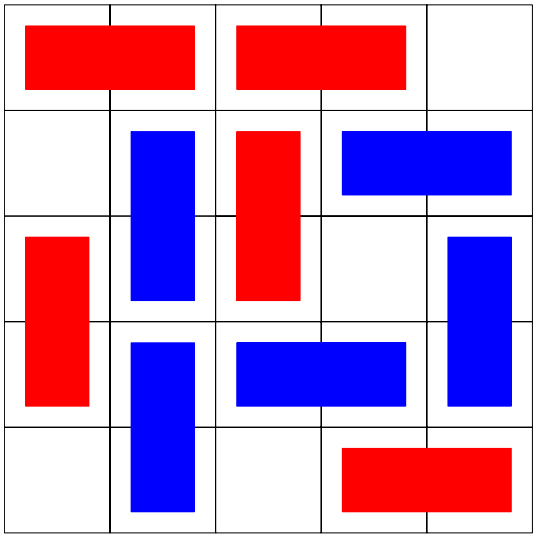
Masterarbeit

Das Spiel Juvavum



Martin Schneider, Bakk. rer. nat

2009

Danksagung

Mein Dank gilt meinen Eltern für ihre Geduld und Unterstützung sowie O. Univ. Prof. Dr. Peter Gerl für die Betreuung dieser Masterarbeit.

Inhaltsverzeichnis

1 Vorwort

Den Ausgangspunkt zu dieser Diplomarbeit bildete das Projektpraktikum im Rahmen des Bakkalaureatsstudiums Mathematik im Wintersemester 2004/05 sowie dem Sommersemester 2005 an der Universität Salzburg. Zur Auswahl standen damals zwei Lehrveranstaltungen, die unterschiedliche Themenbereiche behandelten.

Die Beschreibung des Projektpraktikums „Spiele“bei Prof. Gerl weckte dabei von Anfang an mein Interesse:

Jeder Teilnehmer erhält (oder erfindet) ein Spiel, für welches ein möglichst gutes (lernfähiges) Programm geschrieben werden soll, sodass der Computer ein guter (ernst zu nehmender) Mitspieler wird. Es handelt sich also um ein Teilgebiet der künstlichen Intelligenz.

Gerade als Student mit Zweitfach Informatik fiel mir die Auswahl daher nicht besonders schwer und auch rückblickend war es sicher die richtige Entscheidung.

Nach einer interessanten Einführungsvorlesung zum Thema wurden die Spiele Juvavum und Domino Juvavum vorgestellt, mit denen wir uns in weiterer Folge in Gruppen von 3 bis 4 Studenten beschäftigten. Das erste Ziel war es, gute Spielstrategien zu entwickeln und Juvavum bzw. Domino Juvavum mit Hilfe eines Computersprogramms zu analysieren. Danach sollte ein Programm geschrieben werden, das in der Lage ist, die beiden Spiele (gut) zu spielen.

Beide Ziele stellten sich jedoch schon bald als zu ambitioniert heraus und konnten am Ende nur teilweise erreicht werden. Umso größer war bei mir der Ehrgeiz und die Neugier, mich auch nach Abschluss des Projektpraktikums weiter mit dem Thema zu beschäftigen. Als Prof. Gerl das Spiel Juvavum Ende 2007 als Thema meiner Masterarbeit vorschlug, musste ich daher nicht besonders lange überlegen.

Das Ziel dieser Arbeit ist es, die Resultate aus dem Projektpraktikum auszubauen und vermehrt Zusammenhänge zwischen Juvavum und anderen kombinatorischen Spielen herzustellen.

Eine vollständige Lösung des Spiels steht zwar weiterhin aus, für viele Fälle können aber inzwischen Gewinnstrategien angegeben werden. Darüber hinaus werden einige grundlegende Bemerkungen zur Komplexität von Juvavum und Domino Juvavum präsentiert und entsprechende Resultate bewiesen.

Auch das im Rahmen des Projektpraktikums enstandene JAVA-Programm wurde stark erweitert. Es bietet nun neben diversen Funktionen zur spieltheoretischen Analyse von Juvavum-Spielen die Möglichkeit, gegen verschiedene Computergegner anzutreten, die in vielen Fällen in der Lage sind, perfekt zu spielen. Das Programm liegt dieser Arbeit (inklusive aller Sourcecodes) bei und wird im Anhang ausführlich beschrieben. Darüber hinaus kann es von meiner Homepage[1] heruntergeladen werden.

Den Anfang der Arbeit bildet eine Einführung in den Bereich der kombinatorischen Spieltheorie, in dem an Hand einiger Beispiele die Begriffe gebildet und Theorien erarbeitet werden, die im weiteren Verlauf auf das Spiel Juvavum angewandt werden.

[1] http://www.mschneider.cc/projects/juvavum/

2 Motivation und Grundlagen der kombinatorischen Spieltheorie

2.1 Was ist ein Spiel?

Bevor wir uns der Analyse des Spiels Juvavum widmen, beschäftigen wir uns zunächst mit den Grundlagen sowie einigen Beispielen der kombinatorischen Spieltheorie.

Beispiel 2.1 (Datumsspiel) Wir betrachten dazu zuerst folgendes Spiel[2]: Gegeben sei ein zufällig gewähltes Datum im Monat Jänner, das die Startposition des Spiels darstellt. Zwei Spieler ziehen nun abwechselnd, wobei ein Zug darin besteht, entweder den Tag oder den Monat beliebig zu erhöhen (nicht jedoch beides). Einzige Einschränkung ist, dass nach jedem Zug ein gültiges Datum erreicht wird (z. B. sind der 31. April, der 31. September, aber auch der 29. Februar[3] nicht erlaubt). Sieger ist, wer den 31. Dezember erreicht.

Ein Beispiel: Angenommen, das Spiel beginnt am 4. Jänner. Gültige erste Züge sind alle weiteren Tage im Jänner (5., 6., ..., 31.) sowie der 4. Tag jedes anderen Monats (4. Februar, 4. März, ..., 4. Dezember). Es stellen sich nun einige Fragen: Was ist ein guter erster Zug? Gibt es eine Gewinnstrategie? Wie sieht diese aus? Ist bei diesem Spiel derjenige im Vorteil, der beginnt oder jener, der als zweiter zieht? Welchen Einfluss hat die Auswahl der Startposition?

Wir werden im Laufe der Einführung auf dieses Spiel zurückkommen und sämtliche Fragen beantworten.

Die Spiele, die wir im Folgenden betrachten, werden mit verschiedensten Hilfsmitteln (Spielfiguren etc.) auf Spielbrettern unterschiedlicher Form und Größe gespielt. Sie haben verschiedene Spielpositionen sowie Regeln, die festlegen, auf welche Positionen man von einer gegebenen Position aus ziehen kann. Unter den Spielpositionen gibt es eine Startposition (von der aus der 1. Spieler das Spiel eröffnet) sowie eine oder mehrere Zielpositionen, von denen kein weiterer Zug mehr möglich ist.

In diesem ersten Abschnitt legen wir zunächst die Klasse jener Spiele fest, die wir betrachten möchten, und führen einige wichtige Begriffe zur Untersuchung solcher Spiele ein.

Definition 2.2 (Spielzug) Unter einem Spielzug verstehen wir das (regelkonforme) Ziehen von einer Spielposition zu einer anderen.

Definition 2.3 (Zweipersonenspiel) Ein Spiel, bei dem zwei Spieler abwechselnd ziehen, heißt Zweipersonenspiel.

Definition 2.4 (Vollständige Information) Unter einem Spiel mit vollständiger Information verstehen wir ein Spiel, bei dem alle zum Spiel gehörigen Elemente (Spielsteine, Karten usw.) offen liegen, d. h. für alle Mitspieler sichtbar sind.

[2] Dieses Spiel wird in [Bog07] als *Date Game* beschrieben. [3] Wir spielen also nicht in einem Schaltjahr.

Definition 2.5 (Kombinatorisches Spiel) Ein Spiel mit vollständiger Information und ohne Zufallseinfluss heißt kombinatorisch.

Nicht unter obige Definition fallen zum Beispiel all jene Spiele, bei denen ein Würfel oder ähnliches zum Einsatz kommt (Zufallseinfluss). Ebenso sind alle Spiele mit verdeckten Karten (Poker, Bridge, Cluedo usw.) keine kombinatorischen Spiele (da keine vollständige Information gegeben ist).[4]

Definition 2.6 (Endliches Spiel) Ein Spiel heißt endlich, wenn stets nach einer endlichen Anzahl von Zügen eine Zielposition erreicht wird.

Definition 2.7 (Stark endliches Spiel) Ein Spiel heißt stark endlich, wenn es endlich ist und es zudem von jeder Spielposition nur endlich viele Zugmöglichkeiten gibt.

Beispiel 2.8 (Münzen legen) Münzen legen ist ein Beispiel für ein Spiel, das zwar endlich, aber nicht stark endlich ist. Dabei legen zwei Spieler abwechselnd Münzen auf ein Spielbrett. Diese dürfen sich dabei weder überlappen noch über den Spielfeldrand hinausragen. Das Spiel ist zu Ende, sobald kein weiterer Zug mehr möglich ist (d. h. keine weitere Münze Platz hat, ohne eine andere zu überdecken oder über das Spielfeld hinauszuragen). Der Spieler, der den letzten Zug macht, gewinnt. Man erkennt leicht, dass dieses Spiel stets nach einer endlichen Anzahl von Zügen endet. Es ist allerdings nicht stark endlich, da die Anzahl der möglichen Positionen für eine Münze auf dem Spiefeld entweder 0, 1, in vielen Fällen aber auch unendlich ist. Sofern das Spielfeld größer ist als die kleinste vorhandene Münze (was eine sinnvolle Bedingung ist, da ansonsten kein einziger Zug möglich ist), gibt es z. B. bereits für den ersten Zug unendlich viele Möglichkeiten.

Definition 2.9 (Neutrales Spiel) Ein Spiel heißt neutral, wenn alle Spieler die gleichen Zugmöglichkeiten haben.[5]

Das Datumsspiel ist neutral. Nicht neutral sind zum Beispiel alle Spiele, bei denen die Spieler nur ihre eigenen Spielsteine bewegen dürfen (Schach, Go, Mensch ärgere dich nicht, Vier gewinnt usw.).

Im Folgenden verstehen wir unter einem Spiel ein stark endliches, neutrales Zweipersonenspiel.[6]

[4] Arbeiten über die Theorie einiger Spiele ohne vollständige Information findet der interessierte Leser unter dem Begriff *Knowledge Games* (z.B. [Dit00]) [5] In der angelsächischen Literatur spricht man von *impartial games* im Gegensatz zu *partisan, partizan, partial* oder auch *unimpartial games*, bei denen sich die Zugmöglichkeiten der einzelnen Spieler unterscheiden. [6] Die einzigen Ausnahmen sind das Spiel Münzen legen (nicht stark endlich) und Domineering (nicht neutral), auf das wir in Kapitel 2.5.5 kurz eingehen werden.

2.2 Spielende und Sieger

Jedes Spiel endet, sobald eine Zielposition erreicht wird, was auf Grund der Beschränkung auf endliche Spiele stets der Fall ist, oder aber sobald einer der beiden Spieler aufgibt[7]. Im letzteren Fall verliert natürlich jener Spieler, der aufgegeben hat (und der andere gewinnt). Für den anderen Fall treffen wir folgende Konvention: Sieger ist jener Spieler, der den letzten Zug macht.

> Das ist sicherlich eine natürliche Konvention, denn üblicherweise betrachten wir uns schon als Verlierer, wenn wir keinen guten Zug mehr machen können; umso eher sollten wir verlieren, wenn wir überhaupt keinen Zug mehr machen können![8]

Dennoch ist es naheliegend, auch den umgekehrten Fall zu betrachten, in dem jener Spieler gewinnt, der zuerst keine Zugmöglichkeit mehr hat. Wir definieren dazu zwei Varianten eines Spiels:

Definition 2.10 (Normales Spiel) Sei G ein Spiel. G heißt normal, wenn jener Spieler gewinnt, der den letzten Zug macht.

Definition 2.11 (Misère-Form eines Spiels) Ein Spiel $M(G)$, das dieselben Regeln und Zugmöglichkeiten wie ein Spiel G hat, heißt Misère-Form oder Misère-Spiel von G, wenn nicht der Spieler gewinnt, der den letzten Zug macht, sondern jener, der zuerst nicht mehr ziehen kann. Im Misère-Spiel verliert also, wer den letzten Zug macht.

Bemerkung 2.12 Sowohl im normalen Spiel als auch in der Misère-Form gilt Zugzwang: Wenn ein Spieler eine Zugmöglichkeit hat, muss er auch einen Zug machen. Nicht zu ziehen (z. B. weil man sonst den letzten Zug in einem Misère-Spiel machen müsste), ist nicht erlaubt.

Bemerkung 2.13 In beiden Fällen ist einer der Spieler eindeutig als Sieger festgelegt. Ein Unentschieden ist nicht möglich.

Bemerkung 2.14 Obwohl der Unterschied zwischen dem normalen und dem Misère-Spiel auf den ersten Blick trivial erscheint, gestaltet sich die Analyse eines Misère-Spiels im Allgemeinen wesentlich schwieriger. In der Tat gibt es bei der spieltheoretischen Analyse eines Spiels oft weitaus mehr Resultate für die normale Form als für die Misère-Variante.

Wie wir in Kapitel 7.1 sehen werden, gibt es zum Beispiel für das Spiel Juvavum auf einem $2n \times 2m$-Spielfeld eine sehr einfache Gewinnstrategie für den 2. Spieler. Hingegen ist für das entsprechende Misère-Spiel kein Resultat für allgemeine m und n bekannt.

---

[7] Für die Theorie ist die Aufgabe eines Spielers nicht wesentlich, da wir uns nur dafür interessieren, welcher Spieler in einer bestimmten Position im Vorteil ist. Ein Spieler wird meist nur dann aufgeben, wenn er weiß (bzw. es offensichtlich ist), dass er (auch bei perfektem Spiel) verlieren wird. Sobald dies klar ist, ist es für uns unerheblich, ob und wie das Spiel zu Ende gespielt wird. [8] [Con83], S.56

Bemerkung 2.15 Wir haben zu Beginn dieses Abschnitts John H. Conway zitiert. Er argumentiert (wie viele andere auch), dass es eine natürliche Konvention darstellt, dass derjenige Spieler gewinnt, der den letzten Zug macht. Vielleicht suchen Mathematiker in diesem Fall aber auch nur nach einer Legitimation, das als nahe liegend zu bezeichnen, wofür es eine einfache und elegante Theorie gibt. Thane E. Plambeck, der einige wichtige Arbeiten über Misère-Spiele geschrieben hat, gibt hingegen Gründe dafür an, die Misère-Variante eines Spiels als die natürlichere Spielform zu sehen:

> When nonmathematicians play impartial games, they tend to choose the misere play convention. [9] [...] But why do people prefer the misere play convention? The answer may lie in Fraenkel's observation that impartial games lack *boardfeel*, and simple *Schadenfreude*:
>
> > For many MathGames, such as Nim, a player without prior knowledge of the strategy has no inkling[10] whether any given position is "strong" or "weak" for a player. Even two positions before ultimate defeat, the player sustaining it may be in the dark about the outcome, which will stump him. The player has no boardfeel....[11]
>
> If both players are "in the dark," perhaps it's only natural that the last player compelled to make a move in such a pointless game should be deemed the loser. Only when a mathematician gets involved are things ever-so-subtly shifted toward the normal play convention, instead—but this is only because there is a simple and beautiful theory of normal-play impartial games—the Sprague-Grundy theory[12]. Secretly computing nim-values, mathematicians win normal-play impartial games time and time again. Papers on normal play impartial games outnumber misere play ones by a factor of perhaps fifty, or even more.[13]

Wir halten fest, dass sowohl die normale Form als auch die Misère-Form ihre Daseinsberechtigung haben und werden im Laufe der Arbeit — soweit möglich — beide analysieren.

Eine neue Theorie der Misère-Spiele stellt einen sehr jungen Teilbereich der kombinatorischen Spieltheorie dar. In dieser Arbeit wird darauf nur am Rande eingegangen werden. Dies liegt auch daran, dass sich die Ergebnisse nur bedingt zur Analyse von Juvavum heranziehen lassen. Für eine gute Einführung seien [Sie08], [Pla06], [PS08] sowie die Internetseiten [Pla07b] und [Pla07a] empfohlen.

