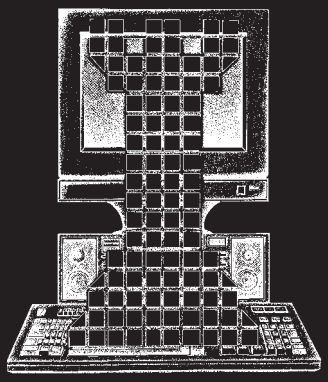
Chrilly Donninger



Von Bytes und Bauern

W. Steinitz gegen L. Gott

Die Schlange war schlauer als alle Tiere des Feldes, die Gott, der Herr, gemacht hatte. Sie sagte zu der Frau: Hat Gott wirklich gesagt: Ihr dürft von keinem Baum des Gartens essen? Genesis 3,1.

Der Schachlegende nach wollte der erste Weltmeister Wilhelm Steinitz am Ende seiner Karriere unbedingt gegen Gott spielen. Um Gott diese Partie schmackhafter zu machen, bot Steinitz ihm einen Bauern als Vorgabe an. Aufgrund der schlechten Erfahrungen beim letzten Auswärtsspiel dürfte Gott auf einem Heimmatch bestanden haben. Steinitz reiste zu diesem auch kurze Zeit später ab.

In der schönen neuen Internet-Welt könnte sich Wilhelm Steinitz die beschwerliche Überfahrt über den Hades ersparen. Er bräuchte nur die Homepage des UNIX-Schöpfers Ken Thompson (http://cm.bell-labs.com/cm/cs/who/ken) zu besuchen und könnte dort unter der Rubrik „Play with God – Chess Endgames“ zumindest teilweise seinen Traum verwirklichen. Der Thompson-Gott kann erst mit sechs Steinen spielen, während Steinitz ein 31-Steiner vorgeschwebt hat. Zieht man aber die unterschiedlichen Anfahrtsstrapazen in Betracht, ist das wahrscheinlich ein erträglicher Kompromiss. Steinitz' Bauernvorgabe ist übrigens überhaupt nicht verrückt. Nachdem Gott für die Erschaffung der Welt sechs Tage – plus einen Rasttag – gebraucht hatte, befürchtete Steinitz wohl, dass Gott mit der Konstruktion der 32-Steiner Datenbank noch nicht fertig sei. Bauern sind wegen der eingeschränkten Symmetrie, dem Schlagen en passant und der Umwandlung besonders rechenaufwendig. Die Chancen, dass Gott zumindest einen 31-Steiner fertig hatte, waren daher erheblich größer.

Zweifaltigkeit?

Im Grunde ist die Frage, ob 6-, 31- oder 32-Steiner, nur ein nebensächliches mathematisches Detail. Viel interessanter ist, ob der reale und der virtuelle Gott auf gleiche Weise spielen. Ziemlich genau weiß man, wie ein Thompson-Gott spielt. Das Checkers (8x8-Dame)-Programm Chinook von Jonathan Schaeffer rechnet sehr oft aus dem Eröffnungsbuch direkt in die Achtsteiner-Endspieldatenbank. Ohne perfektes Datenbank-Wissen ist Chinook ein feuriger Angreifer, mit eingeschaltetem Achtsteiner ein kaum zu besiegender, aber fader Remisschieber. Der über 40 Jahre lang regierende Checkers-Weltmeister Marion Tinsley beschloss noch während des Kampfes um den Mensch-Maschine-Titel, dass ein Kampf gegen L. Gott doch wesentlich interessanter sein müsste. Er trat wie W. Steinitz die Überfahrt an. Ein nettes Beispiel für die groteske Auswirkung von perfektem Wissen auf ein Schachprogramm stammt vom niederländischen Schriftsteller und Schachspieler Tim Krabbé. In Diagramm 1 spielt das Schachprogramm HIARCS aufgrund seines perfekten Wissens den unglaublichen Zug 1. Df7†. Nach 1. ... Kxf7 ist HIARCS glücklich, weil es in seine 5-Steiner-Datenbank gelangt. Die Datenbank signalisiert HIARCS ein Matt. Das ist viel besser als der schnöde Vorteil von Dame und zwei Bau-

ern nach 1. Dxf4. Das logische und kürzere Matt nach 1. Kb2 ist für Hiarcs zu weit weg. 1. Df7† ist skurril, ändert aber nichts am Ergebnis. Derselbe Vorgang könnte sich aber auch mit umgekehrten Vorzeichen abspielen. Das Programm verschenkt Figuren, weil ihm seine Datenbank meldet, dass es ansonsten Matt in 250 gesetzt wird.

