Spiel‹ ein solch herausgehobener Stellenwert zukommt.

Dass es ausgerechnet das prototypische Strategiespiel Schach ist, ist dabei kein Zufall. Seine kulturelle Bedeutung als Strategiespiel macht einen wesentlichen Teil seiner Attraktivität aus: Schachspielen kann vor dem Hintergrund seiner kulturellen Attribuierung als Entsprechung für ›Denken‹ herangezogen werden, eng verknüpft mit einer spezifischen Vorstellung von Rationalität, ausgestattet mit Herausforderungen zur Problemanalyse, zur Bewertung von Situation und dem Treffen von Entscheidungen.

Darüber hinaus kann die Fokussierung auf das Strategiespiel Schach in gewissem Sinne als Gamification von weiterreichenden Forschungsprojekten angesehen werden. Es geht im Computerschach-Diskurs der frühen Jahre weniger darum, dass man mit oder gegen die Maschine ›spielen‹ will, also weniger um Vergnügen oder um eine Ludifizierung des Rechners, als darum Perspektiven für Grundlagenforschung zu formulieren.◂**24** Schach wird bei Turing ebenso wie bei Shannon als Testfall und Probe für Fortschritte in der Computerentwicklung konzipiert.

Trotz zahlreicher Gemeinsamkeiten markieren die Ansätze von Turing und Schach auch gegenläufige Pole des Computerdiskurses. Turings Interesse an der Möglichkeit ›denkender Maschinen‹ und einer ›Mechanisierung‹ des Denkens wird später von der Künstlichen-Intelligenz-Forschung aufgegriffen und zu einem ihrer Hauptanliegen. Die Frage denkender Maschinen, exemplarisch verdichtet in der Idee des schachspielenden Computers, fungiert darüber hinaus als Schnittpunkt zwischen philosophischen, computerwissenschaftlichen und populären Diskursen (vgl. Ensmenger 2011; Sprague 1972). Dies wird insbesondere vor dem Hintergrund der 1950er und 1960er Jahre virulent, als Computer im öffentlichen Verständnis noch als ›Giant Brains‹ und ›Elektronengehirne‹ galten.

Wenn Schach ein Mittel ist, um den Computer selbst zu denken, so ist es für Turing ein Mittel, um den Rechner als »denkende Maschine« zu begreifen. Bei Shannon dagegen geht es um eine andere Fragestellung. Er setzt das Schachspiel als Teil von Grundlagenforschung, weil es in der Programmierung bestimmte Lösungsstrategien erzwingt, die mit den praktischen Grenzen von ›number crushing‹ zu tun haben. Schachprogramme lassen sich, das exemplifiziert die berühmte Shannon-Zahl 10^{120}, nicht durch bloße Maximierung von Rechenleistungen zu erfolgreichem Spiel bringen.

In Absetzung vom numerischen Rechnen visiert Shannon eine ›höhere‹ Form von Denken an, als sie in der maschinellen Abarbeitung von eindeutig präskribierten Algorithmen gegeben ist. Entscheidend für die Frage nach dem Zusammenhang von Strategiespiel und dem Computer als ›Medium‹ ist dabei, dass Shannon die Programmierung von Spielen ins Zentrum von Grundlagenforschung

rückt. Insbesondere »Entscheidungsfähigkeit« (»something of the nature of judgment«) soll ›komputierbar‹ werden. Konkret geht es um eine Weiterentwicklung in der ›Kunst‹ der Programmierung angesichts der Herausforderung von Komplexität: »The engineers construct machines, and the mathematicians write the programs to use them.[…] Now let us take a look at the type of research the mathematicians are doing to improve the scope and use of computing machines.« (Shannon 1953, 692).

Abb.5: Programmierarbeit am ENIAC (1942-1955) - durch Neuverkabelung.

Shannons Artikel zur Programmierung von Computerschach ist dahingehend wegweisend, dass hier erstmals eine Programmierlösung – eine Software – als Probestein für die Computerentwicklung vorgestellt wird und eben nicht in erster Linie eine Verbesserung der durch Hardware zu erreichenden Rechengeschwindigkeit oder Speicherkapazitäten.◂**25** Schach als Strategiespiel bietet eine Problemstellung, die sich nicht auf »Rechnen« reduzieren lässt, und darum anspruchsvolle ›Strategien‹ der Programmierung erfordert.