Nach diesen grundlegenden Begriffsbildungen führen wir nun eine Reihe von Hilfsmitteln zur Analyse von Spielen ein.

---

[9] Ein Beispiel dafür ist das Spiel Nim, das wir in Kapitel 2.5.1 beschreiben. Die wohl bekannteste Nim-Variante ist das Spiel Marienbad, das durch den Film *Letztes Jahr in Marienbad* von Alain Resnais aus dem Jahr 1961 berühmt wurde. Übersetzt in unsere Bezeichnungen handelt es sich dabei um Misère-Nim mit 4 Haufen zu 1, 2, 3 bzw. 4 Steinen. Vgl. Kapitel 2.5.1. [10] Ahnung [11] [Fra07], S. 3. Die Quelle dieses Zitats ist ein so genanntes Dynamic Survey, das laufend aktualisiert wird. Das Zitat kann daher in der aktuellen Version vom hier angegebenen abweichen. [12] Wir führen diese Theorie in Kapitel 2.4 ein. [13] [Pla06], S.2f

2.3 Spielegraph

Zur Analyse eines Spiels empfiehlt sich die Darstellung mit Hilfe eines Graphen.

Definition 2.16 (Spielegraph) Sei G ein Spiel mit n Spielpositionen $x_1, x_2, \ldots x_n$. Dann bezeichnet $SG(G)$ den Spielegraphen von G. $SG(G)$ ist ein gerichteter Graph, der alle Spielpositionen von G als Ecken enthält und eine Kante von x_i nach x_j genau dann, wenn es in G einen Zug von der Spielposition x_i zur Position x_j gibt.

Definition 2.17 (Startecke, Zielecken) Als Startecke bezeichnen wir jene Ecke, welche die Startposition des Spiels darstellt. Als Zielecken bezeichnen wir all jene Positionen, die eine Zielposition repräsentieren, das heißt, eine Position, von der aus kein Zug mehr möglich ist.

Lemma 2.18 Sei G ein Spiel. Dann gibt es in $SG(G)$ keine Kanten, die zur Startecke führen. Ebenso existieren keine Kanten, die von einer Zielecke ausgehen.

Beweis: Folgt sofort aus den Definitionen der Begriffe Startecke, Zielecke bzw. Startposition und Zielposition.

Das Spiel beginnt bei der Startecke. Der erste Spieler zieht entlang einer Kante zu einer weiteren Ecke, danach zieht der zweite Spieler von dieser Ecke weiter zu einer dritten und so fort. Die beiden Spieler ziehen so lange abwechselnd, bis einer von ihnen eine Zielecke erreicht. Dieser Spieler ist Sieger im normalen Spiel bzw. Verlierer im Misère-Spiel.[14]

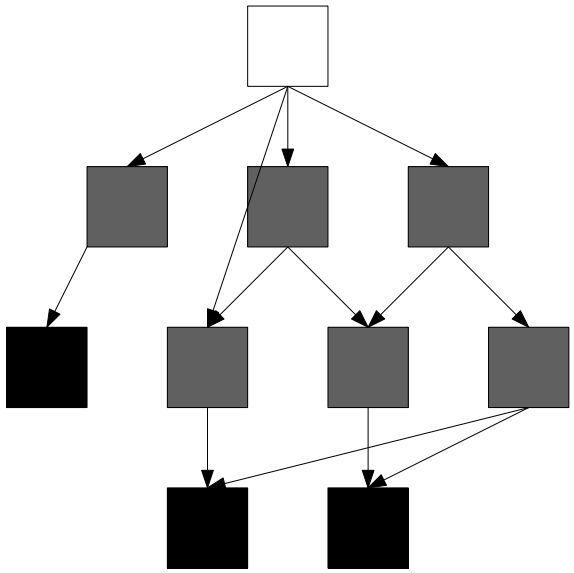


Abbildung 1: Beispiel für einen Spielegraph (Startecke weiß, Zielecken schwarz)

[14] Der Einfachheit halber vernachlässigen wir im Folgenden die Möglichkeit, dass ein Spieler vorzeitig aufgibt.

Bemerkung 2.19 [Spielegraph für Misère-Spiele]

Für die Analyse eines Spiels möchten wir erreichen, dass jener Spieler das Spiel gewinnt, der auf eine Position zieht, die einer Endecke des Spielegraphen entspricht. Für Misère-Spiele (bei denen der Spieler verliert, der eine Zielecke des normalen Spiels erreicht) müssen wir dazu den Spielegraphen anpassen.

Eine Möglichkeit besteht darin, eine zusätzliche uneigentliche Spielposition A sowie Kanten von jeder Zielecke des normalen Spiels nach A hinzuzufügen, was dazu führt, dass A zur einzigen Zielecke des Graphen wird.

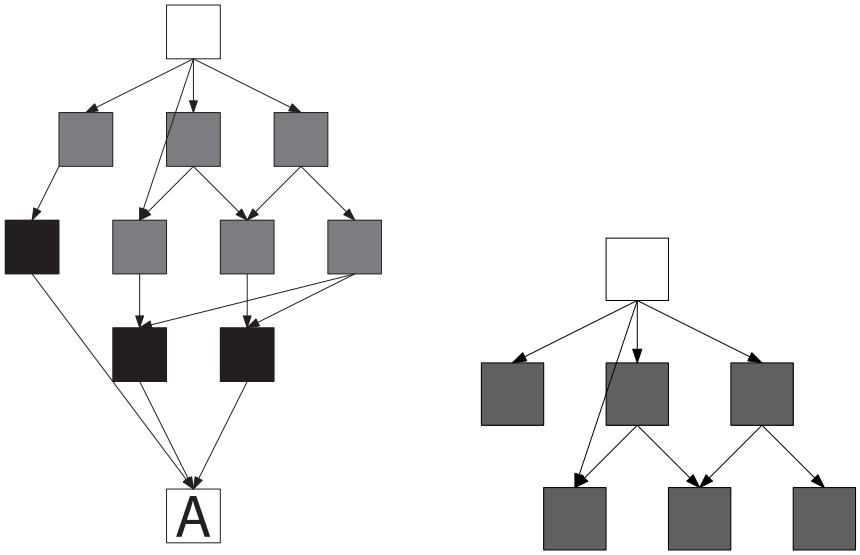


Abbildung 2: Misère-Graph

Ebenso ist es möglich, alle Zielecken des normalen Spiels aus dem Spielegraphen zu entfernen. Dadurch werden alle Positionen, von denen im normalen Spiel noch ein Zug möglich war, zu Zielecken. Dies führt ebenfalls dazu, dass jener Spieler gewinnt, der eine Zielecke erreicht.

Es gilt allgemein:

Satz 2.20 Sei $SG(G)$ der Spielegraph eines (stark endlichen) Spiels G. Dann folgt:

- $SG(G)$ hat endlich viele Ecken und Kanten.
- $SG(G)$ enthält keine Kreise.
- Von jeder Ecke $x \in SG(G)$ gibt es einen Weg zu einer Zielecke.

Beweis: Alle drei Aussagen folgen sofort aus der Tatsache, dass G stark endlich vorausgesetzt wurde.

Definition 2.21 (Gleichheit von Spielen) Seien G und H zwei Spiele. G und H heißen gleich, wenn sie gleiche (d. h. isomorphe) Spielegraphen haben. Wir schreiben im Folgenden: $G = H$.

Definition 2.22 (Menge der Spielpositionen) Sei G ein Spiel. Mit $P(G)$ bezeichnen wir die Menge aller Spielpositionen von G.

Definition 2.23 (Nachfolger einer Position) Seien x und y Ecken eines Spielegraphen. y heißt (direkter) Nachfolger von x, wenn es im Spielegraphen eine Ecke von x nach y gibt.

Definition 2.24 (Menge der Nachfolger) Sei x eine Ecke eines Spielegraphen. Dann bezeichnen wir mit $N(x)$ die Menge aller direkten Nachfolger von x.

Bemerkung 2.25 Die Menge $N(x)$ enthält alle Spielpositionen, die von x in einem Zug erreichbar sind.

Wir definieren nun eine Partition von $P(G)$. Dazu weisen wir jeder Position einen so genannten Grundywert zu.

2.4 Grundyfunktion und Grundywerte

Definition 2.26 (Die Funktion *mex*) Sei $M \subset \mathbb{N}_0$. Dann bezeichnet $mex(M)$[15] die kleinste Zahl $\in \mathbb{N}_0$, die nicht in M enthalten ist: $mex(M) = min\{n, n \in \mathbb{N}_0 \backslash M\}$.

Bemerkung 2.27 Man sieht leicht, dass $mex(M)$ für alle echten Teilmengen von $\mathbb{N}_0$ wohldefiniert ist. Für $M = \mathbb{N}_0$ existiert $mex(M)$ nicht.

Definition 2.28 (Grundywert und Grundyfunktion) Unter dem Grundywert[16] $g(x)$ einer Position x verstehen wir die kleinste Zahl $\in \mathbb{N}_0$, die bei keinem direkten Nachfolger von x als Grundywert vorkommt:

$$g(x) := mex(\{g(y_1), \ldots, g(y_n)\}, N(x) = \{y_1, \ldots, y_n\}) =$$

$$= min\{k \in \mathbb{N}_0 \backslash \{g(y_1), \ldots, g(y_n)\}, N(x) = \{y_1, \ldots, y_n\}\}.$$

Die Funktion $g : P(G) \to \mathbb{N}_0$ heißt Grundyfunktion des Spielegraphen.[17]

Mit Hilfe der Grundyfunktion lässt sich eine Partition der Spielpositionen definieren.

Definition 2.29 $\mathcal{G}_n := \{x \in P(G) : g(x) = n\}$.

[15] *mex* steht für minimal excluded value (kleinster ausgeschlossener Wert). [16] In der Literatur wird dieser Wert auch als Sprague-Grundy-Wert oder einfach G-Wert bezeichnet. [17] Diese Definition geht auf die Arbeiten von R. P. Sprague [Spr36] sowie P. M. Grundy [Gru39] zurück, die unabhängig voneinander die Grundlagen für eine Theorie zur Analyse kombinatorischer Spiele bereitet haben.

Lemma 2.30 Die Mengen $\mathcal{G}_n$ bilden eine Partition von $P(G)$, d. h.:

- $\mathcal{G}_m \cap \mathcal{G}_n = \varnothing \quad \forall m \neq n$
- $\cup_{n \geq 0} \mathcal{G}_n = P(G)$

Beweis: Die Funktion $g(x)$ weist jeder Spielposition $x \in P(G)$ einen eindeutigen Grundywert zu. Daraus folgt:

$$\forall x \in P(G) \exists! n : x \in \mathcal{G}_n.$$

□

Definition 2.31 (Grundywert eines Spiels) Sei G ein Spiel. Als Grundywert von G bezeichnen wir den Grundywert der Startposition von G.

Bemerkung 2.32 Jede beliebige Position x eines Spiels G kann als eigenständiges Spiel aufgefasst werden, wenn man alle Ecken, die von x aus nicht mehr erreicht werden können, sowie die dazugehörigen Kanten aus dem Spielegraphen $SG(G)$ entfernt und x als Startecke festlegt.

Lemma 2.33 Der Grundywert einer Zielecke ist 0.
Beweis: Sei x eine Zielecke. Dann ist $N(x) = \varnothing$ und $g(x) = mex(\varnothing) = min\{n, n \in \mathbb{N}_0\} = 0$. □

Definition 2.34 (Kern eines Spiels) Der Kern eines Spiels G ist die Menge aller Ecken mit Grundywert 0 : $Ker(G) = \mathcal{G}_0 = \{x \in P(G), g(x) = 0\}$

Satz 2.35 (Existenz und Eindeutigkeit der Grundyfunktion) Der Graph jedes (stark endlichen) Spiels hat eine eindeutige Grundyfunktion (*ohne Beweis*).

Satz 2.36 (Existenz und Eindeutigkeit des Kerns) Jedes (stark endliche) Spiel hat einen Kern. Dieser ist eindeutig.
Beweis: Die Aussage folgt sofort aus Satz 2.35. □

Es gelten nun zwei für die Spielanalyse wesentliche Folgerungen:

Korollar 2.37 Sei G ein Spiel mit Kern $Ker(G)$. Dann gilt:

- Jede Zielecke liegt im Kern.
- Jeder Zug von einer Ecke im Kern führt zu einer Ecke, die nicht im Kern liegt. $\forall x \in Ker(G) : \quad \forall y \in N(x) : y \notin Ker(G)$
- Von jeder Ecke, die nicht im Kern liegt, gibt es einen Zug zu einer Ecke im Kern. $\forall x \notin Ker(G) : \quad \exists y \in N(x) : y \in Ker(G)$

Beweis:

- Sei x eine Zielecke. Daraus folgt $N(x) = \varnothing$ und aus der Definition der Grundyfunktion weiters $g(x) = 0$ und somit $x \in Ker(G)$.
- Sei x eine Ecke im Kern. Daraus folgt $g(x) = 0$. Weil der Grundywert von x nach Definition der kleinste Wert aus $\mathbb{N}_0$ ist, der bei keinem direkten Nachfolger von x als Grundywert vorkommt, folgt, dass alle Nachfolger von x Grundywert > 0 haben und daher nicht im Kern liegen.
- Sei x eine Ecke nicht im Kern. Daraus folgt $g(x) > 0$. Aus der Definiton der Grundyfunktion folgt nun, dass es zumindest einen Nachfolger von x mit Grundywert 0, also $x \in Ker(G)$, gibt. □

Korollar 2.38 Ein Spieler, der eine Position im Kern erreicht hat, kann bei fehlerfreiem Spiel stets gewinnen. Im nächsten Zug muss sein Gegner auf eine Position ziehen, die nicht im Kern liegt, wodurch es für ihn erneut die Möglichkeit gibt, in den Kern zu spielen. Nach diesem Prinzip kann ein Spieler, der einmal eine Position mit Grundywert 0 erreicht hat, stets eine Zielecke erreichen und somit gewinnen.

Korollar 2.39 Ein Spiel G hat Grundywert 0 genau dann, wenn der 2. Spieler bei optimalem Spiel stets gewinnen kann.

Beweis: ($\Rightarrow$): G habe Grundywert 0. Das bedeutet, dass die Startposition Grundywert 0 hat. Nach Korollar 2.37 muss der 1. Spieler also in seinem ersten Zug auf eine Position mit Grundywert > 0 ziehen. Nun hat der 2. Spieler die Möglichkeit, auf eine Position mit Grundywert 0 zu ziehen. Dies setzt sich so lange fort, bis der 2. Spieler schließlich eine Zielposition erreicht.

($\Leftarrow$): Besitze umgekehrt der 2. Spieler eine Gewinnstrategie. Angenommen der Grundywert des Spiels ist > 0. Dann hätte der 1. Spieler die Möglichkeit, auf eine Position mit Grundywert 0 zu ziehen und folglich zu gewinnen. Widerspruch! Daher ist der Grundywert des Spiels gleich 0. □

Korollar 2.40 Ein Spiel G hat Grundywert > 0 genau dann, wenn der 1. Spieler bei optimalem Spiel stets gewinnen kann.

Beweis: Analog zum vorigen Korollar.

Bemerkung 2.41 Beginnend von den Endecken, kann man (theoretisch) allen Spielpositionen eines Spiels Grundywerte zuweisen und so den Grundywert dieses Spiels bestimmen. Für Misère-Spiele kann man entweder den Spielegraphen (wie in Bemerkung 2.19 beschrieben) anpassen, oder einfach allen Zielecken den Grundywert 1 (statt 0) zuweisen.

Für Spiele mit geringer Anzahl von Spielpositionen ist eine komplette Analyse auch praktisch möglich. Für größere Spiele beschränkt man sich darauf, allen Ecken, die höchstens n Züge von einer Zielecke entfernt sind, Grundywerte zuzuweisen (Endspielanalyse[18]).

[18] Als Beispiel sei hier das Schachspiel genannt (auch wenn es kein neutrales Spiel ist und daher keine Grundyfunktion in unserem Sinne besitzt). Dort analysiert man oft Endspiele für alle Positionen mit $\leq n$ Spielfiguren. Für $n = 6$ sind diese Daten frei im Internet zugänglich [Kry07].

