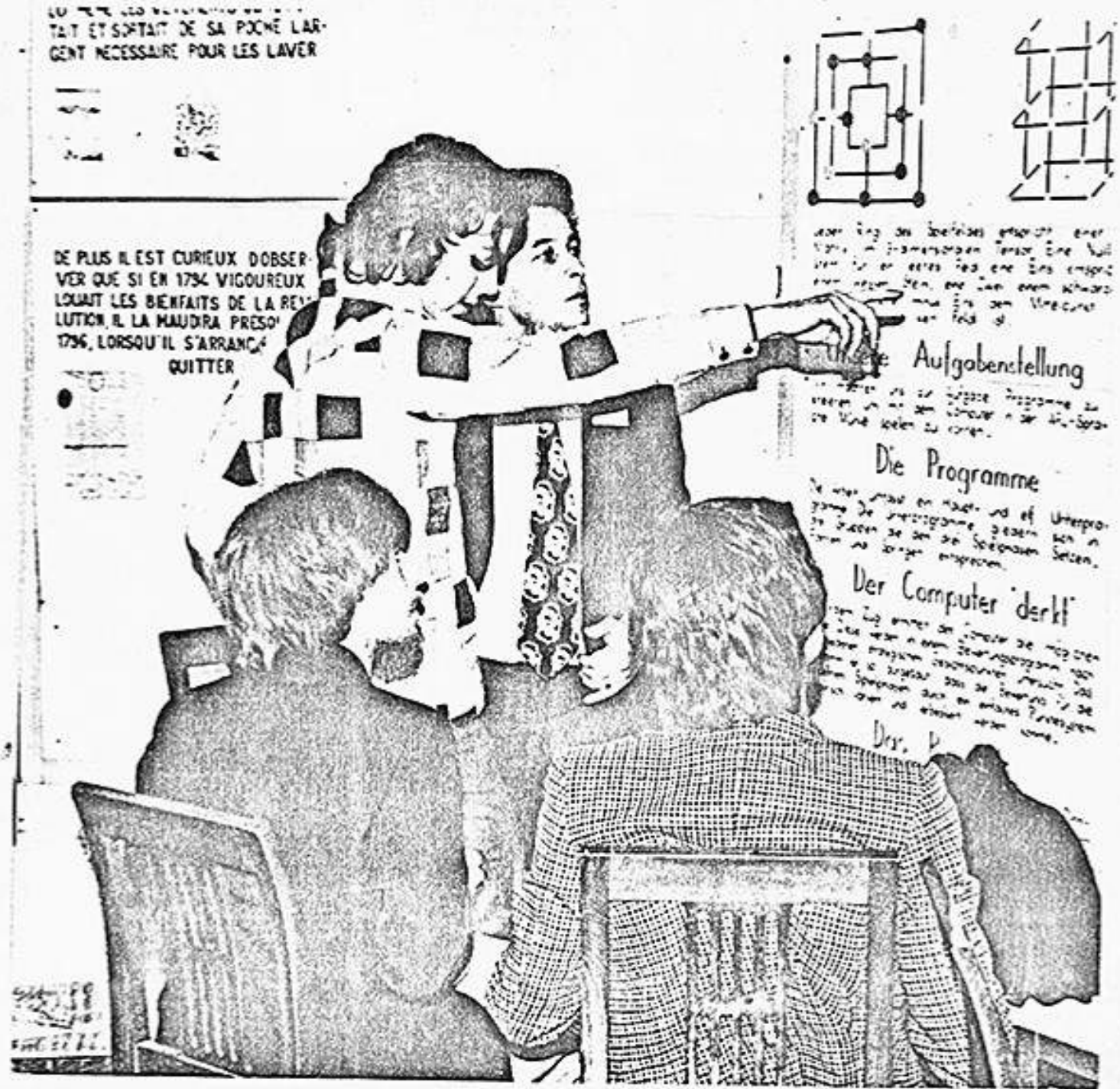
Schweizer Jugend forscht

La science appelle les jeunes

Scienza e gioventù

9. Jahrgang Nr. 2 März/April 1976



P. Vernazza, M. Bärtschi

Mühlespiel mit dem Computer

Das Resultat der Arbeit von Pierino Vernazza (1956), Herrliberg, und Martin Bärtschi (1956), Zürich, beide Schüler des Realgymnasiums Rämibühl, ist ein APL-Programm, bestehend aus einem Hauptprogramm und Unterprogrammen, das den Computer zum vollwertigen Mühlespieler macht. Es setzt sich aus den Teilen Setzen, Fahren, Springen zusammen. Zentral steht das Bewertungsprogramm, womit jeweils der optimale Zug gewählt wird. Aber gerade hier stösst das Programm an seine Grenzen in bezug auf Rechenzeit und Speicherplatz. Der Grund liegt in der mangelnden Erfahrung zu Beginn der Arbeit. Die Verfasser haben jedoch das Problem voll erkannt und gerade hier richtigerweise die ehrgeizige Aufgabe etwas eingeschränkt.
Der Erfolg der Arbeit liegt in einem korrekt spielenden Mühlespiel-Programm. Beim heutigen Stand ist der Computer nicht der «beste», aber ein guter und vor allem ein korrekter Spieler.
Hervorragend sind die eigene originelle Idee der Aufgabe, das logische Denken und der Fleiss, womit die Aufgabe erfolgreich gelöst wurde. *F. Kuhlen, Dietikon*

Einleitung

Durch unser gemeinsames Interesse am Computer und am Schachspiel kamen wir auf die Idee, dieses Spiel im Rahmen unserer Semesterarbeit zu programmieren. Bald wandten wir uns aber dem einfacheren Mühlespiel zu, was sich als vernünftige Beschränkung erwies.
Das Mühlespiel ist ein Brettspiel für zwei Personen. Das Brett weist 24 untereinander verbundene Felder auf (vgl. Abb. 1).
Jeder Spieler bekommt neun Steine. Diese werden zuerst gesetzt, und darauf wird mit ihnen gefahren. Ziel des Spiels ist es, dem Gegner 7 Steine zu nehmen und vom Brett zu entfernen. Dies ist jedesmal dann erlaubt, wenn es gelingt, drei eigene Steine auf eine Gerade zu bringen. (Diese drei Steine werden Mühle genannt.)
Zu diesem Spiel gibt es praktisch keine theoretischen Abhandlungen. So bestand denn ein Teil unserer Arbeit darin, empirisch eine möglichst gute Strategie herauszufinden.
