Early Chess Programs

Seminar Knowlegde Engineering und Lernen in Spielen SS04

Von: Jürgen Georg Lugsch

Inhalt

- Turing‘s Schachsimulation von 1951
- NSS Chess Program von 1958
- Greenblatt Chess Program von 1967

Turing‘s Hand –Simulation: Vorüberlegungen

- Kann man eine Maschine bauen, die Schach spielen kann?
- Turing fragt vorsichtiger (1950): „ Kann man eine Maschine bauen, die einigermaßen Schach spielen kann, d.h. konfrontiert mit einer gewöhnlichen Schachposition, nach 2-3 Minuten einen annehmbaren Zug berechnet? “
- Seine Antwort: „Ja, aber je besser sie spielen soll, desto komplexer wird sie und umso erfinderischer muss der Designer sein.“

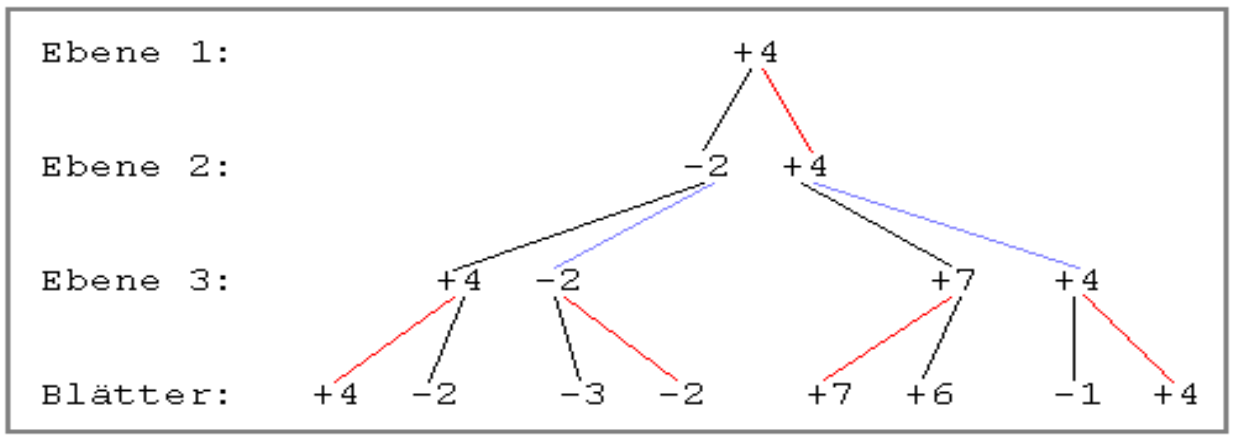
Konstruktion einer Maschine, die Schach spielt

- 1.) Annahme: Wenn man eindeutig mit einer Sprache und mathematischen Symbolen eine Berechnung beschreiben kann, lässt sich auch ein Computer derart programmieren, dass er diese Berechnung durchführt.

- 2.) Annahme: Computer ist unendlich schnell und es steht unendlich Speicher zu Verfügung

1. Ansatz: primitives MINIMAX

- Betrachte alle Zugmöglichkeiten von Weiß aus gesehen sukzessiv von gegebener Position aus bis zum *Ende*, dann MINIMAX Algorithmus
- Minimax allein sei bei Schach nicht praktikabel
- Turing‘s 1. Minimax hat nur 1, 0,-1 als Werte



2.Ansatz: selektiver sein

- 4 Halbzüge weit *alle* Folgepositionen einer Position bestimmen. Endpositionen dann nach bestimmten Kriterien bewerten
- Die Werte für vorherige Positionen lassen sich mit Minimax bestimmen
- Züge, die zu den Positionen mit den „besten“ Werten führen, werden gewählt
- Wäre theoretisch machbar, dass keine zwei Positionen denselben Wert haben

Beispielbewertung für 2. Ansatz:Material Auswertung

- Einfach, jedoch nicht eindeutige Werte
- Numerische Werte: 1 für Bauer, 3 für Springer etc.
- Material Weiß/ Material Schwarz Maß für den Wert einer Position
- Nachteil vom 2. Ansatz: Beachtet nur Material, allen Zügen wird gleich weit nachgegangen