Eine gröbere Klassifizierung der Spielpositionen (als die Zuweisung von Grundywerten), die bereits auf (den Mathematiker und Schachweltmeister Emanuel) Lasker zurückgeht, wäre die Unterteilung in Gewinn- und Verlustecken. Gewinnecken sind all jene Spielpositionen, von denen jener Spieler, der dorthin zieht, stets gewinnen kann (Grundywert 0). Verlustecken sind jene, von denen er bei perfektem Spiel des Gegners stets verlieren wird (Grundywert > 0).[19] Formal definiert bedeutet das:

Definition 2.42 Sei G ein Spiel. Eine Spielposition x heißt Gewinnecke, wenn gilt: $x \in \mathcal{G}_0$. Andernfalls ($x \in \bigcup_{n\geq 1} \mathcal{G}_n = P(G)\backslash\mathcal{G}_0$) heißt x Verlustecke.

Definition 2.43 (Gewinnstrategie) Ein Spieler besitzt eine Gewinnstrategie, wenn er bei optimalem Spiel stets gewinnen kann (egal wie sein Gegner zieht).

Definition 2.44 (Summe von Spielen) Werden n Spiele $G_1, \ldots, G_n$ gleichzeitig gespielt (das heißt in jedem Zug wird in einem der Spiele gezogen), sprechen wir im Folgenden von der Summe dieser Spiele und schreiben dafür $G_1 + G_2 + \ldots + G_n = \sum_{k=1}^{n} G_k$. Eine Spielposition eines Summenspiels ist ein Vektor $x = (x_1, \ldots x_n)$, wobei die $x_i, 1 \leq i \leq n$ Spielpositionen der einzelnen Spiele sind.

Lemma 2.45 Es gilt:

- Die Summe von stark endlichen Spielen ist stark endlich.
- Die Summe von stark endlichen Spielen besitzt eine Grundyfunktion und einen Kern.

Die erste Aussage folgt sofort aus der Definition. Damit folgt die zweite Behauptung aus Satz 2.35 und Satz 2.36. □

Wir können die Grundyfunktion eines Summenspiels sogar explizit angeben. Dies stellt eines der grundlegendsten und wichtigsten Ergebnisse der kombinatorischen Spieltheorie dar. Es gilt allerdings nur für die Normalform eines Spiels und nicht für Misère-Spiele.

Definition 2.46 (Binäre Summe in $\mathbb{N}_0$) Seien $a, b \in \mathbb{N}_0$ und seien $a_0, \ldots, a_k$ beziehungsweise $b_0, \ldots, b_m$ die Ziffern der Binärdarstellung von a und b:

$$a = \sum_{i=0}^{k} a_i \cdot 2^i \quad a_i \in \{0, 1\}$$
$$b = \sum_{i=0}^{m} b_i \cdot 2^i \quad b_i \in \{0, 1\}.$$

Sei o.B.d.A. $k \leq m$ und $a_i := 0 \quad \forall i > k$. Sei schließlich $c_i \equiv a_i + b_i \pmod 2, \quad \forall 0 \leq i \leq m$ und $c = \sum_{i=1}^{m} c_i \cdot 2^i$. Dann nennen wir c die binäre Summe[20] von a und b und schreiben dafür $a \dot{+} b$.

[19] Vor allem in der englischsprachigen Literatur sind auch die Begriffe P-position und N-position gebräuchlich. P bedeutet *previous player wins* und bezeichnet eine Gewinnecke. N steht für *next player wins* und bezeichnet eine Verlustecke. [20] Ebenfalls gebräuchlich sind die Bezeichnungen XOR-Summe oder Nim-Summe.

Lemma 2.47 Es gilt: $a \dot{+} a = 0 \quad \forall a \in \mathbb{N}_0$.
Beweis: Es gilt $1 + 1 \pmod 2 \equiv 0$ und daher $c_i \equiv a_i + a_i \pmod 2 \equiv 0, \quad \forall 1 \leq i \leq m$. Daher folgt $a \dot{+} a = \sum_{i=1}^{k} c_i \cdot 2^i = 0$. □

Satz 2.48 (Hauptsatz über Summenspiele) Seien $G_1, \ldots, G_n$ Spiele mit Grundyfunktionen $g_1, \ldots, g_n$. Dann gilt: Das Summenspiel $G := G_1 + G_2 + \ldots + G_k = \sum_{k=1}^{n} G_k$ hat die Grundyfunktion $g(x) := g(x_1, \ldots, x_n) = g_1(x_1) \dot{+} \ldots \dot{+} g_n(x_n)$.

Beweis: Es ist zu zeigen, dass g Grundyfunktion von G ist. Dazu zeigen wir:

1. $y \in N(x) \Rightarrow g(x) \neq g(y)$
2. Sei $b \in \mathbb{N}_0$: $0 \leq b < g(x) \Rightarrow \exists y \in N(x) : g(y) = b$

ad 1: Seien $x = (x_1, x_2, \ldots, x_n)$ und $y = (y_1, y_2, \ldots, y_n)$ Ecken in G und sei $y \in N(x)$. Es folgt, dass x und y sich in genau einer Koordinate unterscheiden. Der Index dieser Koordinate sei i. Es gilt daher $x_i \neq y_i$ und $x_k = y_k \quad \forall k \neq i$. Da $\overrightarrow{x_i y_i}$ eine Kante in G_i ist, folgt weiters, dass $g(x_i) \neq g(y_i)$. Aus $x_k = y_k \quad \forall k \neq i$ folgt zudem sofort $g(x_k) = g(y_k) \quad \forall k \neq i$.

Wir schließen nun indirekt: Angenommen $g(x) = g(y)$. Nach Definition ist $g(x) = g_1(x_1) \dot{+} \ldots \dot{+} g_n(x_n)$ und $g(y) = g_1(y_1) \dot{+} \ldots \dot{+} g_n(y_n)$. Sei nun $c := g(x_1) \dot{+} g(x_2) \dot{+} \ldots \dot{+} g(x_{i-1}) \dot{+} g(x_{i+1}) \dot{+} \ldots g(x_n)$. Es gilt $g(x) = g_i(x_i) \dot{+} c$ und wegen $g(x_k) = g(y_k)$ für alle $k \neq i$ auch $g(y) = g_i(y_i) \dot{+} c$.

Wir haben $g(x) = g(y)$ angenommen. Es folgt $g_i(x_i) = g_i(y_i)$. y kann daher kein Nachfolger von x sein. Widerspruch!

ad 2: Sei $0 \leq b < g(x) = p$. Wir setzen $d = b \dot{+} p$ und schreiben d in Binärschreibweise:

$$d = d_k \cdot 2^k + d_{k-1} \cdot 2^{k-1} + \ldots + d_1 \cdot 2 + d_0 = [d_k d_{k-1} \ldots d_1 d_0].$$

Es sei $d_k = 1$ und $d_j = 0 \quad \forall j > k$ (d.h. $2^k \leq d < 2^{k+1}$). Aus $0 \leq b < p$ folgt dann, dass $p_k = 1$ und $b_k = 0$. Wir wissen:

$$p = g(x) = g_1(x_1) \dot{+} \ldots \dot{+} g_n(x_n).$$

p hat also an der k-ten Stelle der Binärdarstellung eine 1. Daraus folgt: $\exists i : g_i(x_i)$ hat an der k-ten Stelle ihrer Binärdarstellung eine 1. Sei o.B.d.A. $i = 1$. Wir betrachten die Binärdarstellungen von $g_1(x_1)$ und d:

$$g_1(x_1) = c_0 + c_1 \cdot 2 + \ldots + \mathbf{1} \cdot 2^k + c_{k+1} \cdot 2^{k+1} + \ldots$$

$$d = d_0 + d_1 \cdot 2 + \ldots + \mathbf{1} \cdot 2^k + \mathbf{0}$$

Wir betrachten nun $d \dot{+} g_1(x_1)$. Diese Zahl hat an der k-ten Stelle ihrer Binärdarstellung eine 0. Alle Stellen mit Index $> k$ sind dieselben wie bei $g_1(x_1)$, da diese Stellen bei d gleich 0 sind. Es folgt daher $d \dot{+} g_1(x_1) < g_1(x_1)$.

Nach der Definition der Grundyfunktion für G_1 gilt:

$$\exists y_1 \in N(x_1) : g_1(y_1) = d \dot{+} g_1(x_1).$$

$\overrightarrow{x_1y_1}$ ist eine Kante in G_1. Weiters gilt $g_j(x_j) = g_j(y_j) \quad \forall 2 \leq j \leq n$. Daraus folgt:

$$\begin{aligned} g(y) &= g_1(y_1) \dot{+} g_2(y_2) \ldots \dot{+} g_n(y_n) \\ &= g_1(y_1) \dot{+} g_2(x_2) \ldots \dot{+} g_n(x_n) \\ &= d \dot{+} g_1(x_1) \dot{+} g_2(x_2) \ldots \dot{+} g_n(x_n) \\ &= d \dot{+} g(x) \\ &= d \dot{+} p \\ &= b \end{aligned}$$

Es folgt: $y \in N(x)$ und $g(y) = b$. □

Bemerkung 2.49 Die Grundywerte eines Summenspiels ergeben sich also in einfacher Weise als binäre Summe der Grundywerte der ursprünglichen Spiele.

Eine wichtige Anwendung des Hauptsatzes ist die Summe zweier gleicher Spiele. Conway schreibt in [Con83]:

> Ich erinnere mich an die berühmte Geschichte von dem kleinen Mädchen, das simultan gegen zwei SCHACHGROSSMEISTER (sicherlich ein GROSSES Konzept!) spielte. Wie gelang es ihr, eines der beiden Spiele zu gewinnen? Anne Luis spielte Schwarz gegen Spassky[21], Weiß gegen Fischer[22]. Spassky zog als erster und Anne Luis wiederholte seinen Zug als ihren ersten Zug im Spiel gegen Fischer, dann wiederholte sie Fischers Gegenzug als ihren eigenen Gegenzug zu Spasskys erstem Zug usw.

Korollar 2.50 Sei G ein Spiel. Dann hat $G + G$ Grundywert 0, d.h. der zweite Spieler kann stets gewinnen.

Beweis: Dies folgt mit dem Hauptsatz aus Lemma (2.48). □

Bemerkung 2.51 Die Gewinnstrategie des 2. Spielers für ein Summenspiel $G + G$ lässt sich leicht angeben: Sei H Summe zweier gleicher Spiele: $H = G_1 + G_2$ und $G_1 = G_2$. Wenn A im Spiel G_1 zieht, macht B anschließend den gleichen Zug in G_2 und umgekehrt. Auf diese Weise entsteht nach jedem Zug von B wieder ein Spiel, das sich als Summe zweier gleicher Spiele schreiben lässt. Dies lässt sich so lange fortsetzen, bis der 2. Spieler mit dem letzten Zug das Spiel beendet.

Bemerkung 2.52 Satz 2.48 gilt nur für normale Spiele. Die Tatsache, dass er bei Misère-Spielen nicht anwendbar ist, stellt ein Hauptproblem bei der Analyse dieser Spiele dar.[23]

[21] Boris Spassky ist ein russisch-französischer Schachspieler (geboren 1937). [22] Bobby Fischer (1943-2008) war ein US-amerikanischer Schachspieler (seit 2005 isländischer Staatsbürger). Er bezwang Spassky 1972 im „Match des Jahrhunderts“[Wik07] in Reykjavík. [23] Neue Ergebnisse der Theorie der Misère-Spiele liefern eine Verallgemeinerung des Hauptsatzes, der auch für diese Spiele anwendbar ist. Vgl. dazu z. B. [Sie08] und [Pla06].

2.5 Einige Beispiele

Vor einem abschließenden Kapitel zur Theorie sowie dem eigentlichen Teil dieser Arbeit (der Analyse des Spiels Juvavum) betrachten wir im Folgenden einige klassische Beispiele der kombinatorischen Spieltheorie. Der Schwerpunkt liegt dabei auf solchen Spielen, deren Analyse im weiteren Verlauf der Arbeit von Nutzen ist.

2.5.1 Nim

Nim ist ein Spiel, das mit einer beliebigen Anzahl von Spielsteinen (Münzen, Streichhölzern, Steinen usw.) gespielt wird. Diese werden auf $m > 0$ Haufen aufgeteilt, von denen jeder $n_i > 0, \quad 1 \leq i \leq m$ Spielsteine enthält. Jeder Zug besteht darin, beliebig viele Steine (mindestens aber einen) von **einem** beliebigen Haufen weg zu nehmen. Wer in seinem Zug den letzten Stein entfernt, gewinnt. Analog kann auch Misère-Nim betrachtet werden. Dabei gewinnt jener Spieler, der zuerst nicht mehr ziehen kann (es verliert also, wer den letzten Stein aufnimmt).

Wir betrachten zunächst den Fall $m = 1$. Offensichtlich kann der 1. Spieler stets gewinnen (indem er einfach im ersten Zug alle Steine wegnimmt). Sei nun $m > 1$. Da bei jedem Zug stets nur in einem Haufen gespielt wird, kann Nim als Summenspiel der einzelnen Haufen betrachtet werden. Mit Hilfe des Hauptsatzes können die Grundywerte für beliebige Positionen daher sehr einfach berechnet werden.

Definition 2.53 (Nimbers) Wir bezeichnen einen Nim-Haufen der Größe n im Folgenden mit $*n$. Die Spiele der Menge $\{*1, *2, *3, \ldots\}$ nennen wir Nimbers.

Satz 2.54 Es gilt: $g(*n) = n \quad \forall n \geq 0$.

Beweis: Wir verwenden Induktion nach n. Sei $n = 0$. $*0$ hat Grundywert 0, da es keine Nachfolgepositionen gibt. Angenommen die Behauptung stimmt für alle $0 \leq m < n$. Dann folgt:

$$\begin{aligned} g(*n) &= mex\{g(*0), g(*1), \ldots, g(*(n-1))\} \\ &= mex\{0, 1, \ldots n-1\} \\ &= n. \end{aligned}$$

□

Abbildung 3 zeigt das Spiel $*3 + *4 + *1 + *2$. Es hat Grundywert

$$g(*3, *4, *1, *2) = g(*3) \dot{+} g(*4) \dot{+} g(*1) \dot{+} g(*2) = 3 \dot{+} 4 \dot{+} 1 \dot{+} 2 = 4.$$

Das bedeutet, dass der 1. Spieler stets gewinnen kann. Ein guter Zug ist zur Position $(3, 0, 1, 2)$ (Wegnehmen des kompletten zweiten Haufens). Dies führt auf die Position $*3 + *0 + *1 + *2$, deren Grundywert 0 ist:

$$g(*3, *0, *1, *2) = g(*3) \dot{+} g(*0) \dot{+} g(*1) \dot{+} g(*2) = 3 \dot{+} 0 \dot{+} 1 \dot{+} 2 = 0.$$

Eine vollständige Analyse des Nim-Spiels wurde erstmals von Bouton veröffentlicht [Bou02].

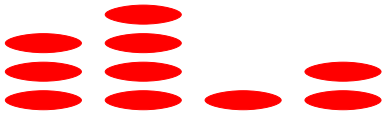


Abbildung 3: $*3 + *4 + * + *2$

2.5.2 Das Datumsspiel

Betrachten wir das Spiel aus Beispiel 2.1. Jede Spielposition lässt sich durch ein Paar natürlicher Zahlen (d, m) repräsentieren, wobei $1 \leq d \leq 31$ den Tag und $1 \leq m \leq 12$ den Monat darstellt. Lockert man die Regeln des Spiels dahingehend, dass jeder Monat 31 Tage hat, lässt sich jede Position (d, m) als Summe zweier Nimbers darstellen $(d, m) = *(12 - m) + *(31 - d)$. Die Tatsache, dass manche Monate kürzer sind als andere, ändert an der Struktur des Spiels (insbesondere der Anordnung der Gewinnpositionen) nichts Wesentliches. Zwar lässt es sich nicht mehr als Summe zweier Nim-Haufen schreiben, die Grundywerte jeder Spielposition lassen sich dennoch leicht angeben (siehe Tabelle 1).