Unsere eigentliche Aufgabe war es nun, Programme in der APL-Sprache, der einzigen uns bekannten Computersprache, so zu erstellen, dass es möglich ist, mit dem Computer Mühle zu spielen, und die es uns erlaubten, die Strategie fortlaufend zu verbessern.

Vorgehen bei der Arbeit

Als erstes stellte sich die Frage, wie sich der Computer das Spielfeld «denken» soll. Wir wählten dafür einen dreidimensionalen Tensor (vgl. Abb. 2), dessen drei Matrizen je einem der drei Ringe (oder Quadrate) des Spielfelds entsprechen.
In diesem Tensor merkt sich nun der Computer die Steine: Eine Null bezeichnet ein leeres Feld, eine Eins einen weissen Stein und eine Zwei einen schwarzen Stein.
Das Spiel gliedert sich in drei Phasen: Setzen, Fahren und Springen. In allen drei Phasen sind wir prinzipiell gleich vorgegangen:

1. Berechnen aller nächstmöglichen Züge
2. Vergleichen und Beurteilen all dieser Züge
3. Ausführen des besten Zuges
4. Kontrolle, ob neue Mühle vorhanden

Das Beurteilen aller Züge geschieht in einem für alle drei Spielphasen gemeinsamen Unterprogramm. Es bewertet verschiedene Kriterien, die wir nach und nach herausfanden und hinzufügten. Diese werden mit Punktzahlen bewertet. Somit konnten wir durch Änderung der Punktzahlen die Strategie des Computers verbessern.
Am Schluss umfasste das ganze programmierte Spiel 12 Programme, die gemäss Abbildung 3 zusammenhängen.

Die Bewertung

Je nach der Spielphase (Setzen, Fahren oder Springen) werden die verschiedenen Stellungsvor- und -nachteile mit verschiedenen Punktzahlen bewertet. Es zeigte sich dann, dass es günstiger ist, das Spiel in mehr Spielphasen einzuteilen, die sich aber in der Bewertung nur geringfügig unterscheiden (besonders A–F). Die einzelnen Spielphasen sind:

A: Setzen: Computer und Spieler haben weniger als je 7 Steine gesetzt
B: Setzen: Computer 6, Spieler 7 Steine gesetzt oder umgekehrt
C: Setzen: Computer und Spieler je 7 Steine gesetzt
D: Setzen: Computer 7, Spieler 8 Steine gesetzt oder umgekehrt
E: Setzen: Computer und Spieler je 8 Steine gesetzt
F: Setzen: Computer 8, Spieler 9 Steine gesetzt oder umgekehrt
G: Computer und Spieler fahren
H: Computer fährt, Spieler springt
I: Computer springt, Spieler fährt
J: Computer und Spieler springen