Als Shannon 1950 seinen Artikel zu Computerschach veröffentlicht, setzt er das Wort »program« noch in Anführungszeichen. Bis Mitte der 1950er Jahre hatte sich der Begriff ›program‹ noch nicht in seiner heutigen Bedeutung etabliert (Chun 2008, 224f.). Frühe Rechenmaschinen mussten zumeist durch Neuverkabelung oder über Schalttafeln jeweils für einzelne Aufgaben konfiguriert werden.

Auch bei Rechnern mit Programmspeicher (›stored-program computer‹) war die Programmierung zunächst noch nicht integrierter Teil der Computerhardware, sondern eine Aufgabe, für die eigene Maschinen (Programmierkonsolen) erforderlich waren (Ceruzzi 2003). Die Formulierung von Programmen in Maschinencode war umständlich und eine Übertragung auf andere Hardware oft nicht möglich. Erst im Laufe der 1950er Jahre wurden nach und nach die ersten Compiler◂**26** und schließlich höhere Programmiersprachen entwickelt,◂**27** um gezielt das Programmieren für unterschiedliche Anwendungsfelder zu erleichtern.

Die Herausforderungen, die Shannon am Beispiel von Computerschach aufzeigt (Symbolverarbeitung, Problemlösen, Such-Algorithmen, Entscheidungsfindung) sind derart anspruchsvoll, dass sie unmittelbar von höheren Programmiersprachen profitieren. Schachprogramme sind somit auch als ›Geburtshelfer‹ für neue Ansätze zur Entwicklung von Programmdesign, Program-

Abb.6: Anspruchsvolle Programmierarbeit: Joneal Williams-Daw an der Programmierkonsole des UNIVAC (ab 1951), des ersten kommerziellen Computers der Firma Remington Rand

mierstilen und ›Strategien‹ der Programmierung zu verstehen, und schließlich nicht zuletzt von Programmiersprachen selbst.◂28

...mehr als nur ein Spiel

Als paradigmatisches ›Spiel des Computers‹ in den 1950er Jahren war das Strategiespiel Schach viel mehr als ›nur‹ ein Spiel. Es wurde zum wichtigen Bestandteil eines Innovationsdiskurses, der Programmierbarkeit mit der Idee ›denkender Maschinen‹ verband. Probleme von Voraussicht, Planung, Entscheidungshandeln, Programmierbarkeit und der Mechanisierung von ›Denken‹ wurden als Herausforderungen konzipiert, an denen sich der Fortschritt der Computerentwicklung messen lassen sollte. Das Strategiespiel Schach wurde dabei über unterschiedliche Denk- und Argumentationsfiguren auf komplexe Weise mit Fortschritts- und Innovationsdiskursen der Computerforschung verknüpft. Es ist somit Teil eines ›Projektgeschichte‹ des Computers, in welcher der ›number crasher‹ zur symbolverarbeitenden Maschine werden sollte – eine notwendige Voraussetzung und unhintergehbarer Zwischenschritt auf dem Weg vom Rechenautomaten zum ›Medium‹. Es nimmt dabei insofern eine entscheidende Stelle im frühen Computerdiskurs ein, als es eine zentrale Funktion für die Diskussion von Programmieransätzen und für die Re-Konzeption des ›Rechenautomaten‹ als programmierbare Maschine erhält. Als Problemstellung, Modellfall und rhetorische Trope im Diskurs der Computerforschung nach dem Zweiten Weltkrieg markiert Schach nicht weniger als den Beginn von ›Software‹.◂29

Aus medienhistorischer Sicht ist der Weg vom Rechenautomaten zum ›Medium‹ keine bloße Frage von Rechengeschwindigkeit und verfügbarem Speicher, sondern umfasst mindestens auch die Geschichte von Compilern, Programmiersprachen, Programmiertechniken, Ein- und Ausgabegeräten, erweiterten Anwendungsszenarien und fortgesetzter Forschung. Die Pointe von Schach in der frühen Computergeschichte besteht schließlich genau nicht darin, der Maschine ein bestimmtes Spiel beizubringen oder einen Vorläufer späterer Software-Spiele auf einem rechnenden Automaten zu implementieren, sondern es geht um eine Forschungs- und Entwicklungsarbeit, bei der Schach elemen-

tarer Bestandteil der Weiterentwicklung und Re-Konzeption eines Computers im Werden ist.

Anmerkungen

01► Vgl. für einen Überblick zu den Schachautomaten Ensmenger (2011, 8-12). Babbage (1864, 467) berichtete später, dass er auch für die niemals fertig gestellte Analytical Engine Ideen für ein Schachprogramm ausgearbeitet habe.

02► Eine spätere Version des elektromechanischen Geräts von 1914 ist noch heute funktionsfähig (Randell 1982, 333).