	1.	*2.*	*3.*	*4*	*5.*	*6.*	*7.*	*8.*	*9.*	*10.*	*11.*	*12.*	*13.*	*14.*	*15.*	*16.*	*17.*	*18.*	*19.*	*20.*	*21.*	*22.*	*23.*	*24.*	*25.*	*26.*	*27.*	*28.*	*29.*	*30.*	*31.*
Jänner	35	34	33	21	30	19	20	18	17	26	25	24	23	22	7	8	4	3	1	0	2	5	16	15	14	13	11	12	9	10	6
Februar	23	24	32	31	20	29	17	16	15	18	13	14	22	21	19	7	6	5	3	2	0	1	4	12	11	8	10	9			
März	24	23	22	32	31	30	29	28	27	16	15	12	14	13	21	20	19	17	18	6	1	0	2	3	4	7	8	11	10	9	5
April	34	33	23	24	22	21	28	26	25	15	14	13	12	11	20	19	16	18	17	5	4	2	0	1	3	10	9	6	7	8	
Mai	33	32	31	22	21	20	19	27	26	25	24	23	13	12	11	17	18	10	16	15	14	3	1	0	2	9	6	5	8	7	4
Juni	26	25	24	23	29	22	27	19	18	17	16	22	21	20	12	11	10	9	15	14	13	4	2	2	0	1	7	8	5	6	
Juli	32	31	25	30	23	27	26	20	19	24	22	21	15	14	13	18	17	16	9	8	12	11	10	4	1	0	2	7	6	5	3
August	27	26	30	29	28	22	21	25	24	23	17	16	20	19	18	12	11	15	14	13	8	7	5	10	9	6	0	1	3	4	2
September	28	30	26	25	27	23	22	24	20	19	21	17	16	18	14	13	15	11	10	12	7	6	9	5	8	2	1	0	4	3	
Oktober	31	27	29	28	24	26	25	21	23	22	18	20	19	15	17	16	12	14	13	9	11	10	6	8	7	3	5	4	0	2	1
November	29	28	27	26	25	24	23	*22*	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	11
Dezember	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Tabelle 1: Die Grundywerte des Datumsspiels

Man erkennt, dass die Gewinnpositionen jene Tage sind, für die gilt: $d - m = 19$. Ein Spieler, der eine solche Position erreicht, wird immer gewinnen können. Im Allgemeinen ist bei diesem Spiel der 1. Spieler im Vorteil. Die einzige Startposition, von der aus der 2. Spieler gewinnen kann, ist der 20. Jänner.[24]

2.5.3 Münzen legen

Im Beispiel 2.8 haben wir das Spiel Münzen legen vorgestellt. Wir zeigen nun mit Hilfe eines Symmetriearguments, dass der 1. Spieler auf einem rechteckigen Spielfeld stets gewinnen kann.

Satz 2.55 Bei Münzen legen auf einem rechteckigen Spielfeld kann stets der 1. Spieler gewinnen.

[24] Lässt man die Einschränkung weg, dass die Startposition ein Datum im Jänner sein muss, gibt es natürlich noch weitere (z.B. den 21. Februar).

Beweis: A setzt seine erste Münze in die Mitte des Spielfelds (der Mittelpunkt der Münze liegt genau über dem Mittelpunkt des Spielfelds). Er bedeckt damit das Symmetriezentrum und kann nun jeden Zug seines Gegners daran spiegeln. Diese Vorgehensweise funktioniert für alle Spielfelder, die ein Symmetriezentrum besitzen, das durch eine Münze bedeckt werden kann, insbesondere also für alle rechteckigen Felder. □

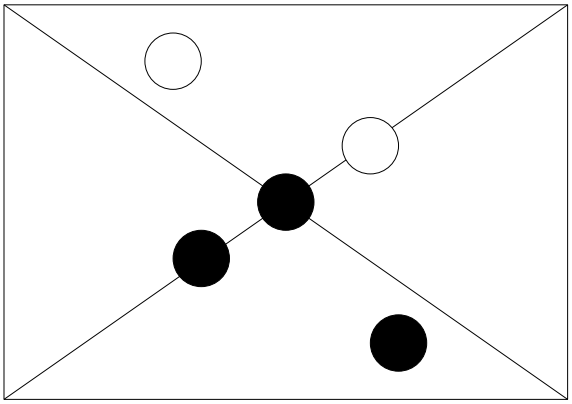


Abbildung 4: Münzen legen: gute Spielstrategie von Schwarz (1. Spieler)

2.5.4 Kayles

Beim Spiel Kayles werden n Kegel in einer Reihe aufgestellt. Jeder Kegel bekommt eine Nummer von 1 bis n. Wir definieren dann den Abstand zweier Kegel mit $dist(i,j) := |i-j|$ und nennen zwei Kegel i und j direkt benachbart, wenn $dist(i,j) = 1$.

Ein legaler Zug bei Kayles besteht nun darin, einen beliebigen Kegel oder zwei beliebige direkt benachbarte Kegel umzuwerfen. Anders ausgedrückt dürfen in jedem Zug zwei Kegel mit Abstand = 1 oder aber ein einziger Kegel entfernt werden.[25] Wer den oder die letzten beiden (direkt benachbarten) Kegel entfernt, gewinnt.

Bei der Variante Dawson's Kayles dürfen in jedem Zug nur zwei direkt benachbarte Kegel entfernt werden. Es ist in dieser Variante nicht erlaubt, nur einen einzelnen Kegel umzuwerfen.

Definition 2.56 (Kayles) Wir bezeichnen Kayles mit n Kegeln im Folgenden mit $K(n)$. Analog bezeichnen wir Dawson's Kayles mit $DK(n)$.

Satz 2.57 Kayles kann stets der 1. Spieler gewinnen.

Beweis: Sei $K(n)$ Kayles mit n Kegeln. Im Fall n gerade entfernt der 1. Spieler die beiden mittleren Kegel. Dies führt auf das Summenspiel $K(\frac{n-2}{2}) + K(\frac{n-2}{2})$. Im Fall n ungerade entfernt er den mittleren Kegel. Dies führt auf das Spiel $K(\frac{n-1}{2}) + K(\frac{n-1}{2})$. In beiden Fällen bleibt nach Folgerung 2.50 ein Spiel mit Grundywert 0 übrig. □

[25] Der Einfachheit halber nehmen wir an, dass die beiden Spieler so gut sind, dass ihnen das stets gelingt.

Als Variation kann man die Kegel auch kreisförmig anordnen.

Satz 2.58 Kreisförmiges Kayles kann stets der 2. Spieler gewinnen.

Beweis: Nach jedem beliebigen Eröffnungszug bleibt ein gewöhnliches Kayles-Spiel übrig, das nach dem vorigen Satz stets von jenem Spieler gewonnen werden kann, der beginnt. In unserem Fall ist dies der 2. Spieler. □

Abbildung 5: Die Kayles- bzw. Dawson's Kayles-Position $(1, 3, 3)$

Man kann auch die Summe mehrere Kayles-Spiele betrachten und für das Summenspiel Grundywerte mit Hilfe des Hauptsatzes berechnen. Analog zu Nim beschreiben wir Positionen von Kayles und Dawson's Kayles durch eine Liste natürlicher Zahlen. Anstatt Haufen von Spielsteinen betrachten wir hier Reihen von Kegeln. Abbildung 5 zeigt die Kayles-Position $(1, 3, 3)$. Die erste Reihe enthält einen Kegel, die beiden anderen enthalten je drei. [26]

Ein Zug ist stets nur in einer der Reihen erlaubt. Formal können wir das dadurch erreichen, dass wir für je zwei Kegel i und j, die nicht in der gleichen Reihe stehen, $dist(i, j) := \infty$ setzen.

2.5.5 Domineering

Domineering oder Cross-Gate[27] ist für uns in erster Linie auf Grund der Ähnlichkeit zu Domino Juvavum, das in Kapitel 3 eingeführt wird, interessant. Zwei Spieler setzen dabei abwechselnd Dominosteine auf ein $m \times n$-Schachbrett derart, dass jedes Domino genau zwei horizontal oder vertikal benachbarte Felder überdeckt. A legt seine Dominos horizontal, B vertikal. Jedes Feld darf nur von einem Domino belegt werden. Wer den letzten Zug macht, gewinnt.

Die neutrale Variante von Domineering, bei der beide Spieler sowohl horizontal als auch vertikal ziehen dürfen, heißt Cram. Wir werden auf dieses Spiel im weiteren Verlauf der Arbeit noch zurück kommen.

Einige Resultate für Domineering finden sich unter anderem bei [Bul02]. Nathan Bullock hat mit Obsequi auch ein Programm zum Lösen von Domineering-Spielen geschrieben, das auf seiner Homepage heruntergeladen werden kann.[28]

[26] O.B.d.A. können wir die Reihenfolge der Zahlen vernachlässigen. Z.B. ist die Position $(2, 4, 1)$ für uns die gleiche wie die Position $(1, 4, 2)$. Der Einfachheit halber ordnen wir die Elemente der Liste der Größe nach. [27] Martin Gardner prägte auch die Bezeichnung Crosscram.
[28] http://www.nathanbullock.org/nathan/obsequi

2.5.6 Chomp

Für den weiteren Verlauf der Arbeit weniger wichtig ist das Spiel Chomp. Da es aber ein sehr schönes Beispiel für die Methode des *strategy stealing* darstellt, soll es hier trotzdem kurz Erwähnung finden.

Chomp wurde 1952 von Fred Schuh als *Spiel der Teiler* veröffentlicht und in anderer Form von David Gale 1974 wiederentdeckt. Der Name Chomp stammt von Martin Gardner.

Das Spielfeld ist ein rechteckiges $m \times n$-Schachbrett. Ein legaler Zug besteht darin, ein beliebiges Feld zu wählen und es gemeinsam mit allen rechts und/oder darüber liegenden Feldern zu entfernen (es werden also immer rechteckige Teilfelder entfernt). Wer das letzte Stück (die linke untere Ecke) nehmen muss, verliert. Zur Motivation stelle man sich das Spielfeld als eine Tafel Schokolade vor, deren linkes unteres Stück vergiftet ist. Abbildung 6 zeigt die Startposition eines 3×5-Spiels sowie drei legale erste Züge.

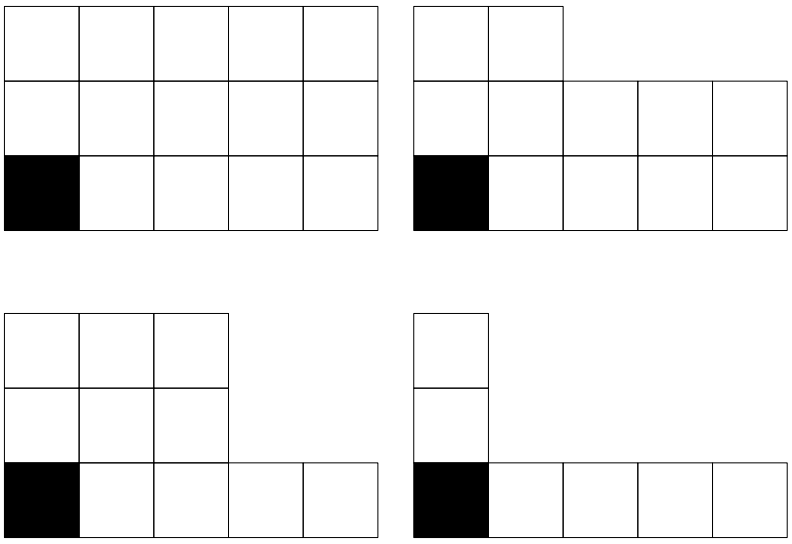


Abbildung 6: Chomp 3×5

Mit Hilfe eines so genannten *strategy stealing*-Arguments kann man zeigen, dass der 1. Spieler bei Chomp eine Gewinnstrategie besitzt.

Satz 2.59 Der 1. Spieler kann bei Chomp immer gewinnen.

Beweis (indirekt): Da kein Unentschieden möglich ist, muss einer der beiden Spieler eine Gewinnstrategie besitzen. Angenommen der 2. Spieler hat eine Gewinnstrategie, d. h. er kann bei jedem beliebigen Eröffnungszug des 1. Spielers gewinnen, insbesondere, wenn dieser in seinem Eröffnungszug nur das rechte obere Eckfeld wegnimmt. Nach unserer Voraussetzung hat der 2. Spieler nun eine Zugmöglichkeit, die sicher zum Sieg führt. Diesen Zug hätte der 1. Spieler aber gleich zu Beginn wählen können und damit gewonnen. Widerspruch! Es folgt, dass der 2. Spieler keine Gewinnstrategie haben kann. □

Obwohl wir gezeigt haben, dass der 1. Spieler stets gewinnen kann, ist bis heute keine allgemeine Gewinnstrategie bekannt.

Weitere Informationen (inklusive einiger Literaturangaben) zu Chomp finden sich unter anderem auf der Internetseite [Bro].

2.6 Lösen von Spielen

Unser Ziel ist es natürlich, ein Spiel "zu lösen". Dazu müssen wir zunächst festlegen, was wir unter einer Lösung eines Spiels verstehen. Wir definieren dazu drei Lösungskategorien.

Definition 2.60 (Lösungskategorien) Sei G ein Spiel. G heißt ultra-schwach gelöst, wenn bekannt ist, welcher Spieler eine Gewinnstrategie besitzt. G heißt schwach gelöst, wenn eine Spielstrategie bekannt ist, mit deren Hilfe von der Startposition des Spiels perfekt gespielt werden kann. G heißt stark gelöst, wenn von jeder Spielposition perfekt gespielt werden kann.

Bemerkung 2.61 Wir beschränken uns in dieser Arbeit mit wenigen Ausnahmen auf stark endliche, neutrale Zweipersonenspiele. Unter perfekt spielen verstehen wir in diesem Fall, dass von jeder Position mit Grundywert > 0 stets auf eine Position mit Grundywert 0 gezogen wird.

Beispiel 2.62 In dieser Einführung haben wir Chomp (durch ein *strategy stealing*-Argument) ultra-schwach, Münzen legen[29] (durch ein Symmetrieargument) schwach sowie Nim (durch eine Berechnungsvorschrift für Grundywerte) stark gelöst.

2.7 Eine Bemerkung zur Schreibweise

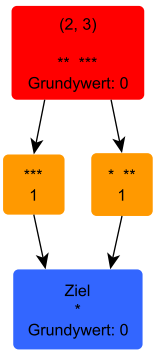


Abbildung 7: Spielegraph von DK(2,3)

[29] Es soll uns an dieser Stelle nicht weiter stören, dass Münzen legen kein stark endliches Spiel ist und wir streng genommen erst definieren müssten, was wir in diesem Fall unter *perfekt* spielen verstehen.

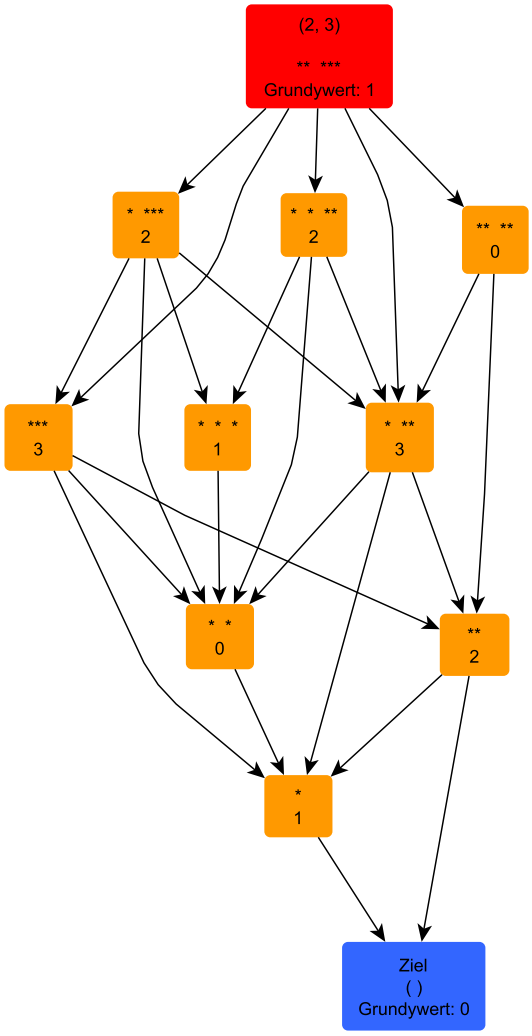


Abbildung 8: Spielegraph von K(2,3)

Abschließend machen wir eine einfache, aber wichtige Bemerkung über Spielpositionen, die in verschiedenen Spielen auftreten können und deren Schreibweise. Als Beispiel betrachten wir dazu Kayles und Dawson's Kayles.

Beispiel 2.63 Wir betrachten die Spielposition $(2,3)$ von Kayles. Sie hat Grundywert 1 (vgl. Abbildung 8). Betrachten wir $(2,3)$ allerdings als Spielposition von Dawson's Kayles, hat sie Grundywert 0 (vgl. Abbildung 7).