Abbildung 1

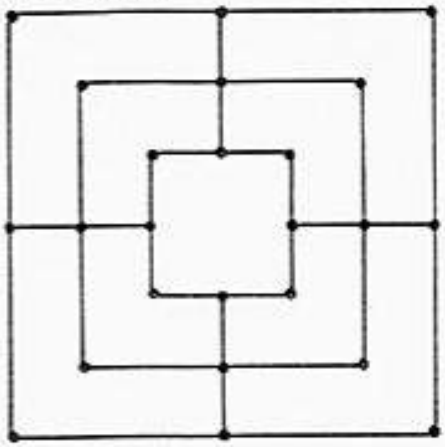


Abbildung 2

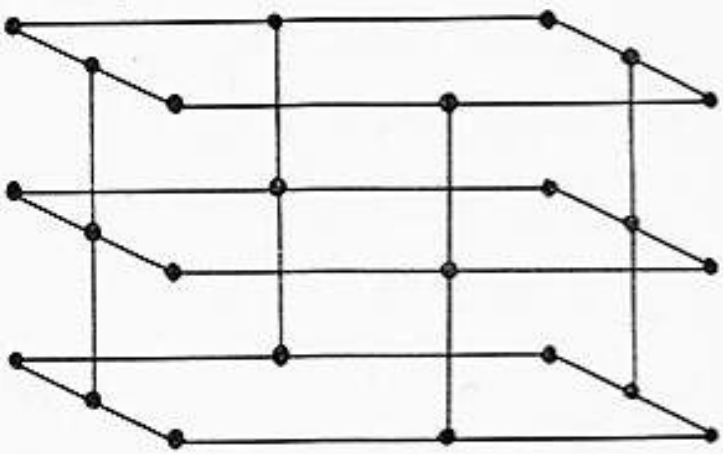
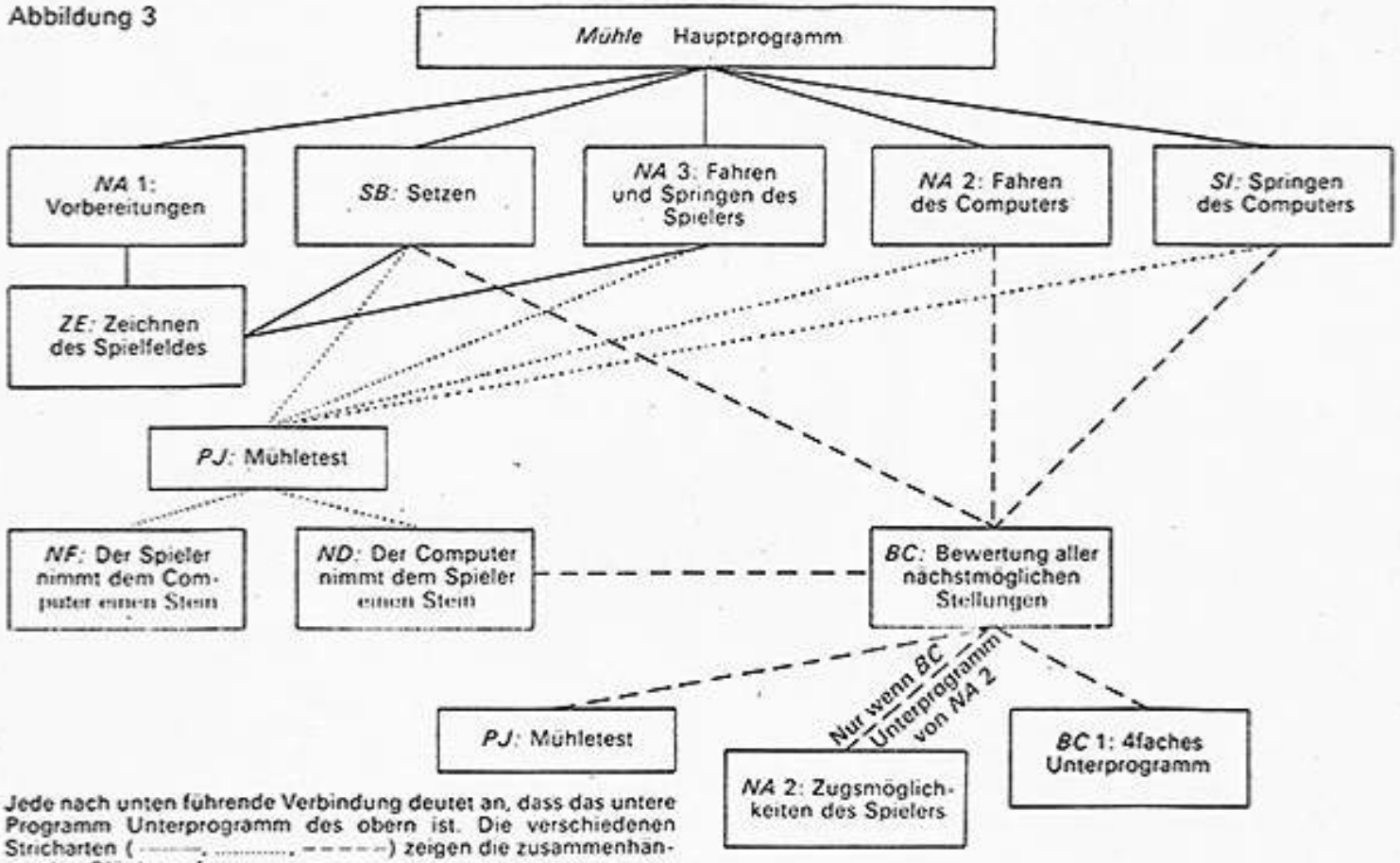