03► Allerdings wurde das Plankalkül erst in den 1970er Jahren implementiert, so dass Zuses Programm nur »theoretisch« lauffähig sein konnte. Vgl. Wikipedia (2013): Schachprogramm sowie Rojas (o.J.).

04► 1948 erwähnt er in einem Bericht an das National Physical Laboratory seine Erfahrung im Spiel gegen die ›Papiermaschine‹. Vgl. dazu Turings Schlussanmerkung in Turing (1948). Gemeinsam mit dem Ökonom und Mathematiker David Champernowne hatte Turing einige Schachroutinen unter dem Namen Turochamp zusammengestellt. Die erste überlieferte Partie von Turochamp, von Hand berechnet, fand im Jahr 1952 statt, eine Partie gegen Alick Glennie aus Manchester. Glennie wird später der Verdienst zugeschrieben, 1952 für den Mark 1 einen der ersten Compiler geschrieben zu haben, ein vereinfachtes »Codesystem« namens Autocode, ein Vorläufer späterer höhere Programmiersprachen.

05► So hatte der britische Computerforscher Christopher Strachey bereits 1952 den Code für ein funktionsfähiges Dame-Spielpgrogramm auf dem Ferranti Mark I geschrieben und erfolgreich getestet. Anfang der 1950er begeisterte eine von Ferranti gebaute Nimrod-Maschine auf internationalen Ausstellungen das Publikum. Zwar war der Nimrod kein universell programmierbarer Rechner, sondern eine ein speziell für das Spiel Nim gebauter Röhrenrechner, gleichwohl konnte Nimrod publikumswirksam für die Fähigkeiten der neuen »Elektronengehirne« werben. Vgl. zu Nimrod auch Turing (1953) sowie heise (o.J.).

06► Für eine (anglo-amerikanisch geprägte) Übersicht zur Computerschach-Literatur der ersten Jahrzehnte vgl. Marsland (1979).

07► »We want to report here what we believe is the first satisfactory program - one with which the machine plays a game sophisticated enough so that its opponent has to be something more than a novice to beat it«, so Bernstein und Roberts (1958) in ihrer Beschreibung des Spiels.

08► Als wegweisend für diese neue Spielstärke gilt das 1966/67 am MIT entwickelte Mac Hack, programmiert von Richard Greenblatt und Kollegen auf einem DEC PDP-6.

09► Pias formuliert diese These sehr direkt: »Daß sich Babbage, Zuse, Shannon, Turing oder Wiener mit dem Schachspiel beschäftigt haben, ist nicht Zufall oder nachträgliche ›Benutzung‹ des Computers zu Spielzwecken, sondern eine Hilfe, den Computer selbst erst zu denken« (Pias 2000, 159).

10► In der Computerforschung wird heute auch vermehrt das Spiel Go als interessanteres Forschungsobjekt und Alternative zu Schach angesehen.

11► Für die Auffassung von Schach als einem Ideal von Rationalität folgend, im Unterschied zu anderen Auffassungen von Schach als Kunstform, ästhetisches Spiel oder als ›Sport‹ vgl. aus philosophischer Sicht Siitonen/Pihlström (1998).

12► Vgl. zur Schachnotation als medientechnische Revolution des Schachspiels und ihre Folgen weiterführend Wiemer (2008).

13► Vgl. zu diesem Argument Ensmenger (2011, 18) sowie weiterführend Ensmenger (2010).

14► Das Harkness-System, entwickelt von Kenneth Harkness, wurde 1956 veröffentlicht. Es war vor diesem Zeitpunkt jedoch bereits praktisch im Gebrauch und wurde u.a. vom amerikanischen Schachverband USCF spätestens seit 1950 verwendet. Es ist ebenso wie das seit ca. 1960 gebräuchliche Elo-System ein Verfahren, um die Spielstärke von Schachspielern in einem Punktesystem zu bewerten und dadurch Ranglisten erstellen zu können. Das System der Punktvergabe basiert auf der Teilnahme und dem Spielerfolg in Turnierwettkämpfen.

15► Vgl. für Schach als »operationalisierte Wissensformation« Wiemer (2008, 144ff.). Rolf Nohr hat mehrfach zeigen können, wie Spiele als Übergangsobjekte zwischen unterschiedlichen Diskursformationen funktional werden. Vgl. hierzu weiterführend das Modell von Spiel als »Interspezialdiskurs« im Rahmen eines diskursanalytischen Ansatzes in Nohr (2008).