Wir müssen in diesem Fall also zwischen dem Grundywert bzgl. Kayles und dem Grundywert bzgl. Dawson's Kayles unterscheiden. Ersteren bezeichnen wir mit $g_K(3,2)$ oder $g(K(3,2))$, letzteren mit $g_{DK}(3,2)$ oder $g(DK(3,2))$.

Bemerkung 2.64 Im Laufe dieser Arbeit werden wir noch einige andere Spiele kennen lernen, die auf den gleichen Spielfeldern (z. B. $m \times n$-Schachbrettern) und mit den gleichen

Spielsteinen (z. B. Dominos) gespielt werden. In diesem Fall verwenden wir Indizes, um klar zu machen, auf welches Spiel sich der Grundywert einer Spielposition bezieht.

Seien zum Beispiel S_1 und S_2 zwei Spiele und sei x eine Spielposition, die in beiden dieser Spiele auftreten kann. Dann unterscheiden wir zwischen $g_{S_1}(x)$ (der Grundyfunktion bzgl. S_1) und $g_{S_2}(x)$ (der Grundyfunktion bzgl. S_2). Wenn aus dem Zusammenhang klar hervorgeht, welches Spiel gemeint ist, lassen wir die Indizes weg.

3 Das Spiel Juvavum und seine Varianten

3.1 Spielbeschreibung

3.1.1 Regeln, Spielbrett, Spielsteine

Sowohl Juvavum[30] als auch Domino Juvavum sind stark endliche Zweipersonenspiele, die auf einem $m \times n$-Spielbrett gespielt werden ($m, n \in \mathbb{N}$). Beide Spieler verfügen über die gleichen Spielsteine. Dies sind bei Juvavum Münzen, bei Domino Juvavum (wie der Name bereits vermuten lässt) Dominosteine.

Ein Juvavum-Spielzug besteht darin, **beliebig viele Steine (Münzen)** derart **in eine beliebige Zeile *oder* Spalte** zu setzen, dass eine Münze stets genau ein 1×1-Feld bedeckt.

Analog besteht ein Domino Juvavum-Zug darin, **beliebig viele Dominos in eine beliebige Zeile *oder* Spalte** zu setzen. Jedes Domino bedeckt dabei zwei horizontal oder vertikal benachbarte Felder. Die Dominos dürfen sich weder überlappen, noch darf ein Domino über den Spielfeldrand hinausragen. In jedem Zug dürfen nur Felder einer Zeile **oder** einer Spalte belegt werden (nicht aber beides), das bedeutet, dass alle Dominos in einem Zug entweder horizontal oder vertikal gesetzt werden.

Definition 3.1 (Zeilenzug, Spaltenzug) Werden die Spielsteine in einem Zug horizontal gelegt, sprechen wir von einem Zeilenzug. Einen Zug, bei dem die Steine vertikal gesetzt werden, nennen wir Spaltenzug.

Bemerkung 3.2 Ein Zug bei (einfachem) Juvavum, bei dem nur ein einziger Stein gesetzt wird, ist sowohl ein Zeilen- als auch ein Spaltenzug.

Das Spiel Cram wurde bereits in Kapitel 2.5.5 beschrieben. Es ist eine Variante von Domino Juvavum, bei der in jedem Zug nur ein Domino gesetzt werden darf. Auf Grund dieser Ähnlichkeit werden wir es im weiteren Verlauf dieser Arbeit ebenfalls analysieren.

Beispiel 3.3 Abbildung 9 zeigt einige mögliche Züge von der Startposition (leeres Spielfeld) eines 4×5-Juvavum-Spiels. Beim ersten und dritten Zug handelt es sich um Zeilenzüge. Die Beispiele 2, 3 und 4 sind Spaltenzüge.

Abbildung 10 zeigt einige regelkonforme Züge von der Startposition eines Domino Juvavum-Spiels auf einem 4×5-Spielbrett.[31] Beispiel 1 und 4 zeigen jeweils einen Zeilen-, die Beispiele 2 und 3 einen Spaltenzug.

Abbildung 11 zeigt zwei nicht erlaubte Züge bei Domino Juvavum.

[30] Wir bezeichnen dieses Spiel in manchen Fällen auch als *einfaches* oder *gewöhnliches* Juvavum, um den Unterschied zu Domino Juvavum zu betonen. [31] Man beachte, dass nur die ersten beiden auch in Cram erlaubt sind.

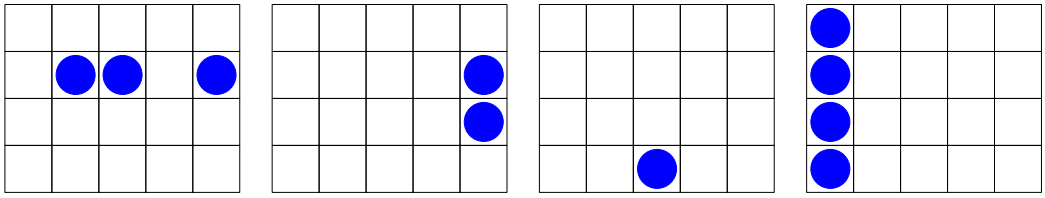


Abbildung 9: Beispiele für gültige Juvavum-Züge

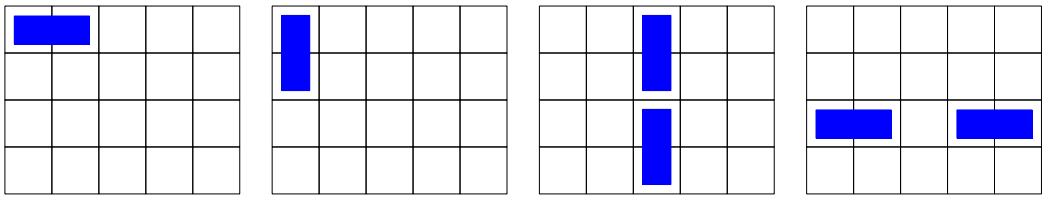


Abbildung 10: Beispiele für gültige Domino Juvavum-Züge

3.1.2 Spielende und Sieger

Sieger im normalen Spiel ist, wer den letzten Zug macht. Bei Juvavum bedeutet dies, dass kein Feld auf dem Spielbrett mehr frei ist, bei Domino Juvavum (und Cram) ist das Spielende erreicht, sobald keine zwei horizontal bzw. vertikal benachbarten Felder mehr frei sind, das heißt, kein weiteres Domino Platz hat. Sieger bei Juvavum beziehungsweise Domino Juvavum und Cram in der Misère-Form ist jener Spieler, der als Erster nicht mehr ziehen kann.

3.2 Definitionen und Schreibweisen

Definition 3.4 (Abkürzungen) Der Einfachheit halber verwenden wir im Folgenden einige Abkürzungen, die je nach Zusammenhang entweder ein Spiel oder dessen Startposition bezeichnen. $\mathcal{JUV}(m,n)$ steht für Juvavum auf einem $m \times n$-Feld. m bezeichne die Höhe, n die Breite des Spielfelds. Analog bezeichnen wir mit $\mathcal{DJUV}(m,n)$ und $\mathcal{CRAM}(m,n)$ die Spiele Domino Juvavum bzw. Cram auf den entsprechenden Spielfel-

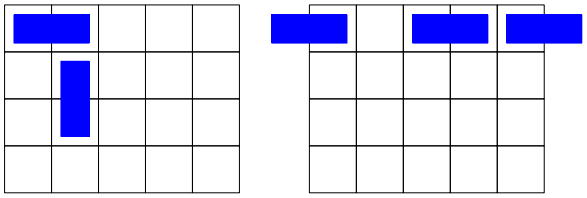


Abbildung 11: Beispiele für nicht erlaubte Domino Juvavum-Züge

dern. Die Abkürzungen $\mathcal{JUVM}$, $\mathcal{DJUVM}$ und $\mathcal{CRAMM}$ bezeichen jeweils die Misere-Form der Spiele.

Definition 3.5 Weiters bezeichne A den 1. und B den 2. Spieler.

Bemerkung 3.6 Da sich durch Drehung eines Spielbretts um 90 Grad nichts Wesentliches an der Analyse ändert, betrachten wir im Folgenden nur Spiele mit $m \leq n$.

Bemerkung 3.7 Man kann alle Varianten von Juvavum, Domino Juvavum und Cram auch in anderer Form spielen (ohne dabei die Struktur der Spiele zu verändern). Dazu beginnt man das Spiel nicht mit einem leeren Spielfeld, sondern mit einem, auf dem alle Felder mit Münzen belegt sind, und definiert einen Spielzug als Wegnehmen von Münzen nach bestimmten Regeln (zum Beispiel: nur zwei benachbarte bei Cram).

In dieser Version des Spiels wird, wie wir gleich sehen werden, der Zusammenhang mit anderen Spielen deutlicher.

Lemma 3.8 $\mathcal{JUV}(1, n)$ entspricht dem Spiel Nim mit einem Haufen aus n Elementen.

Beweis: Wir betrachten die im obigen Kommentar beschriebene Variante des Spiels. Da bei Juvavum stets beliebig viele 1×1-Steine ($\widehat{=}$ Münzen) in eine beliebige Zeile oder Spalte gesetzt werden dürfen, spielt in jeder Zeile bzw. Spalte nur die Anzahl der freien Felder eine Rolle, nicht jedoch deren Position. Wir dürfen daher beliebige Zeilen- und Spaltenvertauschungen vornehmen, ohne dass sich an den spieltheoretischen Eigenschaften (Anzahl der Nachfolger, Grundywert usw.) einer Position etwas ändert. Wir können daher bei $\mathcal{JUV}(1, n)$ (in der oben beschriebenen Variante) die Felder einer Position mit k Steinen ($0 \leq k \leq n$) (mit Hilfe von Spaltenvertauschungen) so ordnen, dass die ersten k Felder belegt und die restlichen $n - k$ frei sind, was gerade einem Nim-Haufen mit k Steinen entspricht. □

Korollar 3.9 $g(JUV(1, n)) = g(*n) = n$.

Beweis: Die Behauptung folgt aus Lemma 3.8 und Satz 2.54.

3.3 Bisherige Ergebnisse

Juvavum und Domino Juvavum wurden im Rahmen des Projektpraktikums „Spiele“an der Universität Salzburg (2004/05) eingeführt. Im Laufe dieser zweisemestrigen Lehrveranstaltung wurden einige Resultate erarbeitet, die im Folgenden wiedergegeben und ausgebaut werden. Darüber hinaus sind uns keine Arbeiten bekannt, in denen Juvavum oder Domino Juvavum (bzw. ein äquivalentes Spiel unter anderem Namen) Erwähnung finden.

Anders sieht es bei Cram aus, das 1974 von Gardner [Gar74] vorgestellt wurde und seither in vielen Publikationen vorkommt, wenn auch deutlich weniger oft als das ebenfalls als Cross-Cram bekannte Domineering.

Neue Resultate sind aber auch hier rar, da Gardner bereits die meisten Fälle selbst gelöst hat. Immerhin konnten Cowen und Dickau [CD98] zeigen, dass der 2. Spieler $\mathcal{CRAM}(5,5)$ gewinnen kann.[32]

[32] Das dazu benutzte Mathematica-Programm ist unter http://qcpages.qc.edu/ cowen/ erhältlich und sehr empfehlenswert. Das Analyseprogramm, das wir in dieser Arbeit vorstellen werden, konnte dieses Resultat bestätigen.

4 Spielfelder und Spiele (sinnvoll) speichern

Bevor wir mit der Analyse der Spiele beginnen, werden wir in diesem Abschnitt demonstrieren, wie eine Spielposition in möglichst einfacher Form dargestellt und gespeichert werden kann. Dies wird insbesondere für das Analyse-Programm, das wir am Ende dieser Arbeit vorstellen werden, von großem Nutzen sein.

Zudem werden wir zeigen, wie man den Verlauf einer gesamten Juvavum-Partie platzsparend und dennoch übersichtlich darstellen kann.

4.1 Speichern von Spielfeldern

Jede $m \times n$-Spielposition lässt sich durch eine Zahl aus $\mathbb{N}_0$ darstellen. Dabei wird jedes der $m \cdot n$ Felder durch eine Zweierpotenz repräsentiert. Die Felder werden zeilenweise, beginnend von links oben, nummeriert: zuerst alle Felder der ersten Zeile, dann die Felder der zweiten Zeile und so fort. Es spielt dabei keine Rolle, ob wir Juvavum-, Domino Juvavum- oder Crampositionen betrachten.

Sei x eine Spielposition auf einem $m \times n$-Brett. Wir nummerieren die Felder von x wie in Abbildung 12 gezeigt und definieren eine Funktion, die für alle $1 \leq i \leq m \cdot n$ angibt, ob das jeweilige Feld belegt ist oder nicht.

$$a_x(i) := \begin{cases} 2^i, & \text{wenn das } i\text{-te Feld von } x \text{ belegt ist} \\ 0, & \text{wenn das } i\text{-te Feld von } x \text{ nicht belegt ist} \end{cases}$$

2^0	2^1	2^2
2^3	2^4	2^5
2^6	2^7	2^8

Abbildung 12: Stellenwert der Felder auf einem 3×3-Feld

Summierung über alle i ergibt schließlich die Binärcodierung einer Spielposition x:

$$B(x) := \sum_{i=1}^{m \cdot n} a_x(i).$$

Um eine Spielposition eindeutig identifizieren zu können, ist neben der Binärcodierung noch die Angabe der Höhe und Breite (Anzahl der Zeilen bzw. Spalten) notwendig. Zum

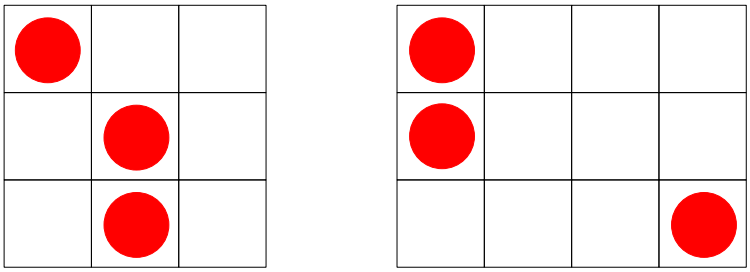


Abbildung 13: Das 3×3-Feld mit der Binärcodierung $145 = 2^0 + 2^4 + 2^7$ und das 3×4-Feld mit der gleichen Codierung

Beispiel stellt die Zahl 145 auf einem 3×4-Brett eine andere Spielposition dar als auf einem 3×3-Brett.

Definition 4.1 (Binärdarstellung einer Spielposition) Sei x eine $m \times n$-Spielposition und sei $B(x)$ wie oben angegeben. Dann nennen wir das Tripel $(B(x), m, n)$ die Binärdarstellung von x.

Bemerkung 4.2 Im Allgemeinen gilt $(B(x), m, n) \neq (B(x), n, m)$. Die Konvention, zuerst die Anzahl der Zeilen und danach die Anzahl der Spalten anzugeben, ist daher wichtig.

4.2 Speichern von Spielen

Oft möchte man nicht nur eine Spielposition, sondern den Verlauf einer gesamten Partie in sinnvoller Weise speichern. Die folgenden Abbildungen illustrieren ein $\mathcal{DJUV}(3,3)$-Spiel. Die Steine von A wurden blau, jene von B rot gefärbt.

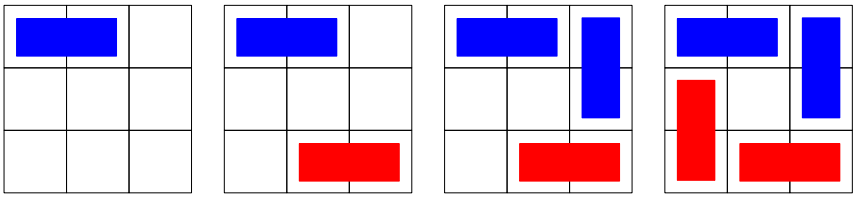


Abbildung 14: Demo-Spiel

Aus den obigen Abbildungen lässt sich der Spielverlauf vollständig rekonstruieren. Es ist jedoch vorteilhaft (insbesondere bei größeren Spielfeldern), eine platzsparendere Darstellung zu verwenden. Zum Beispiel kann man ein Spielfeld der entsprechenden Größe mit Zahlen $i \in \mathbb{N}$ füllen, wobei in einem Feld genau dann die Zahl i steht, wenn es im i-ten Zug belegt wurde. Auf diese Weise können auch größere Spiele relativ platzsparend dargestellt werden.