Turing‘s Regeln für seine Handsimulation (1)

- Considerable moves

- Es werden immer nur max. 4 Halbzüge betrachtet
- Generell wird *jede* Zugmöglichkeit für Weiß und die Antwort von Schwarz betrachtet
- Zugmöglichkeiten, bei denen Figuren geschlagen werden können, haben höchste Priorität, besonders wenn Figuren nicht bzw. unzureichend gedeckt sind oder eine Figur mit niedrigem Wert gegen eine mit hohem getauscht werden kann
- Züge, die zu einem Schachmatt führen, haben die allerhöchste Priorität

Turing`s Regeln für seine Handsimulation (2)

- Dead Position
- Die Analyse einer Position ist tot, wenn sie in den nächsten 2 Zügen keine der Kriterien (hier die considerable moves) erfüllt
- Value of Position
- Wert einer Dead Position = W/B
- Falls keine Dead Position, d.h. Weiß am Zug, der größte Wert der „considerable moves“ und falls *alle* Züge „considerable“, Wert wie Dead Position. Der selbe Prozess für die Züge von Schwarz, nur wählt die Maschine hier den kleinsten Wert von Schwarz

Turing‘s Regeln für seine Handsimulation (3)

- Position-Play Value of a Piece
- Jede weiße Figur bekommt einen bestimmten Wert zugeteilt, den sie in einer Position hat, ebenso der schwarze König
- Teilwert für Figur ist z.B. die Quadratwurzel der Anzahl Züge, die eine Figur machen kann. Andere Werte ergeben sich aus Spielsituationen, z.B. Wert 1.0, falls Figur gedeckt oder 1.5, falls doppelt gedeckt etc. Teilwerte werden addiert und ergeben Position-Play Value einer Figur
- Zugwahl
- Gewählter Zug hat größte „ Value of Position“ (Vorrang vor PPV) und konsistent dazu die größte „Position-Play Value“

Spielstärke und Resümee

- Maschine spielt sehr schlecht und verliert auch gegen miesen Spieler
- Zu wenig Regeln, im Endspiel schwach
- Turing meint, spielt schlecht, weil sie ansatzweise so spielt wie er (spielt schwach)
- Turing auch überzeugt, selektives Vorgehen besser als Brute–Force und Geschwindigkeit

NSS Chess Program

- Schach Programme werden immer mehr durch Militär gefördert
- Hoffnung auf Erschaffung einer K.I.
- Vorbild Mensch, Selektivität sei beste Option, um Geschwindigkeit komplexer Programme zu erhöhen

Basic Organization of NSS

GOALS

GOAL Specification	**Move Generator**	**Static Evaluation**	**Analysis Generator**
King Safety			
Material Balance	implementiert	implementiert	
Center Control	implementiert	implementiert	
King-Side Attack			
Promotion			

Goals

- Voneinander unabhängig, einzelne goals können hinzugefügt und entfernt werden
- Bei Zugbeginn wird einleitende Analyse der Brettsituation durchgeführt, die geeignete Menge von Zielen hervorbringt
- Stehen in Liste, wichtigste goals zuerst
- Bestimmen Restablauf des Zug-Prozesses und im Endeffekt den Spielverlauf

Move Generation

- Zu *einem* goal gehöriger move generator schlägt Züge vor, auf Teilaufgaben spezialisiert
- z.B. nur Center Control Generator würde Bauer d2-d4 vorschlagen

- Welcher Zug der „move generators“ genommen wird, entscheiden separate „analysis generators“, welche Konsequenzen und weiterführende Züge der neuen Position prüfen

Auswertung eines Zuges

- Erkunden von Folgezügen bis zu einer gewissen Tiefe des Baums
- Analysis move generators bestimmen dabei, welche Verzweigungen im Suchbaum genommen werden
- An der Endposition einer Zugfolge bestimmen „static evaluation“ Routinen der goals Werte als Komponenten eines Vektors
- Effektiver Wert der Züge wird dann per „minimaxing“ der static values bestimmt