16► Einen Eindruck davon, wie groß ein solches Interesse bereits Anfang der 1950er Jahre war, gibt die Darstellung von Richard Sprague. Sprague (1972) schildert das immense öffentliche Interesse, das aus einer missverständlichen Äußerung entsprang, die er 1951 bei der Präsentation des CRC 102 tätigte. Ein Journalist zitierte ihn fälschlich mit der Aussage, der neue Computer sei in der Lage Schach zu spielen und dabei jeden menschlichen Gegner zu besiegen. Die Zeitungsmeldung eines kleineren Blatts an der Westküste führte zu zahlreichen Anfragen von Journalisten, zu Berichterstattungen in überregionalen Medien, zu einer Einladung zum öffentlichen Schachwettkampf, den das Life Magazin ausrichten wollte und schließlich zu einem Angebot, den Computer in der Dean Martin und Jerry Lewis Show in einem Schaukampf gegen Jerry Lewis antreten zu lassen. Dieses immense öffentliche Interesse bezog sich dabei stets auf die vermeintlichen Schach-Fähigkeiten der Maschine. Ab Mitte der 1960er Jahre wurde die Popularität von Computerschach immer wieder öffentlichkeitswirksam als Kampf Mensch-Gegen Maschine inszeniert, bis hin zu dem berühmten Zweikampf Garry Kapsarow gegen Deep Blue in den 1990er Jahren.

17► Der Aufsatz wurde im Herbst 1947 abgefasst und war zunächst nicht für die Öffentlichkeit bestimmt. Er wurde 1948 als Bericht an das National Physical Laboratory übersendet. Vgl. [http://www.alanturing.net/intelligent_machinery/] Erstmals veröffentlicht wurde der

Text schließlich 1968/69, mehr als ein Jahrzehnt nach Turings Tod. Auf Deutsch erschien der Text in Kittler/Dotzler (1987).

18► Dass bei dieser Versuchsanordnung die Geschlechteraufteilung den Intelligenztest der Maschine ›überlagert‹, dass es sich also zugleich um einen »gender-switching« Test handelt, ist eine interessante Pointe dieser Versuchsanordnung. Bettina Heintz hat herausgestellt, dass die im Imitationsspiel aufgeworfene Frage der Geschlechterdifferenz lange Zeit übergangen oder »als exzentrisches Vorspiel« abgetan wurde. Sie hat zugleich die verzwickte Doppelbödigkeit der Aufgabenstellung des Spiels deutlich auf den Punkt gebracht. Das »Geschlechterspiel« als Identitätsspiel wird nämlich durch das von Turing vorgeschlagene Verfahren selbst fragwürdig und destabilisiert: »Anhand der Antworten hat der Fragesteller herauszufinden, wer von den beiden ein Mann ist und wer eine Frau. Was aber passiert, wenn er sich irrt? Wenn er den Mann für eine Frau hält? Ist ein Mann, der für eine Frau gehalten wird, immer noch ein Mann? Oder ist er eine Frau? Oder etwas Drittes?« Vgl. Heintz (1993, 264ff.). Die Diskussion des Gender-Aspekts im Turing-Test wird allerdings auch dadurch erschwert, dass Turing im Verlauf der Argumentation auf die Ausgangsbedingungen zu sprechen kommt und es nicht leicht zu entscheiden ist, wie wichtig dabei die Gender-Aufteilung weitergehend auch für das Imitationsspiel von Mensch und Maschine sein soll. So schreibt er: »It might be urged that when playing the ›imitation game‹ the best strategy for the machine may possibly be something other than imitation of the behaviour of a man. This may be, but I think it is unlikely that there is any great effect of this kind. In any case there is no intention to investigate here the theory of the game, and it will be assumed that the best strategy is to try to provide answers that would naturally be given by a man.« In der deutschen Übersetzung des Textes, herausgegeben von Friedrich Kittler und Bernhard Dotzler, wird die Irritation über die Funktion des »gender-switching« als Bestandteil des Turing-Tests unglücklicherweise dadurch konterkariert, dass in dem zuletzt zitierten Abschnitt »man« schlicht als »Mensch« übersetzt wird. Vgl. Turing (1950, 151).

19► »Playing against such a machine gives a definite feeling that one is pitting one's wits against something alive.« (Turing 1948, 5).