Für eine textbasierte Darstellung führen wir eine Schreibweise in Anlehnung an die Notation von Schachzügen ein. Dabei werden die Zeilen mit Buchstaben (A bis J) und die Spalten mit Ziffern (0 bis 9) bezeichnet.[33] So lässt sich jedes Feld über seine Koordinaten (z.B. B3) identifizieren. Für jeden Zug wird die Menge der Koordinaten der belegten Felder in einer Kurzform angegeben. Statt $\{A1, A2, A5, A6\}$ schreiben wir zum Beispiel $A1256$. Beide Notationen sind eindeutig. Im Folgenden werden sie für unsere 3×3-Beispielpartie angegeben:

1	1	3
4		3
4	2	2

Abbildung 15: Demo-Spiel in Kurzschreibweise

In der textbasierenden Schreibweise lautet dieses Spiel A01, C12, AB2, BC0.

[33] Auf diese Weise können wir Spiele bis zum 10×10-Feld beschreiben.

5 Analyse von Juvavum, Domino Juvavum und Cram

Bevor wir untersuchen, welcher Spieler bei bestimmten Juvavum- bzw. Domino Juvavum-Spielen im Vorteil ist und Grundywerte für einzelne Spielfelder berechnen, möchten wir in diesem Abschnitt ein paar allgemeine Überlegungen zur Komplexität dieser Spiele präsentieren. Die in diesem Abschnitt gezeigten Ergebnisse gelten sowohl für das normale Spiel als auch für die Misère-Form.

5.1 Juvavum

Satz 5.1 Die Anzahl der Spielpositionen bei $\mathcal{JUV}(m,n)$ ist $2^{m\cdot n}$.

Beweis: Bei Juvavum kann jede beliebige Belegung eines Spielfelds während des Spiels erreicht werden und stellt daher eine gültige Spielposition dar. Da es $m\cdot n$ Spielfelder gibt, ist die Anzahl der möglichen Belegungen $\sum_{k=0}^{m\cdot n}\binom{m\cdot n}{k} = 2^{m\cdot n}$. □

Wir betrachten einige Werte für den Fall $a := m = n$:

a	2^{a^2}
1	2
2	16
3	512
4	65536
5	33554432
10	$1.27\cdot 10^{30}$
100	$2\cdot 10^{3010}$

Natürlich interessieren wir uns auch für die Anzahl der Nachfolger einer Spielposition.

Definition 5.2 Sei x eine beliebige Juvavum-Spielposition. Dann bezeichnet $succ(x)$ die Anzahl der direkten Nachfolger dieser Position.[34]

Wir betrachten eine beliebige $m\times n$-Spielposition x von $\mathcal{JUV}(m,n)$. Sei $z(i)$ die Anzahl der freien Felder in der i-ten Zeile und $s(j)$ die Anzahl der freien Felder in der j-ten Spalte. Für jede Zeile bzw. Spalte gilt: Die Anzahl der möglichen Belegungen ist gerade 2 hoch die Anzahl der freien Felder. Um die Anzahl der möglichen Züge in einer Zeile bzw. Spalte zu erhalten, muss man davon lediglich die aktuelle Belegung abziehen. Summierung über alle Zeilen und Spalten ergibt eine erste Näherung:

$$succ(x) \approx \sum_{i=1}^{m}\left(2^{z(i)}-1\right) + \sum_{j=1}^{n}\left(2^{s(j)}-1\right) = \sum_{i=1}^{m}2^{z(i)} + \sum_{j=1}^{n}2^{s(j)} - (m+n).$$

Dieser Wert ist noch zu hoch, da die Nachfolger, bei denen nur ein Feld belegt wird, doppelt gezählt werden (einmal als Zeilen-, einmal als Spaltenzug). Dies lässt sich jedoch

[34] *succ* steht für successors (Nachfolger).

leicht korrigieren. Dazu genügt es sich zu überlegen, dass die Anzahl der möglichen Züge mit nur einem Stein genau die Zahl der freien Felder auf dem Spielbrett ist. Diese lässt sich als $\sum_{j=1}^{n} s(j)$ schreiben.[35] Wir erhalten schließlich

$$succ(x) = \sum_{i=1}^{m} 2^{z(i)} + \sum_{j=1}^{n} 2^{s(j)} - (m+n) - \sum_{j=1}^{n} s(j)$$

und nach kurzer Umformung das Ergebnis des folgenden Satzes.

Satz 5.3 (Anzahl der direkten Nachfolger einer Juvavum-Position) Sei x eine beliebige Spielposition von Juvavum. Dann gilt:

$$succ(x) = \sum_{i=1}^{m} 2^{z(i)} + \sum_{j=1}^{n} \left(2^{s(j)} - s(j)\right) - (m+n)$$

Die Anzahl der möglichen Züge von der Startposition eines beliebigen Juvavum-Spiels ergibt sich als Spezialfall des obigen Satzes:

Korollar 5.4 Seien $m, n \geq 2$ und sei $a \geq 2$. Dann gilt:

$$succ(\mathcal{JUV}(m,n)) = m \cdot (2^n - 1) + n \cdot (2^m - 1) - m \cdot n$$

$$succ(\mathcal{JUV}(a,a)) = a \cdot (2^{a+1} - 2 - a)$$

Beweis: Für ein leeres Spielfeld gilt $z(i) = n$ für alle Zeilen $1 \leq i \leq m$ und $s(j) = m$ für alle Spalten $1 \leq j \leq n$. Daraus folgt die erste Behauptung durch einsetzen. Sei $m = n$ und $a := m$. Daraus folgt unmittelbar die zweite Behauptung. □

Bemerkung 5.5 Für den Fall, dass m oder n gleich 1 ist, ergibt sich die Anzahl der Nachfolger ebenfalls aus der Überlegung, dass es in einer Zeile bzw. Spalte stets $2^{\text{Anzahl freie Felder}}$ mögliche Belegungen gibt. Es gilt

$$succ(\mathcal{JUV}(1,n)) = succ(\mathcal{JUV}(n,1)) = 2^n - 1$$

In der folgenden Tabelle geben wir die Anzahl der Nachfolger der Startposition von $\mathcal{JUV}(a,a)$ für ausgewählte a an.

a	$succ(\mathcal{JUV}(a,a))$
1	1
2	8
3	22
4	52
5	114
10	4072
100	$5.07 \cdot 10^{30}$
1000	$4.29 \cdot 10^{301}$

[35] Alternativ könnte man auch die Darstellung $\sum_{i=1}^{m} z(i)$ verwenden.

5.2 Domino Juvavum und Cram

Für Domino-Juvavum betrachten wir zunächst die Anzahl der Nachfolger einer Spielposition.

Definition 5.6 (Loch) Unter einem Loch der Länge n verstehen wir n nebeneinander liegende freie Felder (in einer Zeile oder Spalte), die auf beiden Seiten durch ein belegtes Feld oder den Spielfeldrand begrenzt sind.

Definition 5.7 Sei $d(n)$ die Anzahl der möglichen $\mathcal{DJUV}$-Spielzüge in einem Loch der Länge n.

Lemma 5.8 Es gilt:

$$d(n) = 1 + d(n-2) + d(n-1).$$
$$d(0) = 0, \quad d(1) = 0$$

Beweis: Diese Gleichung ergibt sich aus folgender Überlegung:

Es gibt zwei Möglichkeiten das erste Domino zu legen (siehe Abbildung 16). Entweder es wird an die erste mögliche Position gesetzt. In diesem Fall bleiben $n-2$ Felder für die restlichen Dominosteine frei. Dies entspricht dem ersten Teil der Rekursionsbeziehung $(1 + d(n-2))$. Oder das erste Feld wird freigelassen, wodurch noch $n-1$ Felder besetzt werden können $(d(n-1))$.

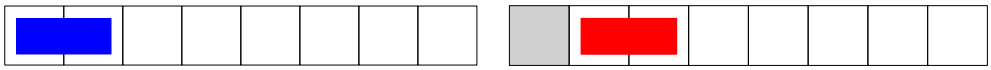


Abbildung 16: Mögliche Positionen für das erste Domino

Addition mit 1 ergibt:

$$d(n) + 1 = (d(n-2) + 1) + (d(n-1) + 1).$$

Wir setzen nun $F_0 := 0$ und $F_n := d(n-1)+1 \quad \forall n \geq 1$, wodurch wir die Fibonacci-Folge erhalten:

$$F_n = F_{n-1} + F_{n-2} \quad F_0 = 0, F_1 = 1.$$

Es gilt schließlich:

Korollar 5.9 $d(n) = F_{n+1} - 1 \quad \forall n \geq 1$.

Beweis: Die Aussage folgt unmittelbar aus den obigen Überlegungen.

Definition 5.10 Sei x eine beliebige Domino Juvavum-Spielposition. Dann bezeichnet $succ(x)$, die Anzahl der direkten Nachfolger dieser Position.[36]

[36] Die Bezeichnung ist die gleiche wie bei einfachem Juvavum. Aus dem Zusammenhang wird aber stets klar sein, welches der beiden Spiele gemeint ist.

Korollar 5.11 Sei $succ(\mathcal{DJUV}(m,n))$ die Anzahl der Nachfolger der Startposition von $\mathcal{DJUV}(m,n)$. Dann gilt:

$$succ(\mathcal{DJUV}(m,n)) = n \cdot d(m) + m \cdot d(n).$$

Ist $a := m = n$, so gilt:

$$succ(\mathcal{DJUV}(a,a)) = 2a \cdot d(a).$$

Beweis: Jede Zeile eines leeren $m \times n$-Spielbretts entspricht einem Loch der Länge n, jede Spalte einem Loch der Länge m. In einem solchen Loch gibt es $d(n)$ bzw. $d(m)$ Nachfolger. Da ein Zug stets nur eine Zeile oder Spalte verändert, ergibt sich die Gesamtanzahl aller möglichen Zeilenzüge als $m \cdot d(n)$, die Gesamtzahl der Spaltenzüge als $n \cdot d(m)$. Addition führt schließlich zum behaupteten Resultat. □

Beispiel 5.12 Von der Startposition von $\mathcal{DJUV}(3,5)$ beziehungsweise $\mathcal{DJUVM}(3,5)$ gibt es $3 \cdot d(5) + 5 \cdot d(3) = 3 \cdot 7 + 5 \cdot 2 = 31$ mögliche erste Züge.

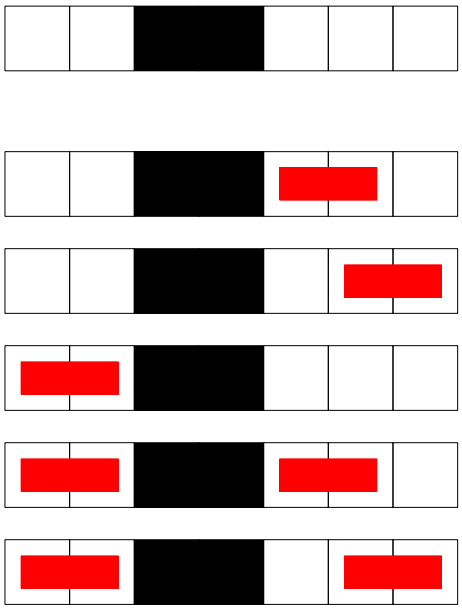


Abbildung 17: Eine $\mathcal{DJUV}(1,7)$-Position und ihre Nachfolger

Satz 5.13 In einer Zeile beziehungsweise Spalte mit k Löchern mit Längen $n_1 \dots n_k$ gibt es $\prod_{i=1}^{k} (1 + d(n_i)) - 1 = \prod_{i=1}^{k} F_{n_i+1} - 1$ mögliche Züge.

Beweis: In einem Loch der Länge n_i gibt es nach den vorigen Überlegungen $d(n_i)$ mögliche Nachfolger. Wir berücksichtigen die Möglichkeit, ein Loch leer zu lassen, indem wir 1 addieren. Da die Belegung eines Lochs keine Einschränkungen für die Belegung der restlichen Löcher in der Zeile bzw. Spalte darstellt, multiplizieren wir diese Werte für

alle k Löcher, um die Anzahl aller Nachfolger in dieser Zeile bzw. Spalte zu erhalten. Schließlich müssen wir noch den Fall abziehen, in dem alle Löcher frei gelassen werden, da er keinen gültigen Zug darstellt. □

Beispiel 5.14 Abbildung 17 zeigt eine $\mathcal{DJUV}$-Zeile mit Löchern der Längen 2 und 3 sowie alle möglichen Nachfolgebelegungen dieser Zeile. Davon gibt es nach dem obigem Satz $F_3 \cdot F_4 - 1 = 2 \cdot 3 - 1 = 5$.

Auch für Domino Juvavum geben wir eine Tabelle mit den Nachfolgern einer $a \times a$-Startposition für einige Werte an und stellen sie den Ergebnissen für Juvavum gegenüber.

a	$succ(\mathcal{DJUV}(a,a))$	$succ(\mathcal{JUV}(a,a))$
1	0	1
2	4	8
3	12	22
4	32	52
5	70	114
10	1760	4072
100	$7.01 \cdot 10^{21}$	$5.07 \cdot 10^{30}$
1000	$8.68 \cdot 10^{211}$	$4.29 \cdot 10^{301}$

Eine Möglichkeit, die n-te Fibonacci-Zahl direkt zu berechnen, ist nach dem französischen Mathematiker Jacques Philippe Marie Binet (1786-1856) benannt. Sie wurde allerdings auch schon früher von Euler (1765) und de Moivre (1730) publiziert (vgl. [GKP94] und [Knu97]).

Satz 5.15 (Formel von Binet) Sei F_n die n-te Fibonacci-Zahl und sei $\varphi = \frac{1+\sqrt{5}}{2} = 1.618\ldots$ der goldene Schnitt (und $1 - \varphi = \frac{1-\sqrt{5}}{2}$). Dann gilt:

$$F_n = \frac{1}{\sqrt{5}} \left[\varphi^n - (1-\varphi)^n\right]$$

Ein Beweis findet sich unter anderem in [CW08].

Korollar 5.16 Sei $d(n)$ die Anzahl der möglichen $\mathcal{DJUV}$-Spielzüge in einem Loch der Länge n. Dann gilt:

$$d(n) = \frac{1}{\sqrt{5}} \left[\varphi^{n+1} - (1-\varphi)^{n+1}\right] - 1$$

.

Beweis: Einsetzen. □

Interessant ist auch das folgende Lemma.

Lemma 5.17

$$F_n = \lfloor \frac{\varphi^n}{\sqrt{5}} + \frac{1}{2} \rfloor$$

Beweis: Es gilt $\left|\frac{(1-\varphi)^n}{\sqrt{5}}\right| < \frac{1}{2}$. Dies ist mittels Fallunterscheidung leicht nachzuweisen:

- Für $n = 0$ ist die Aussage erfüllt: $\frac{1}{\sqrt{5}} \approx 0.447 < \frac{1}{2}$.
- Für $n = 2$ gilt die Behauptung ebenso: $\frac{1}{\sqrt{5}}(1-\varphi)^2 \approx 0.170$. Sei $n = 2m, m \in \mathbb{N}$. Dann ist die Folge $(1-\varphi)^n = (1-\varphi)^{2m} \approx (0.381)^m$ monoton fallend in m und alle Folgenglieder sind positiv. Für gerades n folgt daher: $0 < \frac{(1-\varphi)^n}{\sqrt{5}} < \frac{1}{2}$.
- Analog zeigt man für den Fall n ungerade, dass $\frac{-1}{2} < \frac{(1-\varphi)^n}{\sqrt{5}} < 0$.