Final Choice

- Welcher Zug der alternativen Züge wird genommen ?
- Viele Programme verwenden Version von minimax, bei der Zug mit größtem Wert genommen wird
- NSS verwendet von Hand eingestellte Akzeptanzstufen - der erste Zug, der Stufe erfüllt, wird genommen
- Wenn kein Wert Stufe erfüllt, wird größter gefundener Wert genommen

Analysis Generators

- Basieren auf Turing‘s Konzept der Dead -Position
- Statische Wertung eines goals nur von Bedeutung, falls die Position „dead“ im Hinblick auf die Eigenschaft des goals ist
- Analyse Generator prüft, ob Position „dead“ in Bezug auf ein goal ist, ansonsten kreiert er neue plausible Züge , die das goal erfüllen sollen
- Position, die im Bezug auf **alle** goals tot ist, kann entgültig ausgewertet werden.

Beispiel für Zugfolge (1)

- Center Control Goal
- Immer aktiv, bis Bauern (d2 und e2) zur 4. Reihe gezogen sind

- Move Generator

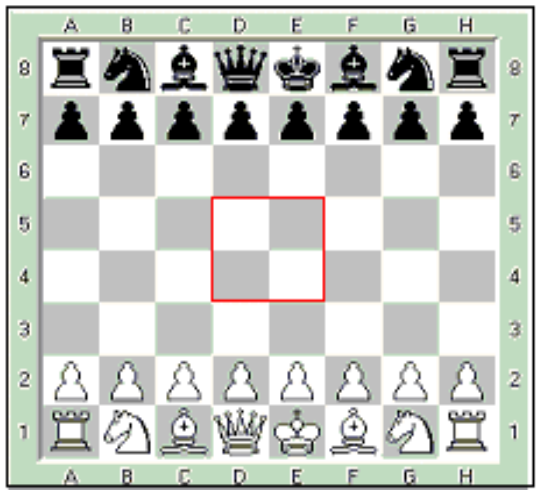
1. Bewege Bauer von d2-d4, e2-e4 (Primärzüge)
2. Verhindere äquivalente Züge des Gegners
3. Unterstütze Ausführung der Primärzüge, z.B. Decken

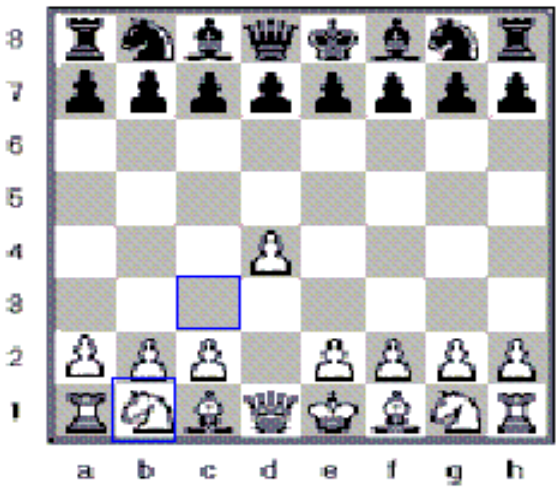
Beispiel für Zugfolge (2)

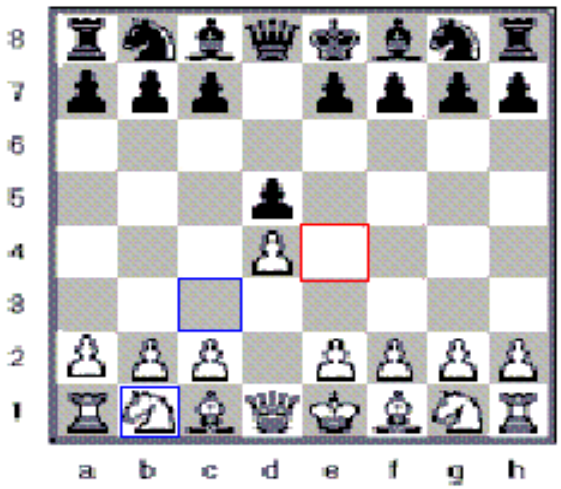
- Statische Auswertungsfunktion
- Zähle Anzahl Blockierungen der primary moves