Abbildung 3



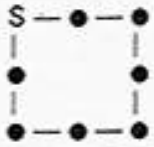
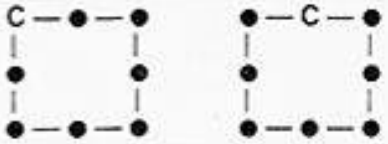
Jede nach unten führende Verbindung deutet an, dass das untere Programm Unterprogramm des obern ist. Die verschiedenen Strichar ten (——, ..., – – – –) zeigen die zusammenhangenden Blocke auf.

In Abbildung 4 sind die Punktzahlen für die Bewertung der wichtigsten Abschnitte wiedergegeben.

(Im folgenden bezeichnet C einen Stein des Computers und S einen Stein des Spielers.)

Abbildung 4

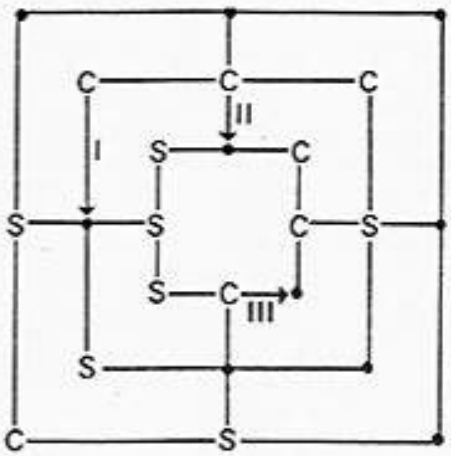
Kriterium: / Spielphase:	A	G	H	I	J
1. Anzahl Nebenplätze: pro Nebenplatz und Stein	1,5	1,5	1,5	1,5	1,5
2. C	14	0	0	0	0
3. Mühle des Computers C—C—C	120	40	40	0	0
4. Neue Mühle des Computers	0	120	a)	1000	1000
5. C—C—●, C—●—C	8	0	0	40	8
6. S—S—●, S—●—S	–90	0	–90	0	–900
7. C, ●, ●—●—C und gleichwertige Stellungen, wie zum Beispiel C—●—●, C	7	0	0	14	7
8. C, ●, C—●—C und gleichwertige Stellungen	60	10	10	50	800
9. S, ●, ●—●—S und gleichwertige Stellungen	–40	0	–30	0	–700

	A	G	H	I	J
10. 	–20	0	0	0	–600
11. 	15	0	0	2	2
12. Platz neben einer Mühle des Spielers	20	30	30	0	0
13. Anzahl offene Mühlen des Spielers – Anzahl neue Mühlen des Computers	0	–90	0	–900	0
14. Offene ungefährdete Mühle des Computers: – wenn der Spieler keine offenen Mühlen besitzt; – wenn Spieler und Computer je mindestens eine offene und eine geschlossene Mühle haben, – sonst (d.h. wenn der Spieler offene Mühle[n] hat)	0 0 0	90 45 0	90 45 0	0 0 0	0 0 0
15. Der Spieler hat x Zugsmöglichkeiten	0	b)	0	b)	0
a) 1000, wenn Spieler keine Mühle; sonst 0 b) 20 + (x + 0,01)					

Zwei Beispiele für die Bewertung:

Fall G: Wir betrachten drei für den Computer mögliche Züge (Pfeil) und wollen sehen, wie er diese mit seinem Programm bewertet.