Es gilt schließlich in allen Fällen $\left|\frac{(1-\varphi)^n}{\sqrt{5}}\right| < \frac{1}{2}$. Damit folgt aus der Formel von Binet

$$F_n \in \left(\frac{\varphi^n}{\sqrt{5}} - \frac{1}{2}, \frac{\varphi^n}{\sqrt{5}} + \frac{1}{2}\right)$$

und wegen $F_n \in \mathbb{N}$ die Behauptung. □

Wir gehen einen Schritt weiter und vernachlässigen den Term $(1-\varphi)^n$, der für $n \to \infty$ gegen 0 strebt, ganz. Dadurch erhalten wir folgende Näherung:

$$F_n \approx \frac{\varphi^n}{\sqrt{5}}.$$

Korollar 5.18 Damit ergeben sich folgende Näherungen für die Anzahl der Nachfolger einer Domino Juvavum-Startposition:

$$succ(\mathcal{DJUV}(m,n)) \approx \frac{1}{\sqrt{5}}\left(n \cdot \varphi^{m+1} + m \cdot \varphi^{n+1}\right) - (m+n)$$

$$succ(\mathcal{DJUV}(a,a)) \approx \frac{1}{\sqrt{5}}\left(2a \cdot \varphi^{a+1}\right) - 2a$$

Als nächstes versuchen wir einige Aussagen über die Anzahl aller möglichen Spielpositionen eines $\mathcal{DJUV}(m,n)$-Spiels zu treffen. Dazu machen wir zunächst eine wichtige Feststellung, die vor allem für die Computeranalyse von Domino Juvavum und Cram von Bedeutung sein wird.

Im Allgemeinen gibt es mehrere Anordnungen von Dominosteinen, die die gleiche Spielposition darstellen.

Für die Berechnung von Grundywerten ist die Anordnung der Steine allerdings unerheblich. Es genügt zu wissen, ob ein Feld belegt oder frei ist. Wir führen dazu den Begriff der Positionsmatrix ein.

Definition 5.19 (Positionsmatrix eines Spielfeld) Sei x eine Spielposition auf einem $m \times n$-Feld. Sei weiters m_{ij}=0, wenn das Feld mit Zeilenindex i und Spaltenindex j von x frei ist, und $m_{ij} = 1$, wenn es belegt ist. Die Matrix $MD(x) = (m_{ij})_{1 \le i \le n, 1 \le j \le m}$ heißt Positionsmatrix von x.

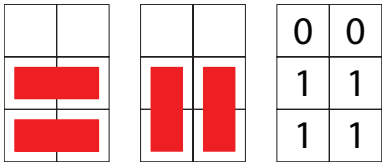


Abbildung 18: Zwei Dominobelegungen auf einem 3×2-Feld, die die gleiche Spielposition erzeugen

Definition 5.20 (Dominobelegung) Unter einer Dominobelegung eines Spielbretts verstehen wir eine beliebige Anordnung von Dominos auf diesem Spielbrett, die nach den Regeln von Domino Juvavum und Cram erlaubt ist. Das bedeutet: Die Dominos bedecken je zwei horizontal oder vertikal benachbarte Felder, ragen nicht über den Spielfeldrand hinaus und jedes Feld ist durch höchstens ein Domino belegt.

Zur klareren Unterscheidung präzisieren wir unsere Definition einer Domino Juvavum-Spielposition.

Definition 5.21 (Spielposition) Eine $m \times n$-Matrix mit Elementen aus $\{0, 1\}$ ist eine Spielposition von $\mathcal{DJUV}(m, n)$, wenn sie die Positionsmatrix einer Dominobelegung eines $m \times n$-Spielbretts ist.

Definition 5.22 (Menge der Spielpositionen) Die Menge aller Spielposition von $\mathcal{DJUV}(m, n)$ ist die Menge aller Positionsmatrizen von Dominobelegungen eines $m \times n$-Spielbretts. Wir bezeichnen sie mit $DP(m, n)$.

Bemerkung 5.23 Beide Definitionen gelten analog auch für das Spiel Cram.

Definition 5.24 Mit $posd(m, n)$ bezeichnen wir die Anzahl aller möglichen Spielpositionen eines $\mathcal{DJUV}(m, n)$-Spiels, also die Kardinalität der Menge $DP(m, n)$.

Definition 5.25 Mit $assd(m, n)$ bezeichnen wir die Anzahl aller möglichen Dominobelegungen eines $\mathcal{DJUV}(m, n)$-Spiels.[37]

Wie wir an folgendem Beispiel sehen werden, ist diese Unterscheidung wichtig.

Beispiel 5.26 Es gibt 7 Möglichkeiten, Dominos (nach unseren Regeln) auf ein 2×2-Feld zu legen. Diese werden in Abbildung 19 gezeigt. Allerdings existieren bei $\mathcal{DJUV}(2, 2)$ nur 6 Spielpositionen (Positionsmatrizen), die in Abbildung 20 zu sehen sind. Die zwei Möglichkeiten, alle 4 Felder zu belegen (durch zwei horizontale oder zwei vertikale Dominos), führen auf die gleiche Spielposition.

Korollar 5.27 Im Allgemeinen gilt $assd(m, n) > posd(m, n)$.

[37] *ass* steht für assignments (Belegungen).

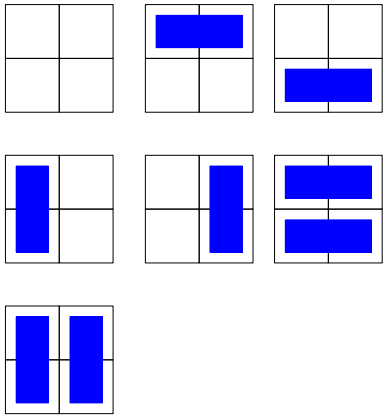


Abbildung 19: Die 7 Dominobelegungen bei $\mathcal{DJUV}(2,2)$

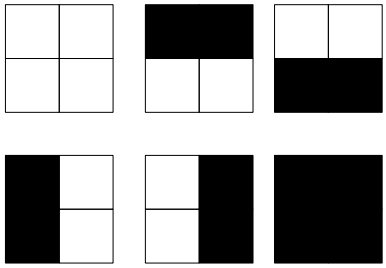


Abbildung 20: Die 6 $\mathcal{DJUV}(2,2)$-Spielpositionen

Bemerkung 5.28 Im Gegensatz dazu ist bei normalem Juvavum die Anzahl der möglichen Belegungen eines Spielfelds gleich der Anzahl der Spielpositionen .

Wir betrachten nun $posd(m,n)$ für einige spezielle m.

Satz 5.29 $posd(1,n) = F_{n+1}$.

Beweis: Mit Ausnahme des leeren Spielbretts kann jede mögliche Position bereits mit dem ersten Zug erreicht werden. Daher ist die Anzahl der möglichen Positionen gerade $d(n)+1 = F_{n+1}$. □

Satz 5.30 $posd(2,n+1) = 3 \cdot posd(2,n) \quad \forall n \geq 2$ und $posd(2,2) = 6$.

Beweis: Wir betrachten ein $2 \times n$-Spielfeld. Für jede Spalte, insbesondere auch für die n-te Spalte gibt es 4 mögliche Belegungen. Entweder sind beide Felder belegt (Fall 1), beide Felder frei (Fall 2) oder je ein Feld frei und das andere belegt (Fall 3 und 4).

Sei k die Anzahl jener Spielpositionen von $\mathcal{DJUV}(2,n)$, bei denen beide Felder der n-ten Spalte frei sind (Fall 2). Sei x eine beliebige dieser Positionen. Dann gibt es genau eine Position, bei der die n-te Spalte belegt ist und die sich in den übrigen $n-1$ Spalten nicht von x unterscheidet. Daraus folgt, dass es insgesamt auch k Positionen von $\mathcal{DJUV}(2,n)$ gibt, bei denen die n-te Spalte belegt ist (Fall 1).

Sei l die Anzahl jener Spielpositionen von $\mathcal{DJUV}(2,n)$, bei denen das obere Feld der n-ten Spalte frei und das untere belegt ist (Fall 3). Durch horizontales Spiegeln entsteht ebenfalls eine gültige Spielposition, bei der das untere Feld belegt und das obere frei ist (Fall 4). Folglich ist auch die Anzahl dieser Spielpositionen gleich l. Da es nur diese 4 Fälle gibt, ist klarerweise $posd(2,n) = 2k + 2l$.

Wir fügen nun eine weitere leere Spalte rechts an das $2 \times n$-Feld an und untersuchen, wie viele mögliche neue Positionen dadurch entstehen. Dazu betrachten wir für jede der $posd(2,n)$ Positionen die n-te Spalte und unterscheiden die vier Fälle von oben. Im Fall 2 erhalten wir zwei mögliche Positionen (entweder die $n+1$-Spalte bleibt leer oder wir setzen dort ein Domino). Im Fall 3 bzw. 4 gibt es jeweils 3 Möglichkeiten: Entweder wir setzen ein horizontales Domino, ein vertikales Domino oder aber gar kein Domino. Im Fall 1 erhalten wir ein leeres 2×2-Feld. Man kann sich leicht überlegen, dass es dafür 6 mögliche Belegungen gibt, von denen allerdings zwei auf Positionen führen, die schon durch Fall 2 abgedeckt sind. Der 1. Fall liefert also für jede $2\times n$-Position 4 neue $2\times(n+1)$-Positionen.

Die Anzahl aller Positionen von $\mathcal{DJUV}(2,n+1)$ ergibt sich somit zu

$$posd(2,n+1) = 2k + 4k + 3l + 3l = 3 \cdot (2k + 2l) = 3 \cdot posd(2,n).$$

□

Korollar 5.31

$$posd(2,n) = 2 \cdot 3^{n-1} \quad \forall n \geq 2.$$

Beweis: Nach obigem Satz gilt:

$$\begin{aligned} posd(2,n) &= 3 \cdot posd(2,n-1) \\ &= 3^2 \cdot posd(2,n-2) \\ &\vdots \\ &= 3^{n-2} \cdot posd(2,2) \\ &= 3^{n-2} \cdot 6 \\ &= 2 \cdot 3^{n-1}. \end{aligned}$$

□

Mit etwas mehr Aufwand sollte es möglich sein, ähnliche Rekursionsformeln für die Berechnung von $pos(m,n)$ für $m > 2$ zu finden. Der Ansatz ist allerdings bereits im Fall $m = 3$ deutlich komplizierter.

Für ein allgemeines Resultat über die Anzahl der möglichen Dominobelegungen betrachten wir eine andere Darstellung des Spiels Domino Juvavum:

Definition 5.32 (Dominograph) Sei $V := \{x_{ij}, \quad 1 \leq i \leq n, 1 \leq j \leq m\}$, wobei jedes x_{ij} ein Feld eines $m \times n$-Spielbretts repräsentiert. Sei E jene Menge, die ein Paar (x_{ij}, x_{kl}) mit $1 \leq i < k \leq n, 1 \leq j < l \leq m$ genau dann enthält, wenn die durch x_{ij} und x_{kl} dargestellten Felder (horizontal oder vertikal) benachbart sind. Den durch (V,E) festgelegten ungerichteten Graphen bezeichnen wir mit $DG(m,n)$. Er enthält die Felder des Spielbretts als Ecken sowie Kanten zwischen je zwei benachbarten Feldern.

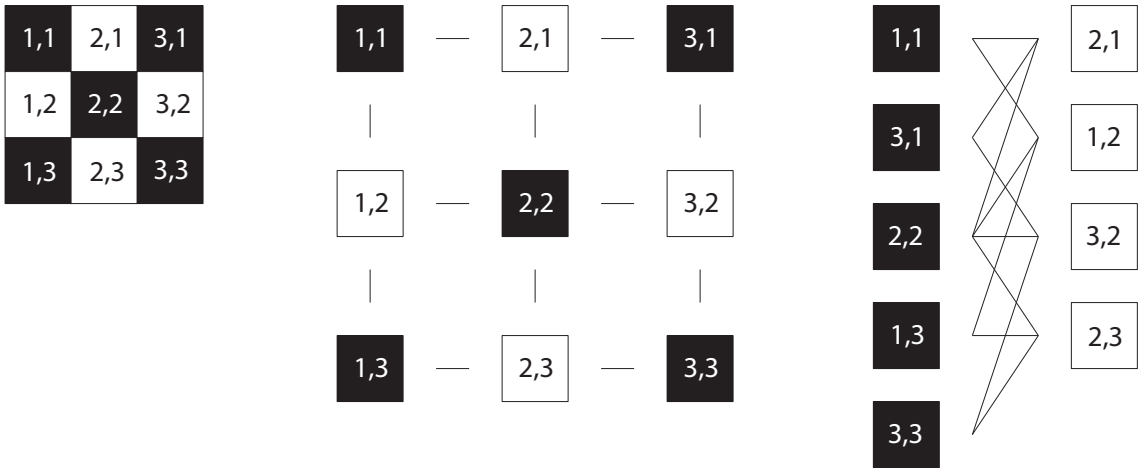


Abbildung 21: Ein 3×3-Brett und zwei Darstellungen von $DG(3,3)$

Lemma 5.33 $DG(m,n)$ ist bipartit.

Beweis: Sei V die Eckenmenge von $DG(m,n)$. Wir betrachten zwei Teilmengen von V. $A := \{x_{ij}, i+j \text{ gerade}\}$ und $B := \{x_{ij}, i+j \text{ ungerade}\}$. Dann gilt: $V = A \cup B$ und $A \cap B = \varnothing$. Dadurch ergibt sich eine Färbung des Spielegraphen. Von zwei benachbarten Feldern liegt stets eines in A und eines in B. Kanten verlaufen nach Definition nur zwischen benachbarten Feldern. Jede Kante verläuft daher zwischen einer Ecke aus A und einer Ecke aus B. Folglich ist $DG(m,n)$ bipartit. □

Definition 5.34 Sei x eine beliebige $\mathcal{DJUV}$-Position. Dann definieren wir $DG(x)$ analog zu $DG(m,n)$ mit der Einschränkung, dass jene Ecken und Kanten aus dem Graphen entfernt werden, die bereits durch ein Domino belegt sind.

Lemma 5.35 Sei x eine beliebige $\mathcal{DJUV}$-Position. Dann gilt: $DG(x)$ ist ein Teilgraph von $DG(m,n)$.

Beweis: Folgt sofort aus der Definition. □

Korollar 5.36 Sei x eine beliebige $\mathcal{DJUV}$-Position. Dann gilt: $DG(x)$ ist bipartit.

Beweis: Es gilt: Jeder Teilgraph eines bipartiten Graphen ist bipartit. Damit folgt die Behauptung aus Lemma 5.33 und Lemma 5.35. □

Bemerkung 5.37 Sei x eine $\mathcal{DJUV}(m,n)$-Position. Jede Kante von $DG(x)$ verbindet zwei benachbarte freie Felder und stellt folglich eine potentielle Position für ein Domino dar.

Definition 5.38 (Matching) Sei $G = (V,E)$ ein Graph. Unter einem Matching M verstehen wir eine Menge $M \subseteq E$, deren Kanten jede Ecke aus V höchstens einmal enthalten. Das bedeutet: Ein Matching ist eine Menge von Kanten ohne gemeinsame Ecke.

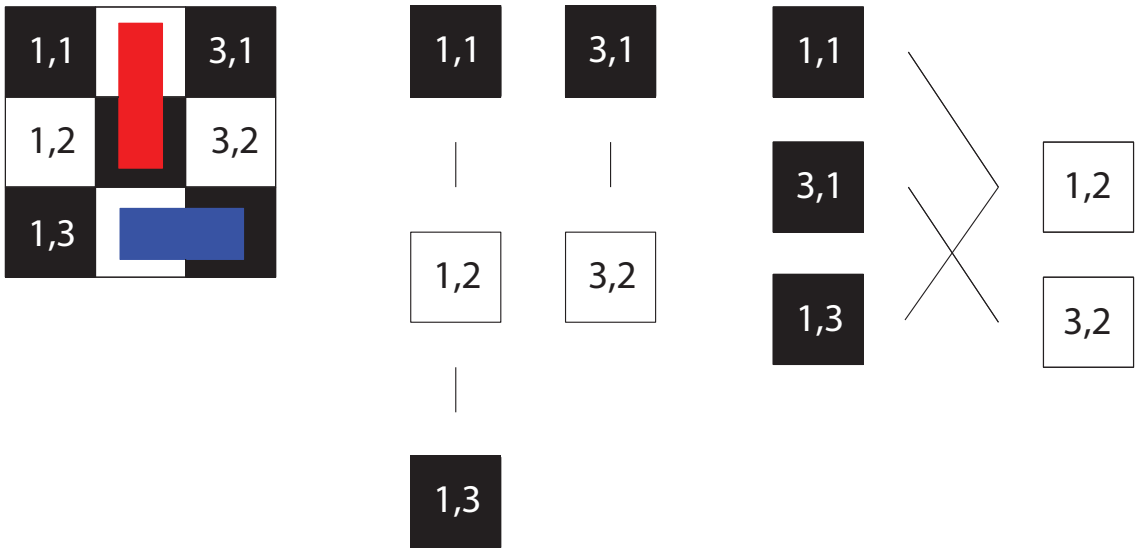


Abbildung 22: Eine 3 × 3-Spielposition und zwei Darstellungen des dazugehörigen Dominographen

Definition 5.39 (Maximales Matching) Unter einem maximalen Matching verstehen wir ein Matching mit der Eigenschaft, dass mit jeder weiteren Kante, die zu M hinzugefügt wird, die Matchingeigenschaft verloren geht: M ist maximal genau dann, wenn $M \cup \{e\}$ kein Matching ist $\forall e \in E \backslash M$.