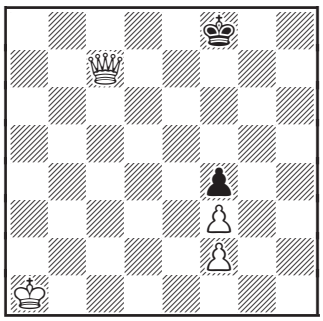


Diagramm 1 Weiß am Zug

HIARCS – N. N.

Nachdem es selbst Michael Ehn noch nicht gelungen ist, das Original-Partieformular W. Steinitz gegen L. Gott, Himmelreich i. A., 1901, aufzutreiben, kann man über die wahre Spielanlage von L. Gott nur spekulieren. Die Wahrscheinlichkeit, die Partiemitschrift jemals zu finden, ist sehr gering. Michael Ehn verfolgt die Angelegenheit auch nicht mit seiner sonst üblichen Beharrlichkeit. Laut Ehn ist die Steinitzsche Ankündigung zwar gut erfunden, aber historisch in keiner Weise belegbar.

Eva und die Schlange

Bei der Erforschung der Spielanlage von L. Gott sind wir daher auf andere Quellen angewiesen. Eine gute Beschreibung findet sich in Genesis 3,1-24. L. Gott geht offensichtlich davon aus, dass sein menschlicher Gegner weniger weiß als er selbst. Wie aus dem Einleitungs-Zitat hervorgeht, stellt die kluge Schlange Eva mit dem Apfel eine Falle. Eine mit göttlichen Wissen ausgestattete Eva hätte den Apfel nie genommen, weil sie das daraus resultierende Schlamassel – „Viel Mühsal bereite ich dir, sooft du Schwanger wirst. Unter Schmerzen gebierst du Kinder ...“ – gesehen hätte. Ein Thompson-Gott hätte folglich die Apfel-Falle nie gestellt, weil er von einer göttlichen Eva ausgeht. Die Strategie von L. Gott ist aber noch viel hintergründiger angelegt. Er bringt Adam und Eva in eine sehr schwer zu durchschauende Situation. Sie dürfen alle Früchte Edens essen, nur jene vom Baum in der Gartenmitte nicht. Die Sonderstellung dieses einen Baumes ist für Eva nicht einsichtig. Hätte L. Gott den Genuss aller Früchte verboten, wäre Eva mit ziemlicher Sicherheit nicht in die Apfelfalle getappt. Ferner startete L. Gott seinen Angriff nicht direkt auf den laut Sigi F. mit einem stärkeren Überich ausgestatteten Adam. Er nützt Adams Schwachstelle Eva aus. Ein Thompson-Gott weiss alles über Schach, über das Verhältnis von Adam zu Eva hat er aber nicht die geringste Ahnung. Aufgrund dieses und ähnlicher Berichte können wir also davon ausgehen, dass L. Gott wahrlich kein fader Remisschieber ist. Wahrscheinlich kam unter den Sterblichen der Hexer Tal seinem Stil am nächsten. Unklar ist, ob L. Gott so wie M. Tal auch objektiv inkorrekte Opfer spielt. Die vorhandenen Quellen weisen darauf nicht hin.

Stein, Papier, Schere

Schachprogramme spielen bis auf unbedeutende Modifikationen wie der Thompson-Gott. Ein mit einer 20-Steiner-Datenbank ausgerüstetes Programm würde bestenfalls ein perfekter T. Petrosjan, niemals aber ein M. Tal. Wahrscheinlich würde es aufgrund seiner Remisneigung kaum Turniere gewinnen. Im Rahmen eines direkten WM-Kampfes wäre es aber kaum zu knacken.