20► Roch gibt eine hervorragende Übersicht zu Shannons unterschiedlichen Spielapparaturen. Der Schachrechner Caissac von 1949 war noch nicht in der Lage, eine vollständige Partie zu spielen. Er war auf eine vereinfachte Spielvariante spezialisiert und konnte insbesondere Endspielsituationen durchrechnen, in denen vier oder sechs Spielfiguren den Ausgang des Spiels entscheiden. Darin ähnelt Caissac dem von Dietrich Prinz programmierten Mate-in-Two, das als erstes Schachprogramm der Computergeschichte gilt. Auch Mate-in-Two operierte mit einer reduzierten Schachvariante, indem es auf die Lösung von Endspielproblemen spezialisiert war. Vgl. weiterführend zu Mate-in-Two [http://chessprogramming.wikispaces.com/Mate-in-two], sowie Copeland (2008).

21► In *Programming a Computer for Playing Chess* wird die ›philosophische‹ Frage, ob Computer denken können, zwar zu Beginn gestreift, jedoch in der Argumentation nicht weitergehend

berücksichtigt. An anderer Stelle kommt Shannon aber durchaus darauf zu sprechen. Im direkten Bezug zu Schachautomaten bspw. in Shannon (1950b). Axel Roch geht davon aus, dass Shannon und Turing 1943 in den Bell Labs in New Jersey diesbezügliche Ideen ausgetauscht haben: »Turing brachte aus England genau die Fragestellung mit, die Shannon immer schon beschäftigt hatte: Bis zu welchem Grad konnten Maschinen, die rechnen, auch denken?« Shannon hatte es als sein Anliegen formuliert, eine Maschine zu bauen, die »wirklich denkt, lernt, mit Menschen kommuniziert […]« (dt. d. Verf.). Vgl. Roch (2009, 93f.).

22► Vgl. zur Geschichte des Minimax-Algorithmus und seinem Einfluss auf die Entwicklung von Computerschach und die Funktion von Schach als Objekt der KI-Forschung weiterführend Ensmenger (2011).

23► Gerade aufgrund der genannten praktischen Vorteile bildet der Minimax-Algorithmus die Grundlage für zahlreiche Schachprogramme. Allerdings ist eine klare Trennung von Typ A und Typ B Strategien innerhalb der Schachprogrammierung nur selten gegeben; häufig verwenden Programme, die auf dem Minimax-Algorithmus aufsetzen, noch zusätzliche verfeinerte Entscheidungsroutinen und weitere Optimierungen, die auch an menschlichem Spiel- und Evaluationsstrategien orientiert sein können. Gleichwohl hat die Unterscheidung von Typ A und Typ B Strategien eine historische Bedeutung für die Entwicklung der Diskussion innerhalb der Forschungen zur künstlichen Intelligenz. Verbunden ist dies etwa mit der Frage unterschiedlicher Forschungsansätze, die mehr dazu neigen können, entweder menschliches Denken durch Nachahmung der zugrundeliegenden kognitiven Prozesse im Computer nachbilden zu wollen (mimetischer Ansatz), oder ohne diese Orientierung am menschlichen Modell Denk- und Entscheidungsprozesse maschinell zu realisieren. Erfolgreiche Schachprogramme gehen bis heute tendenziell den zweiten Weg: Sie imitieren nicht die Spielweisen oder Entscheidungsprozesse menschlicher Spieler, sondern setzen auf maschinelle Stärken wie die Verwendung von großen Datenbanken und hohe Rechengeschwindigkeiten.

24► Das Interesse an Grundlagenforschung bedeutet gleichwohl nicht, dass die Freude am Spiel irrelevant wäre – Turing schreibt, das hauptsächliche Motiv, das den Anstoß zur Arbeit, gegeben habe, sei »der reine Spaß an der Sache« (Turing 1953, 118).

25► Die Programmierbarkeit wird allerdings auch bei Turing als zentrale Herausforderung beschrieben. In seinem späteren Artikel zu Computerspielen von 1953 formuliert Turing dies allerdings sehr viel deutlicher als in dem hier diskutierten Aufsatz von 1947/48: Neben der bloßen Rechengeschwindigkeit der Maschinen und der geringen Größe ihres Speichers sei es vor allem die »Programmiertechnik«, die ein Hindernis für brauchbare Schachprogramme darstellt (Turing 1953, 129).

26► Ein Compiler ist ein Computerprogramm, das ein anderes Programm, das nicht in Maschinencode verfasst ist, in Maschinencode übersetzt, damit es von einem Rechner ausgeführt werden kann. In vielen Darstellungen gilt das *A-O-System*, das 1951/52 von Grace Hopper für den UNIVAC I entwickelt wurde als erster Compiler der Computergeschichte.