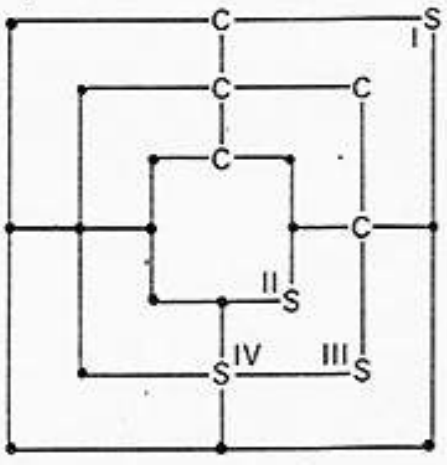
Ausgangsstellung:



Kriterium	Zug I	Zug II	Zug III
1	30	25,5	25,5
3	0	0	80
4	0	0	120
8	0	0	0
12	60	60	0
13	0	–90	0
14	180	0	0
15	2,85	2,22	1,82
Total	272,85	–2,28	227,32

Fall A: Sobald der Computer eine Mühle hat, muss er ja dem Gegner einen Stein nehmen. Im folgenden Beispiel zeigen wir, wie der Computer, nachdem er 5 Steine gesetzt hat, die Wegnahme von einem der 4 Steine des Gegners bewertet. Dies geschieht mit dem gleichen Bewertungsprogramm mit den gleichen Punktzahlen wie die normale Bewertung in dieser Spielphase.

Ausgangsstellung:



Kriterium \ Wegnahme von:	Stein I	Stein II	Stein III	Stein IV
1	24	24	24	24
2	28	28	28	28
3	120	120	120	120
5	8	8	16	8
6	–90	–90	0	0
7	0	0	0	0
8	0	0	60	0
9	–40	0	–40	0
10	0	0	0	0
11	15	15	0	0
12	0	0	0	0
Total	65	105	208	180

Ergebnisse und Verbesserungsmöglichkeiten

Wir haben mit diesen Programmen einen seriösen Spielpartner programmiert. Allzuviele Spiele gewinnt er aber dennoch nicht. Es stellte sich heraus, dass er vor allem defensiv spielt. Dies rührt daher, dass er nur einen Zug im voraus betrachtet. Wir hatten einmal ein Programm erstellt, mit dem er zwei und mit leichten Abänderungen sogar drei Züge hätte vorausberechnen können, doch wurden die Wartezeiten viel zu lange. Heute könnte dieses Problem mit dem erweiterten APL-SV zu einem grossen Teil beseitigt werden, da man die Programme mit den zeitraubenden Schleifen (v.a. das Bewertungsprogramm) direkt in die Maschinensprache übersetzen könnte.

Ein weiteres Problem war die Platzknappheit. Wir haben nämlich viele umfangreiche INTEGER-Variablen, an deren Stelle leicht Boolsche Variablen verwendet werden könnten, gebraucht, da uns zu jener Zeit die Unterscheidung in verschiedene numerische Variablentypen noch unbekannt war und in APL für kleinere Probleme auch nicht bekannt sein muss.

Allgemein gilt, dass Platz- und Zeitaufwand umgekehrt proportional voneinander abhangen, das heisst dass wir durch die Wiederverwendung des auf obige Weise gesparten Platzes auch beträchtliche Rechenzeit hätten gewinnen können. Die Strategie könnte wahrscheinlich stark verbessert werden, wenn es gelänge, sie durch eine Entscheidungstabelle zu bewerten. So könnten in einfacher Weise viel mehr Kombinationen von Bewertungskriterien berücksichtigt werden, als dies in den bestehenden Programmen der Fall ist. (Ein Verfahren, das wir damals noch nicht kannten!)

Abschliessend möchten wir festhalten, wie unterschiedlich Mensch und Maschine doch arbeiten. Der Computer vergleicht *alle* möglichen Züge nach einem bestimmten Schema. Er geht jeden Zug gleich an.

Der Mensch hingegen kann schon auf den ersten Blick auf das Spielfeld einige Züge auswählen und betrachtet nur noch diese genauer und trifft seine Wahl immer wieder nach neuen Gesichtspunkten. Er hat den Überblick über das ganze Geschehen. Dafür geschieht es, dass er einen Zug übersieht, was dem Computer nie passieren kann. Daher ist die Zugwahl des Menschen auch in einem so völlig rationalen Spiel nicht immer nur rational. Er hofft öfters, dass er mit einem bestimmten Zug erreicht, dass er dem Gegner eine Falle bauen kann, dass dieser ein kleines Detail übersieht. Diese Hoffnung bestimmt die Wahl des nächsten Zuges mit.