- Analysis Move Generator
- Hier keiner, statische Auswertung reicht, Position „dead“

Beispiel für Zugfolge (3)







1. Aktiv Center C. Goal 2. Blau Möglichkeit Defender 3. Aktiv Center C. Goal

NSS Struktur kurzgefasst

- Eine Routine spezifiziert goals im Bezug auf eine gegebene Position
- Ein move generator schlägt Züge vor, die das goal erfüllen sollen
- Ein Routine führt eine statische Auswertung für jede Position durch, die in Beziehung zu einem goal steht
- Ein Analysis Generator betrachtet Folgezüge, die eine Position in eine „Dead Position“ überführen sollen

Vergleich Turing vs NSS

	Turing Simulation	**NSS**
Date:	1951	1958
Board:	8*8	8*8
Computer:	Per Hand	RJ 20.000 ops/sec
Programm		
Alternativen:	Alle Züge	Move Generators
Static Evaluation:	Numerisch, viele Faktoren	Vektoren,Akzeptanz nach goals
Integration Werte:	Minimax	Minimax
Final choice:	Material dominiert	First acceptable
Spielerfahrung	1 Spiel, hilflos	0 Spiele, Programm nicht fertiggestellt

Greenblatt Chess Program

- Entwickelt 1966-1967 im Artificial Intelligence Laboratory des M.I.T
- Entwicklungsumgebung verschlang Geld – 256 KB Ram, Grafikdisplay (1967!!)
- Programm wurde nur auf Schach hin entwickelt, nicht als allgemeines Problemlösungssystem

Programmstruktur

- Organisiert über minimax Suche in Spielbaum: Äste Züge, Knoten Position
- Move Generator führt jeden legalen Zug auf und weißt Plausibilitätswerte zu
- Position Evaluator berechnet numerische Werte für die Positionen
- Weiß am Zug: Positiver Wert bedeutet Weiß im Vorteil
- Wert im Baum wird einen Level höher gereicht, bis Wert für aktuelle Position bestimmt ist: Sequence bester Züge heißt Principal Variation

Plausible move generator

- Wählt Untermenge legaler Züge für Aufbau des „Move-Tree“
- Sortiert Züge – notwendig für effektives Alpha -Beta pruning
- 50 komplizierte u. verschachtelte Heuristiken, um Plausibilität des Zuges zu bestimmen

Programmbeschleunigung

- Für die Suche im Baum wird alpha -beta pruning verwendet – Programm schneller, nicht stärker
- Hash Coding: um nicht Position während des Suchprozesses mehrmals zu überprüfen, betrachtete Züge in Hashtabelle ablegen – sofort verfügbar
- Eröffnungsbibliothek von 5000 Zügen, entworfen von zwei M.I.T Studenten, die Großmeister waren

Secondary Search

- Um bessere Suchtiefe und Spielstärke bei minimal größeren Kosten zu erreichen
- Wird aktiv, falls normale Suche neuen Kandidat auf Top Level finden
- Principal Variation ganz runter gehen, die der move generator berechnet hatte, dann 2 Halbzüge weitersuchen und den neu berechneten Wert nehmen

Spielstärke

- Gewinnt 80% gegen Durchschnittsspieler
- Gegen Turnierspieler verliert es ständig

Turnier 1967	WON	LOST	Drew	Rating Gegner	Version
FEB	o	4	1	1243	1.0
MAR	1	4	0	1330	1.1
APR	2	0	2	1450	1.2
MAY	0	4	0	1400	1.2

Quellen

- Alan Turing: „Chess“ first published in „Faster than Thought“, pp.286-295.London, Pitman (1953)
- Allen Newell, Cliff Shaw, Herbert Simon: „Chess Playing Programs and the Problem of Complexity“, IBM Journal of Research and Development, volume 2, October 1958 pp.320-335.
- Richard D. Greenblatt, Donald E. Eastlake and Stephen D. Crocker: „The Greenblatt Chess Program“. First published for the Fall Joint Computer Conference 1967 pp. 801-810.