Dieser Compiler erlaubte Zugriff auf eine Liste von Subroutinen, die Grace Hopper über die Jahre erstellt hatte, und die mit Hilfe einer Nummer aufgerufen werden konnten. Neben Grace Hopper wird auch Alick Glennie als Programmierer des ersten Compilers genannt. Der Brite Glennie spielte nicht nur die erste überlieferte Partie gegen Turings Schach-Papiermaschine, sondern entwickelte auch *Autocode*, das von einigen als erster Compiler der Computergeschichte angesehen wird. Als erster vollständiger Compiler gilt der unter der Leitung von John Backus entwickelte FORTRAN-Compiler, der erst 1957 fertig gestellt werden konnte.

27► Konrad Zuses »Plankalkül« stellt hier eine Ausnahme dar. Zuses frühe Forschungen zu Programmiersprachen erfolgte bereits in den 1930er Jahren. Erst 1972 wurde Zuses Plankalkül jedoch komplett veröffentlicht.

28► Historisch ist die Arbeit von Allen Newell, Clifford Shaw und Herbert A. Simon ein anschauliches Beispiel für den Zusammenhang von Computerschach und Programmiersprachen. Für die Programmierung von Schach verwendeten sie eigene Sprachkonstrukte in der von ihnen entwickelten »Information Processing Language« (IPL). Newell und Simon interessierten sich, angeregt u.a. durch William Ross Ashbys und Norbert Wieners Formulierungen zur Kybernetik, für die Möglichkeiten maschinellen Problemlösens und entwickelten einige der frühesten Programme der Künstlichen Intelligenz. Allen Newell arbeitete mit John Clifford Shaw zu jener Zeit für die RAND Corporation und sie hatten zusammen mit Herbert Simon 1956 den »Logic Theorist« und ab 1957 den »General Problem Solver« entwickelt, ersteres ein Programm, das in der Lage war, eine Menge von logischen Theoremen zu beweisen, letzteres ein »Programm zur Simulation menschlichen Denkens«. Beide Programme gelten als Grundlagenarbeiten für die Künstliche Intelligenzforschung der folgenden Jahrzehnte. Während Newell die Gebiete der Informatik und der Kognitionswissenschaft zu verbinden versuchte, orientierte sich Simon zudem an Wirtschafts- und Sozialwissenschaften und wurde 1978 »für seine bahnbrechenden Erforschung von Entscheidungsprozessen in Wirtschaftsorganisationen« mit dem Nobelpreis für Wirtschaftswissenschaften ausgezeichnet. Shaw war bei RAND als System-Programmierer tätig und leistete einen wichtigen Beitrag zur Entwicklung der Programmiersprachen, in denen der Logic Theorist und GPS programmiert wurden. Die Information Processing Language (IPL), die Newell, Shaw und Simon um 1956 bei RAND entwickelten, gilt als die erste listenverarbeitende Programmiersprache. Sie implementiert umfangreiche Konzepte, die auch spätere höhere Programmiersprachen aufweisen, wie Rekursionen, Dynamische Speicherverwaltung, Datentypen etc., ist allerdings im Sprachstil eng an Assembler-Sprachen angelehnt, und entsprechend schwer lesbar. Ihr Aufsatz *Chess-playing programs and the problem of complexity* von 1958 stellt so etwas wie den vorläufigen Höhepunkt des Schach-Computer-Diskurses der 1950er Jahre dar und markiert zugleich den Übergang von der Schachprogrammierung zu frühen Ansätzen der KI-Forschung. Als Programmiersprache, die umfangreiche Konzepte zur Manipulation von Listen enthielt, stellte IPL eine wichtige Inspiration für John McCarthy dar, der mit LISP

eine der wegweisenden Programmiersprachen der 1960er und 1970er entwickelte. LISP entstand am MIT und der erste Interpreter dazu wurde von Steve Russel programmiert (1958), der später mit SPACEWAR! (1961) in die Geschichte der Computerspiele eingehen sollte. LISP ist nicht nur für Jahrzehnte die bevorzugte Sprache in der KI-Forschung, sondern neben Fortran und Cobol auch eine der ältesten höheren Programmiersprachen überhaupt und auch heute noch in Gebrauch. Im Unterschied zu anderen frühen Programmiersprachen wir FORTRAN (The IBM Mathematical Formula Translating System) (1957), das auf die Formelberechnung zugeschnitten war, oder COBOL, ein Akronym für COmmon Business-Oriented Language (1959), war LISP eine aus theoretischen Überlegungen entstandene Programmiersprache, die auf Symbolverarbeitung und Problemlösen ausgerichtet war. LISP hat bis heute unter Programmierern und Hackern einen ausgezeichneten Ruf: Sie implementiert auf elegante Weise das Lambda-Kalkül von Church und Kleene und gilt aufgrund ihrer flexiblen Struktur als „programmierbare Programmiersprache« (John Foderaro) oder auch »Meta-Programmiersprache« (zit. nach [http://en.wikiquote.org/wiki/Lisp_programming_language]).