Im September 1999 wurde die erste internationale Computer-Meisterschaft im RoShamBo (Stein, Papier, Schere) abgehalten. Ein RoShamBo-Match besteht aus 1000 Stein-Papier-Schere-Duellen. Eine nicht zu widerlegende RoShamBo-Strategie ist bekannt. Man spielt mit je 1/3 Wahrscheinlichkeit eine der drei möglichen Alternativen. Das mit dieser Strategie spielende Programm META-META-RANDOM belegte nur den 27. Platz unter 55 Teilnehmern. META-META-RANDOM ist zwar im direkten Duell nicht zu biegen, es war aber – im Gegensatz zum Sieger IOCAINE POWDER – nicht in der Lage, die Schwächen anderer Programme auszunützen. Offensichtlich kann man ein RoShamBo-Turnier nur gewinnen, wenn man das Risiko eingeht, eine Partie zu verlieren. RoShamBo ist ein primitives Kinderspiel. Die Aufgabe, den Gegner auszuhorchen, seine Strategie zu erkennen und auszunutzen, ist aber alles andere als trivial. Ein besonders interessantes Ergebnis des Turnieres war der dritte Platz von DE-BRUIJN. DE-BRUIJN reagiert überhaupt nicht auf die Spielweise des Gegners. Es spielt eine so genannte De-Bruijn-Sequenz. De-Bruijn-Sequenzen sind

relativ kompliziert geschachtelte mathematische Folgen. Jede Schachtelungsebene hat ihr eigenes Muster. Innerhalb einer Schachtelungsebene werden Stein, Papier, Schere mit unterschiedlicher Häufigkeit gespielt, über die verschiedenen Ebenen gemittelt ergibt sich aber die optimale 1/3-Häufigkeit. Der Erfolg des Programmes DE-BRUIJN beruht auf der Irreführung des Gegners. Sie hatten das Muster für eine Ebene durchschaut und versuchten ihn nun auszubeuten. Inzwischen hatte aber DE-BRUIJN seine Ebene gewechselt und spielte ein anderes Muster. Wer sich ein falsches Bild von seinem Gegenüber macht, wird selbst bestraft. Ein besonders intelligenter Gegner – bei diesem ersten Turnier war keines der Programme besonders ausgefuchst – hätte DE-BRUIJN aber durchschaut und daraufhin jedes Duell gewonnen.

Die DE-BRUIJN-Strategie ist Autismus in Reinkultur. Auf einer Meta-Ebene ist die Strategie aber ein sehr gefinkeltes Gegner-Model.

Die Schachprogrammierer haben sich über diesen Aspekt bisher keine sehr tief schürfenden Gedanken gemacht. Schachprogramme sind wie die DE-BRUIJN-Strategie ebenfalls Autisten. Zwar reagieren sie auf die Züge des Gegners, für ihre weitere Berechnung nehmen sie aber an, dass sie gegen sich selbst spielen. Dem Computerschach-Autismus fehlt jede Hintergründigkeit. Er soll nur dem Schachprogrammierer das Leben erleichtern. Schach ist auch so schwierig genug. Das Luxusproblem „Was tue ich mit perfektem bzw. überlegenem Wissen?“ stellt sich erst seit kurzem.

Der Forster-Ansatz

Die bisher umfangreichste Arbeit zu diesem Thema stammt vom Schweizer IM Richard Forster. Forster hat im Rahmen seines Informatikstudiums an der Universität Zürich die Semesterarbeit „Introducing the Human Factor in Computer Chess Programs“ geschrieben. Die Zusammenfassung dieser Arbeit lautet:

„Anstelle eines perfekten Gegners wird ein normaler, fehlbarer menschlicher Spieler angenommen. In ein paar Situationen wird unter dieser Annahme der Computer eine andere Entscheidung treffen als unter der Annahme eines perfekten Gegners. Die Annahme des fehlbaren Gegners wird auf drei Ebenen untersucht: Auf der Ebene der Bewertungsfunktion, der Suche und im endgültigen Entscheidungsprozess zur Auswahl eines Zuges. Es wird besonderer Augenmerk auf den Versuch gelegt, das menschliche Spiel zu modellieren und vorauszusagen. Die besondere Betonung liegt dabei auf den typischen Schwächen und Stärken von Menschen im Verhältnis zu jenen des Computers.“

(Übersetzung aus dem Englischen vom Verfasser)

Der Fall Duchess – Kaissa

Ausgangspunkt von Forsters Überlegungen ist eine berühmte Computerpartie. Bei der zweiten Computerschach-Weltmeisterschaft in Toronto kam 1977 zwischen dem amerikanischen Programm DUCHESS und dem sowjetischen Titelverteidiger KAISSA nach 34. Da8† die Stellung von Diagramm 2 aufs Brett.

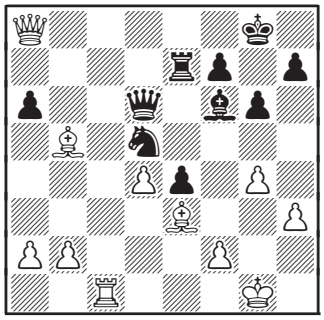


Diagramm 2 Schwarz am Zug

DUCHESS – KAISSA, Toronto 1977.

Die Zuschauer im Saal, unter ihnen auch Michail Botwinnik, erwarteten den logischen Zug 34. ... Kg7. Zum Entsetzen seiner Schöpfer spielte KAISSA aber **34. ... Te8** und die Partie war endgültig gelaufen. Die Anwesenden waren einhellig der Meinung, dass KAISSA von einem schrecklichen Bug heimgesucht wurde.

Die Programmierer begannen auch sofort mit der Fehlersuche und stellten fest: 34. ... Te8 ist aus Sicht KAISSAS vollkommen korrekt. Nach 34. ... Kg7 folgt 35. Df8†!! Kxf8 36. Lh6† Kg8, und nach 37. Tc8† ist das Matt nicht mehr zu verhindern. KAISSA hatte diese Variante gesehen. Der Turmverlust auf e8 ist immer noch besser als mattgesetzt zu werden. Duchess hätte Df8† wahrscheinlich ebenfalls gesehen – moderne Programme finden den Zug in Sekundenbruchteilen.

Wie die Reaktion der Zuschauer beweist, hätte KAISSA gegen einen menschlichen Gegner mit 34. ... Kg7 aber gute Schwindelchancen gehabt (wenn auch die Stellung nach Kg7 objektiv verloren ist). Es ist offensichtlich nicht immer am besten, seinen Schaden zu minimieren.

Asymmetrie und das Gleichgewicht der Fehler

Ken Thompson hat in einer Reihe von Experimenten starke Spieler wie etwa GM Edmar Mednis gegen seine Datenbank spielen lassen. Trotz eines nicht unbeträchtlichen finanziellen Anreizes waren Mednis & Co. nicht in der Lage, das Endspiel Dame gegen Turm auf Anhieb zu gewinnen. Die längste optimale Sequenz bis zum Verlust des Turmes beträgt 31 Züge. Man kann also einige Züge herschenken, bevor die 50-Züge-Regel zuschlägt. Erst nach einigen Versuchen gelang es Mednis, die Datenbank zu durchschauen und doch innerhalb der 50-Züge-Regel zu gewinnen.

Thompson stellte weiters fest, dass die Zuganzahl bis zum Fall des Turmes in menschlichen Partien gut mit der theoretischen Distanz übereinstimmt. In diesen Auseinandersetzungen herrscht ein Gleichgewicht der Fehler.

Als Konsequenz aus diesen Experimenten sollte ein Programm das Endspiel Dame gegen Turm für sich mit Matt bewerten. Für den menschlichen Gegner ist das Endspiel aber nicht viel mehr als ein Remis wert. Mit anderen Worten: Vor die Alternative gestellt, zwei Bauern herzugeben oder in das theoretisch verlorene Endspiel Dame gegen Turm abzuwickeln, sollte das Programm gegen einen menschlichen Gegner die Endspielvariante wählen.

Diagramm 3 zeigt die Ausgangsstellung für die bisher längste von Ken Thompson berechnete Mattsequenz.

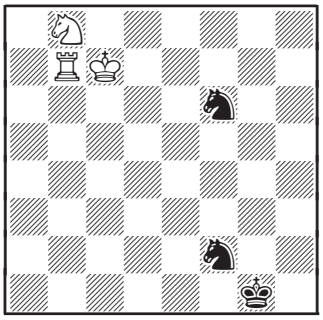


Diagramm 3 Weiß am Zug

Weiss am Zug setzt Matt in 262. Nach den FIDE-Regeln ist diese Position bei optimalem Spiel ein Remis. Ist es aber Remis, wenn ein Programm mit dieser Datenbank gegen einen Menschen spielt? In Diagramm 3 muss der Mensch wahrscheinlich schon einige Hacker machen, um in 50 Zügen zu verlieren. Es sind aber eine Reihe von Folgen mit Matt in 60 oder 70 bekannt, die wesentlich schwerer zu verteidigen sind. Die Bewertung 0 ist für diese Stellungen sicher falsch. Als erste Näherung könnte man sich die Formel 10000/(Mattdistanz-49) vorstellen (1 Bauer = 100). Für das Matt in 262 würde diese Formel einen Vorteil von etwa einem halben Bauern berechnen, ein Matt in 60 entspricht einer Mehrdame.