29► Weil sie Algorithmen formulierte, die exakt für die Prozessierung durch eine Rechenmaschine bestimmt waren, wäre vermutlich Ada Lovelace als die erste Programmiererin zu nennen. Der in den 1950ern einsetzende Diskurs über Programmierung hat ihre Arbeiten dann auch tatsächlich wiederentdeckt: Ihre ›Notizen‹ von 1843 wurden 1953 in *Faster than Thought* (hg. von B.V. Bowden) wiederveröffentlicht, in unmittelbarer Nähe zu Turings Überlegungen zur Programmierung von Spielen. Ihr Portrait schmückt die erste Seite des Sammelbandes. Wenn hier der ›Beginn von Software‹ dennoch nicht im 19. Jahrhundert, sondern in den 1950er Jahren verortet wird, soll dies die historische Leistung von Lovelace nicht infrage stellen, vielmehr ist die Rede vom ›Beginn‹ hier auf die spezifische historische und diskursive Konstellation nach dem Zweiten Weltkrieg bezogen, in der die Grundlagen für die spätere Entwicklung des Computers und der Computerindustrie gelegt wurden, die später schließlich zum Computer als Knotenpunkt digitaler Medienkultur führen.

Bibliografie

Alt, Casey (2011): Objects of Our Affection. How Object Orientation made Computers a Medium. In: Media Archaeology: Approaches, Applications, and Implications. Hrsg. von von Erkki Huhtamo & Jussi Parikka. Berkeley: University of California Press, S. 278-301.

Babbage, Charles (1864): Passages from the life of a philosopher. London: Longman, Roberts & Green.

Bernstein, Alex und Roberts, Michael de V. (1958): Computer v. Chess-Player. In: Scientific American, Vol. 198, S. 96-105.

Ceruzzi, Paul E. (2003): Eine kleine Geschichte der EDV, Bonn: Mitp.

Chun, Wendy Hui Kyong (2008): Programmability. In: Software studies a lexicon. Hrsg. von Matthew Fuller. Cambridge, Mass.: MIT Press, S. 224–229.

Copeland, B. Jack (2008): The Modern History of Computing. In: The Stanford Encyclopedia of Philosophy. Hrsg. von Edward N. Zalta. Online: [http://plato.stanford.edu/archives/fall2008/]; zuletzt abgerufen 24.01.2014.

Ensmenger, Nathan (2010): The computer boys take over computers, programmers, and the politics of technical expertise. Cambridge, Mass.: MIT Press.

Ensmenger, Nathan (2011): Is chess the drosophila of artificial intelligence? A social history of an algorithm. In: Social Studies of Science 42/1, S. 5–30.

Hagen, Wolfgang (1994): Computerpolitik. In: Computer als Medium. Hrsg. von Friedrich Kittler, Georg Christoph Tholen und Norbert Bolz. München: Fink 1994, S. 139–160.

Heintz, Bettina (1993): Die Herrschaft der Regel. Zur Grundlagengeschichte des Computers. Frankfurt/M.; New York: Campus.

heise (o.J.): Vor 50 Jahren fing alles an: das erste ›Elektronenhirn‹, [http://www.heise.de/newsticker/meldung/Vor-50-Jahren-fing-alles-an-das-erste-Elektronenhirn-in-Deutschland-51722.html]; zuletzt abgerufen 24.01.2014.

Kittler, Friedrich / Dotzler, Bernhard J. (1987): Nachwort. In: Kittler/Dotzler 1987, S. 209–235.

Kittler, Friedrich / Dotzler, Bernhard J. (Hrsg.) (1987): Intelligence Service: Schriften. Berlin: Brinkmann & Bose.

Krämer, Sybille (1995): Spielerische Interaktion: Überlegungen zu unserem Umgang mit Instrumenten. In: Schöne neue Welten? Auf dem Weg zu einer neuen Spielkultur. Hrsg. Von Florian Rötzer. München: Klaus Boer, S. 225–236.

Lisp_programming_language (o.J.) In: Wikipedia (en). [http://en.wikiquote.org/wiki/Lisp_programming_language]; zuletzt abgerufen 24.01.2014.