Schwieriges Terrain

Das Asymmetrie-Prinzip ist auch im Mittelspiel anwendbar. Im Gegensatz zum späten Endspiel ist die Programm-Bewertung in dieser Phase alles andere als perfekt. Die zusätzlichen Gewichte müssen daher weit vorsichtiger vergeben werden. Das Programm soll für sich günstiges Terrain ansteuern. In offener Feldschlacht siegt meist die Feuerkraft der schweren Schlachtpanzer, während sich im Häuserkampf, in geschlossenen Stellungen, die wendige Infantrie eher durchsetzt. Die Bewertungsfunktion muss daher speziell das Öffnen einer Stellung mit Hilfe von Bauernabtausch positiv für sich bewerten. Auch die Mobilität, die Anzahl der legalen Züge, sollte asymmetrisch gewichtet werden. Das Programm sollte Stellungen mit großer Mobilität anstreben. Diese beiden Faktoren sind in vielen Programmen schon seit langem eingebaut. Richard Forster geht in seiner Arbeit noch einen Schritt weiter. Die menschliche Stärke ist die effektive Mustererkennung. Ein Programm ist nicht auf Mustererkennung aufgebaut. Der Mangel an bekannten Mustern trifft daher lediglich den menschlichen Spieler. Das ist allerdings auch gleichzeitig das Problem. Wie soll ein Programm eine Stellung mit seltenen Mustern erkennen, wenn es nicht weiß, was ein Muster ist?

Wild um sich schlagen

Das allgemeinste und einfachste Muster, die Anzahl der Figuren auf jeder Seite, wird auch von einem Programm erkannt. Forster erwähnt in seiner Arbeit als Beispiel für eine seltene und für den Menschen schwierig zu spielende Postion die Figurenverteilung Turm + 2 Bauern gegen 2 Leichtfiguren. Man könnte dieses Klassifizierungssystem automatisieren, indem man aus der Mega-Datenbank von ChessBase für jede ungefähr gleichgewichtige Figurenkonstellation die Häufigkeiten berechnet. Stellungen mit geringer Häufigkeit sind für den Computer nach diesem Ansatz besser

als häufig gespielte Figurenverhältnisse.

Das Amateurprogramm HOSSA von Steffen Jakob hat einen asymmetrischen „Wild-um-sich-Schlag-Modus“ eingebaut. Wenn HOSSA mindestens mit 1,5 Bauern im Rückstand ist, werden eigene Königsangriffe stärker bewertet. Das Programm ist dadurch gewillt, bis zu 2 Bauern in einen Königsangriff zu stecken, auch wenn dieser innerhalb seines Suchhorizontes zu keinem Erfolg führt. HOSSA blitzt meist auf Internet-Schachservern. Laut Steffen Jakob hat HOSSA mit diesem Modus schon einige Gegner überrascht. Die HOSSA-Strategie, statt sich langsam abschlachten zu lassen, noch wild um sich zu schlagen und vielleicht einen glücklichen Treffer zu landen, ist vor allem beim Blitzen plausibel.

Suchfehler

Der Suchprozess von menschlichen und maschinellen Spielern ist gänzlich anders. Menschen sind dank der Mustererkennung in der Lage, die meisten Züge als irrelevant auszusortieren. Wenn sie überhaupt suchen, dann geschieht dies entlang ganz schmaler, dafür aber tiefer Varianten. Brute-Force-Programme suchen konzeptionell alle Varianten durch. Die Betonung liegt hier auf konzeptionell, de facto untersuchen auch reine Brute-Force-Programme nur einen Bruchteil der möglichen Züge. Die Erklärung hierfür heißt Alpha-Beta. Alpha und Beta sind die untere und obere Schranke des aktuellen Suchprozesses. Angenommen, wir haben in einer bestimmten Stellung bereits den ersten Zug untersucht und er liefert eine Bewertung von 0.10 zurück. Die untere Schranke Alpha beträgt nun 0.10. Es kann mit einem der restlichen Züge nur noch besser werden. Beim Durchrechnen des zweiten Zuges erhalten wir – nach dem Ausführen des ersten gegnerischen Zuges – eine Bewertung von 0.0. Obwohl wir nur einen einzigen Gegenzug ausgeführt haben, wissen wir bereits, dass wir den zweiten Zug nicht spielen werden. Seine Bewertung liegt unter der Schranke Alpha. Es ist Zeitvergeudung, noch weitere Gegenzüge für einen Zug zu analysieren, der sowieso nicht gespielt wird. Das Alpha-Beta Suchverfahren liefert mit minimalem Aufwand dasselbe Ergebnis wie ein tatsächliches Durchsuchen aller Möglichkeiten. Es entspricht daher – wie bereits erwähnt – konzeptionell einer vollständigen Brute-Force Suche. Die modernen Programme suchen nicht einmal konzeptionell nach dem Brute-Force-Ansatz. Mit Hilfe des Nullmoves wird die Suche wesentlich selektiver. Das Nullmove-Konzept bedeutet, dass überhaupt kein Gegenzug ausgeführt wird. Leider unterscheidet Forster in seiner Arbeit nicht zwischen Brute-Force-Konzept und den tatsächlich mit Alpha-Beta berechneten Varianten. Seine Überlegungen laufen darauf hinaus, den Menschen in solche Stellungen zu „locken“, in denen er besonders exakt rechnen muss. Es gibt jeweils nur eine gute Widerlegung, einmal den besten Zug zu übersehen, bedeutet den Ringstaub zu küssen. Dies erhöht nicht nur kurzfristig die Fehleranfälligkeit, der Spieler steht ständig unter Druck und ermüdet rascher. Ferner erhöht sich die Wahrscheinlichkeit für spätere Zeitnot. Aufgrund des Alpha-Beta-Algorithmus weiß man in der Regel aber nicht, ob es in einer bestimmten Stellung eine oder mehrere annähernd gleich gute Züge gibt. Wie bereits oben erwähnt, genügt oft bereits ein einziger Gegenzug, um einen Zug verwerfen zu können. Dieser Gegenzug muss nicht einmal die beste Antwort sein. Beim Ausführen eines Nullmoves wird kein einziger Gegenzug ausprobiert. Das Programm hat keinerlei Information über die Verteilung der Gegenzüge.

Alpha-Beta wegzulassen und stattdessen die volle Suche zu verwenden, wäre extrem kontraproduktiv. Die Suchtiefe würde durch diesen Schritt halbiert. Allerdings verwendete DEEP BLUE in einer etwas modifizierten Version diese Idee. Die forcierten Züge (DEEP BLUEs Team taufte sie „Singular Moves“) wurden zusätzlich zum normalen Mechanismus Alpha-Beta mit einer seichteren Suche bestimmt. DEEP BLUE verwendete diese Information aber nicht zur Änderung seiner Bewertung. Die Singular Moves bewirkten eine so genannte Singular Extension. Forcierte Varianten wurden tiefer berechnet. Die Singular-Extension-Methode hatte einen defensiven Charakter. Das DEEP-BLUE-Team musste feststellen, dass beim Königsangriff starke menschliche Spieler das Programm ausrechneten. Auch ein 8-Prozessoren-FRITZ mit 3,5 Millionen Stellungen pro Sekunde wurde beim letzten Frankfurter Open auf diese Weise drei Mal in den Ringstaub geschickt. Dank der Singular Extensions wurde in DEEP BLUE diese

Achillesferse weniger verwundbar. Es ist aber umstritten, ob der zusätzliche Aufwand zur Feststellung der Singular Moves das Ergebnis rechtfertigt. Die ersten Artikel über Singular Extensions berichteten einen Elozuwachs von 100 Punkten, spätere Aufsätze tendierten in Richtung Null. Möglicherweise könnte man DEEP BLUES und Forsters Ansatz kombinieren. Nachdem Singular Extensions nicht geschadet haben, erhielte man praktisch gratis einen sehr interessanten Bewertungs-Term.

Wer anderen eine Grube gräbt

Forster definiert eine Falle als eine Position, in der ein unmittelbar einleuchtender (Gegen-)Zug zum Verlust führt. Bei einer echten Falle geht auch der Fallensteller ein Risiko ein. Findet der Gegner die richtige Antwort, kommt der Fallensteller selbst in Bedrängnis. Ein nettes Beispiel für eine echte Falle ist folgende Partie:

SIRE LEGAL DE KERMEUR – N. N.
Paris ca. 1750

1. e4 e5 2. Lc4 d6 3. Sf3 Sc6 4. Sc3 Lg4 5. Sfxe5?!