Marsland, T. A. (1979): A Bibliography of Computer Chess. In: Machine intelligence 9, S. 385–403.

Neumann, John von/ Morgenstern, Oskar (1990) [1944]: Theory of games and economic behavior. (Reprint Aufl.) Princeton: Princeton Univ. Press.

Newell, Allen/ Shaw, John Clifford / Simon, Herbert Alexander (1958): Chess-playing Programs and the Problem of Complexity. In: IBM Journal of Research and Development 2.4 (1958), S. 320–335.

Nohr, Rolf F. (2008): Die Natürlichkeit des Spielens. Vom Verschwinden des Gemachten im Spiel. Münster: LIT Verlag

Petschar, Hans (1986): Kulturgeschichte als Schachspiel. Vom Verhältnis der Historie mit den Humanwissenschaften. Variationen zu einer historischen Semiologie. (= Aachener Studien zur Semiotik und Kommunikationsforschung, Bd. 11). Aachen: Rader.

Pias, Claus (2000): Computer-Spiel-Welten. Phil. Diss. Weimar: Universität Weimar. ftp://ftp.uni-weimar.de/pub/publications/diss/Pias/pias.pdf.

Putnam, Hilary (1996): Viel Lärm um fast nichts. In: Probleme der künstlichen Intelligenz: eine Grundlagendiskussion. Hrsg. von Stephen Richards Graubard. Wien; New York: Springer, S. 257–268.

Randell, Brian (1982): From analytical engine to electronic digital computer: the contributions of Ludgate, Torres, and Bush. In: Annals of the History of Computing 4/4, S. 327–341.

Roch, Axel (2009): Claude E. Shannon: Spielzeug, Leben und die geheime Geschichte seiner Theorie der Information. Berlin: gegenstalt Verlag.

Rojas, Raúl u. a. (o.J.): Konrad Zuses Plankalkül – Seine Genese und eine moderne Implementierung. FU Berlin, FB Mathematik und Informatik. Online: [http://www.zib.de/zuse/Inhalt/Programme/Plankalkuel/Genese/Genese.pdf]; zuletzt abgerufen 24.01.2014.

Schachprogramm (o.J.): In: Wikipedia, Die freie Enzyklopädie. Bearbeitungsstand: 22. Juli 2013, 14:26 UTC. [http://de.wikipedia.org/w/index.php?title=Schachprogramm&oldid=120783650]; zuletzt abgerufen 24.01.2014.

Shannon, Claude E. (1950): Programming a Computer for playing chess. In: Sloane/Wyner 1993, S. 637–656. Zuerst erschienen als C. E. Shannon, »Programming a computer for playing chess«, Phil. Mag., 41, März 1950, S. 256-75.

Shannon, Claude E. (1950b): A Chess-Playing Machine. In: Sloane/Wyner 1993, S. 657–666.

Shannon, Claude E. (1953): The Potentialities of Computers. In: Sloane/Wyner 1993, S. 691–694.

Siitonen, Arto / Sami Pihlström (1998): On the Philosophical Dimensions of Chess. In: Inquiry 41/4, S. 455–475.

Sloane, N.J.A. / Wyner, Aaron D. (Hrsg.) (1993): Claude Elwood Shannon. Collected Papers. New York: IEEE.

Sprague, Richard E. (1972): A western view of computer history. In: Communications of the ACM, 15/7, S. 686–692.

Turing, Alan (1948) [1968]: Intelligente Maschinen. In: Kittler/Dotzler 1987, S. 83–113.

Turing, Alan (1950): Rechenmaschinen und Intelligenz. In: Kittler/Dotzler 1987, S. 149–182. Zuerst erschienen unter dem Titel: »Computing Machinery and Intelligence«, Mind 59/236 (1950), S. 433–460.

Turing, Alan (1953): Spielprogramme. In: Kittler/Dotzler 1987, S. 115–145. Zuerst erschienen unter dem Titel: »Digital computers applied to games«. In: Faster than thought. Hrsg. von B.V. Bowden, London 1953, Pitman Publishing.

Wiemer, Serjoscha (2008): Ein ideales Modell der Vernunft? Überlegungen zur Regelhaftigkeit und strategischen Rationalität des Schachspiels. In: Strategie Spielen. Medialität, Geschichte und Politik des Strategiespiels. Hrsg. von Rolf F. Nohr & Serjoscha Wiemer. Münster 2008: LIT Verlag, S. 136-161.

Zuse, Konrad (2010): Der Computer - Mein Lebenswerk. Berlin, Heidelberg: Springer-Verlag.