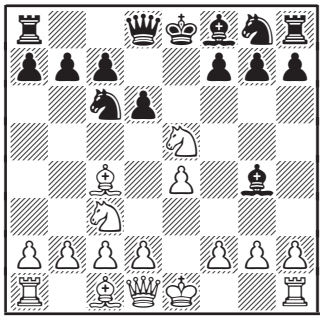


Diagramm 4 Schwarz am Zug

N. N. tappt prompt in die Falle: **5. ... Lxd1?? 6. Lxf7† Ke7 7. Sd5†** matt. Nach 5. ... Sxe5 hätte der Sire das Nachsehen gehabt. Diese Falle war wahrscheinlich auch 1750 nur gegen einen N. N. spielbar.

Fallen sind ein wichtiger Bestandteil des von J. v. d. Herik und J. Uiterwijk von der Universität Maastricht ausgearbeiteten „Opponent Models".

Das v. d. Herik/Uiterwijk Modell ist bisher ein rein akademisches Konzept. Ich kenne kein Programm, das Fallen stellt. Bei der Implementierung von Fallen ergeben sich zwei Probleme. Das Programm muss sicher sein, dass es um zwei Halbzüge tiefer als der Gegner rechnet. Der menschliche und maschinelle Suchprozess ist aber nicht vergleichbar. Manchmal rechnen Menschen tiefer als Programme, manchmal rechnen sie im Grunde gar nicht und verlassen sich auf ihr Gespür. Ein Programm hat ferner nicht die geringste Ahnung, was ein einleuchtender bzw. ein schwierig zu findender Zug ist. Auch wenn ein Mensch um zwei oder mehr Plies weniger tief rechnet, kann er intuitiv noch immer die richtige Antwort finden. Ein Programm, das Fallen stellt, würde mich bei einem Turnier sehr nervös machen. Man müsste Fallen sicherlich auf wenige Situationen – etwa in ohnehin kaputter Stellung oder in Zeitnot des Gegners – beschränken.

Schwindeln

Schwindeln ist das Gegenstück zur Falle. In einer Falle ist der augenscheinliche Zug nicht der beste, in einer Schwindel-Position der beste Zug nicht augenscheinlich. In der Partie gegen DUCHESS hätte KAISSA mit Kg7 geschwindelt, weil der Gegenzug Df8† nicht leicht zu sehen ist. Die Implementierung dieses Konzeptes ist komplizierter als man denkt. Das Programm hat – wie bei der Falle – keinerlei Begriff davon, was für einen Menschen augenscheinlich ist. Der Zug Df8† in DUCHESS – KAISSA wird von einem modernen Programm in Sekundenbruchteilen gefunden. Warum sollte ein Zug, den man selbst sofort auf das Brett knallt, schwierig sein? Ein zweites Problem ist der Alpha-Beta-Algorithmus. In jener Stellung weiß KAISSA nur, dass es nach Df8† mattgesetzt wird. Nach der Alpha-Beta-Logik ist es unsinnig, auch noch die anderen weissen Gegenzüge zu analysieren. Es könnten daher auch eine Reihe von anderen Zügen zumindest zu Turmverlust führen. Nachdem aber nach 34. ... Te8 die Partie auf alle Fälle verloren ist, wäre der Aufwand für die Berechnung dieser zusätzlichen Informationen möglicherweise gerechtfertigt.

Zeitnot

Sollte ich einmal vor die Aufgabe gestellt werden, ein Programm von einem menschlichen Spieler unterscheiden zu müssen, würde ich nicht auf die Züge, sondern auf die Zeiteinteilung schauen. Bei der Zeiteinteilung merkt man am deutlichsten, dass Programme keinen Begriff davon haben, welche Stellung schwieriger oder leichter ist. Obwohl der Zeitalgorithmus sehr wichtig ist, gibt es darüber fast keine Publikationen. Auch im persönlichen Gespräch mit anderen Programmierern erfährt man meist nur, dass die Zeiteinteilung ein Kapitel für sich ist. Meines

Wissens sind die meisten Programme auch bei der Zeiteinteilung Autisten. Sie beziehen die Uhr des Gegners nicht mit ein. In der Zeitnotphase des Gegners ist dies sicher nicht optimal. Wenn das Programm in dieser Phase noch genügend Zeit hat, verschafft das eigene Nachdenken dem Gegner eine Verschnaufpause. Möglicherweise wäre es in der Zeitnot des Gegners manchmal sogar sinnvoll, den zweitbesten Zug zu spielen. Der Gegner würde aus dem Konzept gebracht und verliert weitere Zeit. Ein Hindernis für dieses Konzept ist wieder Alpha-Beta. Das Programm kennt nur den besten Zug, von den anderen weiß es nur, dass sie schlechter sind. Die Reihenfolge der anderen Züge ist willkürlich. Sobald irgendein Widerlegungszug gefunden wird, bricht Alpha-Beta die Variante ab. Es muss dies keineswegs die beste Widerlegung sein. Man könnte in dieser Phase mit dem n-Best Modus rechnen. Das bedeutet aber zusätzlichen Rechenaufwand. Bei gleicher Suchtiefe fällt die Antwort des Programmes langsamer aus. Zusätzliche Zeit ist in dieser Phase das kostbarste Gut des Gegners.

(K)Eine Revolution?

Forster ist selbst von der Wirksamkeit seines Ansatzes nicht sehr überzeugt. Er schreibt in der Einleitung: „Die in dieser Arbeit beschriebenen Techniken haben nicht die Absicht, das Computerschach zu revolutionieren. Es ist höchst unwahrscheinlich, dass sie für Schachmaschinen vom DEEP-BLUE-Typ von großem Vorteil sind. Sie sollten aber für PC-Programme interessant sein“ (Übersetzung aus dem Engl. vom Verf.).

In diesem Punkt bin ich gänzlich anderer Meinung. Ein Modell des Gegners in das Programm zu integieren, wäre ein Paradigmenwechsel. Wie die jüngsten Ergebnisse von FRITZ zeigen, haben sich einige Spitzenspieler bereits sehr gut auf die Programme eingestellt. Ein mit dem menschlichen Faktor geschickt kalkulierendes Programm würde diese Spieler stark verunsichern. Ein nach dem Forster-Ansatz spielendes Programm wäre auch ein wesentlich interessanterer Gegner. Ich kann auch keinen wesentlichen Unterschied zwischen DEEP BLUE und NIMZO oder FRITZ sehen. DEEP BLUE ist „nur“ um einiges schneller.

Im Lichte dieser Analyse stellt sich für mich die Frage, ob der Liebe Gott wirklich so gütig und lieb ist. Zweifellos wäre es für Steinitz & Co. schon schwierig genug, gegen eine 32-Steiner-Thompson-Datenbank remis zu halten. Gegen einen alle menschlichen Schwächen ausnutzenden perfekten Gegner dürfte die Auseinandersetzung hoffnungslos sein.

Warum macht ein gütiger Gott dem Menschen die Aufgabe noch schwieriger, als sie objektiv ohnehin schon ist? Warum stellt er Eva erst eine gemeine Falle und lässt dann in Sippenhaftung auch alle Nachfahren dafür büßen? Warum muss auch die Schlange daran glauben?

L. Gott?

Schließlich war sie sein Werkzeug bei der Ausführung dieses teuflischen Planes. Möglicherweise hat L. Gott diese Strategie im Gegensatz zum Thompson-Gott notwendig, weil er tatsächlich nicht allwissend ist. So sagt er in der Genesis zu Eva: „Du hast Verlangen nach deinem Mann; er aber wird über dich herrschen.“ Das ist ein ziemlicher Patzer. Offensichtlich kann L. Gott Essenz und Akzidenz nicht unterscheiden. Wohl spielt sich so mancher Adam als häuslicher Herrscher auf, die Hosen hat aber meistens doch die Eva an. Auch mit Evas Begehr ist es meist nicht weit her.

Verwendete Informationen:

[1] R. FORSTER: *Introducing the Human Factor in Computer Chess Programs*, Semesterarbeit in Computer Science, Zürich, Dezember 1998.

[2] D. BILLINGS: „Thoughts on RoShamBo“, in: *ICGA-Journal*, Band 23, Nr. 1, März 2000.

[3] ST. JAKOB: Persönliche Kommunikation über das Programm Hossa.

[4] M. EHN: Persönliche Kommunikation über W. Steinitz.

Noch ein Hinweis zum Weiterlesen, auf das ultimative Steinitz-Buch:

[5] KURT LANDSBERGER: *William Steinitz, Chess Champion*. A Biography of the Bohemian Caesar; Jefferson 1993.