Definition 5.40 (Maximum Matching) Unter einem Maximum Matching verstehen wir ein Matching, das die größtmögliche Anzahl von Kanten enthält.

Definition 5.41 (Perfektes Matching) Sei G ein Graph. Ein Matching M von G heißt perfekt, wenn es jede Ecke von G enthält.

Definition 5.42 (Matching-Zahl) Die Matching-Zahl eines Graphen G ist die Anzahl der Kanten in einem Maximum Matching von G. Wir bezeichnen sie mit $mn(G)$.[38]

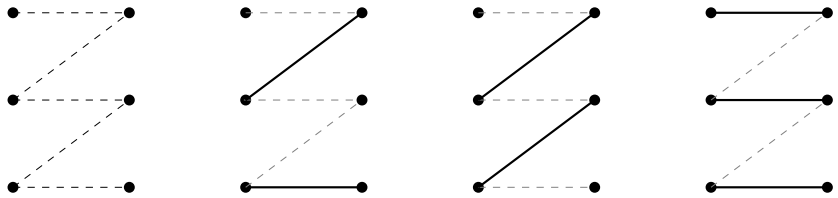


Abbildung 23: Ein Graph, zwei maximale Matchings und ein Maximum Matching

[38] *mn* steht für *matching number*.

Lemma 5.43 Es gilt:

1. M ist ein Maximum Matching $\Rightarrow$ M ist ein maximales Matching. Die Umkehrung gilt im Allgemeinen nicht.

2. Ein Maximum Matching eines Graphen ist im Allgemeinen nicht eindeutig.

3. Jedes perfekte Matching ist ein Maximum Matching und somit auch ein maximales Matching.

4. Ein perfektes Matching von G kann nur existieren, wenn die Anzahl der Ecken von G gerade ist.

ad 1.) Die erste Aussage folgt sofort aus den Definitionen. Als Gegenbeispiel für die zweite Behauptung betrachte man z. B. die zwei maximalen Matchings in Abbildung 23.
ad 2.) Abbildung 24 zeigt drei verschiedene Maximum Matchings eines Graphen.
ad 3.) Ein perfektes Matching von G enthält nach Definition alle Ecken von G. Es gibt folglich keine Möglichkeit, eine Kante zum Matching hinzuzufügen. Daher ist das Matching auch ein Maximum Matching und nach Punkt 1 auch ein maximales Matching.
ad 4.) Sei G ein Graph mit einer ungeraden Anzahl von Ecken. Da jede Kante zwei Ecken verbindet, muss bei jedem Matching zumindest eine Ecke „frei“bleiben. G kann daher kein perfektes Matching haben.

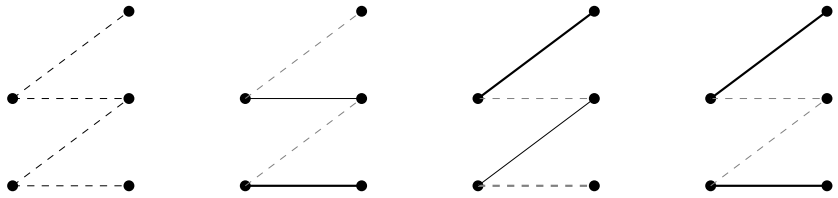


Abbildung 24: Ein Graph und drei verschiedene Maximum Matchings

Korollar 5.44 (Eigenschaften von DG) Es gilt:

- Die Anzahl der Dominobelegungen eines beliebigen Domino Juvavum-Spiels x entspricht der Anzahl aller Matchings von $DG(x)$[39]. Einige Anmerkungen dazu finden sich im Anhang C.

- Die maximale Anzahl von Zügen von einer Position x bis zum Spielende entspricht der Matching-Zahl $mn(DG(x))$.

- Eine Ecke von $DG(m,n)$ hat im Durchschnitt $4 - \frac{2(m+n)}{m \cdot n}$ Nachbarn. Das bedeutet: $DG(m,n)$ ist für große Spielfelder „in etwa“4-regulär.

[39] Diese Aussage gilt analog auch für $\mathcal{DJUVM}$, $\mathcal{CRAM}$ und $\mathcal{CRAMM}$, da die möglichen Spielpositionen bei diesen Spielen dieselben wie bei Domino Juvavum sind.

Beweis:

Die ersten beiden Aussagen folgen aus den jeweiligen Definitionen sowie der Tatsache, dass das Setzen eines Dominos als Entfernen einer Kante im Dominograph interpretiert werden kann.

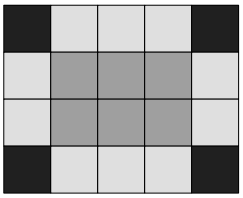


Abbildung 25: Die Felder eines Spielbretts haben stets 2, 3 oder 4 Nachbarn

Zum Beweis der letzten Aussage betrachten wir ein leeres $m \times n$-Spielfeld und stellen fest, dass jedes Feld zu 2, 3 oder 4 anderen benachbart ist. Die Felder eines $(m-2)\times(n-2)$-Feldes in der Mitte des Spielbretts haben jeweils 4 Nachbarn. Bei den Feldern am Rand unterscheiden wir die 4 Eckfelder, die jeweils 2 Nachbarn haben sowie die $2(m-2)+2(n-2)$ Felder mit je 3 Nachbarn. Sei nun k die durchschnittliche Anzahl der Nachbarn. Dann folgt:

$$k = \frac{4 \cdot [(m-2)(n-2)] + 3 \cdot [(2(m-2) + 2(n-2)] + 2 \cdot 4}{m \cdot n} = 4 - \frac{2(m+n)}{m \cdot n}.$$

□

Bemerkung 5.45 Wie wir gerade gesehen haben, endet jedes Spiel mit Startposition x nach spätestens $mn(x)$ Zügen. Natürlich kann eine Partie auch deutlich schneller enden, da mehrere Dominos in einem Zug gesetzt werden können. Auch bei Cram ist ein Spiel nicht notwendigerweise erst nach $mn(x)$ Zügen zu Ende, da isolierte Felder (freie Felder, die nach allen vier Seiten durch belegte Felder oder den Spielfeldrand begrenzt sind) auftreten können. Das Erzeugen solcher Situationen ist sogar eine wichtige taktische Komponente aller Domino-Spiele (Domino Juvavum, Cram, Domineering und anderen). Trotz allem kann die Kenntnis von $mn(x)$ von Vorteil sein.[40]

Bemerkung 5.46 Es gibt eine Reihe von Algorithmen, um ein Maximum Matching eines Graphen zu finden. Ein einfacher Ansatz für bipartite Graphen wird unter anderem in [SL93] beschrieben. Dieser Algorithmus hat eine Laufzeit von $\mathcal{O}(n \cdot m)$, wobei n die Anzahl der Ecken und m die Anzahl der Kanten des Graphen bezeichnet. Eine Verbesserung stellt der Algorithmus von Hopcroft-Karp dar, der mit $\mathcal{O}(\sqrt{n} \cdot m)$ Schritten auskommt [HK73]. Für dicht besetzte Graphen bietet sich ein Algorithmus von Alt, Blum, Melhorn und Paul

[40] Es sollte möglich sein, basierend auf $mn(x)$ sowie einigen anderen Eigenschaften einer Position (Anzahl freier Felder, Anzahl isolierter Felder usw.), eine gute Bewertungsfunktion für Spielpositionen von Domino Juvavum zu finden. Alle Versuche in diese Richtung verliefen bislang allerdings erfolglos.

mit Komplexität $\mathcal{O}\left(\sqrt{n^3 \cdot m/log(n)}\right)$ an [ABMP91]. Wie wir gerade gesehen haben, sind die Graphen $DG(x)$ im Allgemeinen sehr dünn besetzt, weshalb dies in unserem Fall keine Verbesserung gegenüber Hopcroft-Karp darstellt. Zudem sind die Dominographen für alle in einem tatsächlichen Spiel auftretenden Spielfeldgrößen so klein, dass es mit modernen Computern kaum einen Unterschied macht, welchen Matching-Algorithmus man anwendet. Einen Überblick über diverse Matching-Algorithmen und ihre Laufzeiten bietet zum Beispiel [SL93]. Eine allgemeine Einführung zum Thema Matching in Graphen bietet das Buch [LP86] von Lovász und Plummer.

Man kann das Finden eines Maximum Matchings auch als Spezialfall eines Maximum Flow-Problems auffassen, für dessen Lösung es eine Vielzahl von Algorithmen (wie zum Beispiel jenen von Ford-Fulkerson [FF62]) gibt.

Zur weiteren Analyse der Dominographen führen wir den Begriff der Dichte eines Graphen ein.

Definition 5.47 Sei G ein ungerichteter Graph mit n Ecken und m Kanten. Dann verstehen wir unter der Dichte von G den Wert

$$D(G) = \frac{2m}{n \cdot (n-1)}.$$

Lemma 5.48 Es gilt:

- $D(G) = 1$, wenn G vollständig ist.
- $D(G) = 0$, wenn G keine Kante enthält.
- $0 \leq D(G) \leq 1$ für alle Graphen G.

Beweis: Die erste Behauptung folgt, da die maximale Anzahl an Kanten gerade $\frac{1}{2}n(n-1)$ ist. Die beiden anderen sind trivial. □

Wir betrachten die Dichte von $DG(m, n)$.

Satz 5.49 Es gilt:

- $D(DG(m,n)) = \frac{2(2m \cdot n - m - n)}{m \cdot n \cdot (m \cdot n - 1)} \quad \forall m > 1, n \geq 1$
- $D(DG(a,a)) = \frac{4}{a \cdot (a+1)} \quad \forall a > 1.$
- $D(DG(1,1)) = 0$

Beweis: Es reicht zu überlegen, dass $DG(m,n)$ stets $m \cdot n$ Ecken und $m \cdot (n-1) + n \cdot (m-1)$ Kanten hat. Daraus folgt die erste Behauptung:

$$\begin{aligned} D(DG(m,n)) &= \frac{2[m(n-1)+n(m-1)]}{m \cdot n \cdot (m \cdot n - 1)} \\ &= \frac{2(2m \cdot n - m - n)}{m \cdot n \cdot (m \cdot n - 1)}. \end{aligned}$$

Die zweite Behauptung folgt durch Einsetzen. Die dritte ist trivial, da $DG(1,1)$ aus einer Ecke und keiner Kante besteht. □

a	$D(DG(a,a))$
1	0
2	0.667
3	0.333
4	0.200
5	0.133
10	0.036
100	0.00039
1000	$3.99 \cdot 10^{-6}$

Bemerkung 5.50 Bis auf sehr kleine Spielfelder sind die Graphen $DG(x)$ sehr dünn besetzt. Es gilt zudem

$$\lim_{m \cdot n \to \infty} D(DG(m,n)) = 0.$$

Dies ist auch intuitiv klar, da die Zahl der möglichen Kanten, die von einer Ecke ausgehen, auch bei wachsender Anzahl der Ecken auf 4 beschränkt bleibt (da jedes Feld höchstens 4 horizontale und vertikale Nachbarn hat).

5.3 Anzahl der ersten Spielzüge bei Juvavum und Domino Juvavum

In diesem Abschnitt werden wir die Größenordnung der Anzahl der Spielpositionen von Juvavum und Domino Juvavum vergleichen. Wir beschränken uns der Einfachheit halber darauf, Aussagen über die Nachfolger eines leeren $a \times a$-Spielbretts zu treffen.

Definition 5.51 Seien f und g zwei Funktionen. f und g heißen asymptotisch gleich, wenn $lim_{n \to \infty} \frac{f(n)}{g(n)} = 1$. Schreibweise: $f \sim g$.

Mit Hilfe der Darstellungen von *succ* für Juvavum sowie Domino Juvavum-Positionen aus dem vorigen Abschnitt ergibt sich das folgende Lemma.

Lemma 5.52

$$succ(\mathcal{JUV}(a,a)) \sim a \cdot 2^{a+1}$$

$$succ(\mathcal{DJUV}(a,a)) \sim \frac{2a}{\sqrt{5}}\varphi^{a+1} \approx 0.894a \cdot 1.618^{a+1}$$

Beweis: Zu zeigen ist

$$\lim_{a\to\infty} \frac{succ(\mathcal{JUV}(a,a))}{a \cdot 2^{a+1}} = 1$$

bzw.

$$\lim_{a\to\infty} \frac{succ(\mathcal{JUV}(a,a))}{\frac{2a}{\sqrt{5}} \cdot \varphi^{a+1}} = 1$$

Dies ist mit Hilfe der Folgerungen 5.4 und 5.18 leicht nachzuweisen. □

Mit Hilfe der $\mathcal{O}$-Notation folgt daraus sofort:

Satz 5.53

$$succ(\mathcal{JUV}(a,a)) \in \mathcal{O}(a \cdot 2^{a+1})$$
$$succ(\mathcal{DJUV}(a,a)) \in \mathcal{O}(a \cdot \varphi^{a+1})$$

Während sich die Anzahl der 1. Spielzüge bei Juvavum für wachsende Spiefeldgrößen „in etwa“wie Zweierpotenzen verhalten, wächst ihre Anzahl bei Domino Juvavum „in etwa“so schnell wie Potenzen von $\varphi \approx 1.618$.

6 Symmetrien und Normalformen

Möchte man ein Juvavum-Spiel vollständig untersuchen, muss man alle seine Spielpositionen betrachten und ihnen Grundywerte zuweisen. Wir haben im letzten Abschnitt gesehen, dass die Anzahl dieser Positionen bei wachsender Spielfeldgröße exponentiell ansteigt. Wir suchen daher nach Möglichkeiten, dieses Wachstum so weit wie möglich zu bremsen.

6.1 Symmetrien

Definition 6.1 (Symmetrien bei quadratischen Spielfeldern) Seien A und B Spielpositionen eines $n \times n$-Bretts. A heißt symmetrisch zu B, wenn A aus B durch Anwenden endlich vieler der folgenden Operationen hervorgeht:

- Drehen des Spiefelds um 90 Grad (im Uhrzeigersinn)
- horizontales Spiegeln des Spielfelds an der Mitte[41]

Schreibweise: $A \equiv B$.

Lemma 6.2 Für quadratische Spielfelder gilt: A ist symmetrisch zu B genau dann, wenn es aus B durch genau eine der folgenden Operationen entsteht:

- Drehen um 0 Grad (Position bleibt unverändert)
- Drehen um 90 Grad (im Uhrzeigersinn)
- Drehen um 180 Grad
- Drehen um 270 Grad (im Uhrzeigersinn)
- horizontales Spiegeln an der Mitte
- vertikales Spiegeln an der Mitte
- Spiegeln an der Hauptdiagonale
- Spiegeln an der Nebendiagonale

Das bedeutet, dass es zu jeder Spielposition 8 symmetrische Positionen gibt. Diese müssen nicht alle verschieden sein.

Beweis (durch Fallunterscheidung): Wir zeigen für jede Beweisrichtung beispielhaft einen Fall. Alle weiteren Fälle lassen sich analog beweisen.

Jede der 8 Operationen kann durch mehrmaliges Anwenden der beiden in der Definition angegeben Grundoperationen (Drehen um 90 Grad und horizontales Spiegeln) erreicht

[41] Mit *an der Mitte* ist hier gemeint, dass im Falle einer ungeraden Anzahl von Zeilen an der mittleren Zeile, im Fall einer geraden Anzahl von Zeilen an der Kante zwischen den beiden mittleren Zeilen gespiegelt wird. Siehe dazu auch die Abbildungen 26 sowie 27.

werden. Zum Beispiel entspricht Spiegeln an der Hauptdiagonale dreimaligem Drehen um 90 Grad mit anschließendem horizontalem Spiegeln.

Umgekehrt kann man zeigen, dass jede weitere Anwendung einer Drehung um 90 Grad oder einer horizontalen Spiegelung immer zu einer dieser 8 Positionen führt. Als Beispiel betrachten wir ein um 90 Grad gedrehtes Spielfeld: Eine weitere Drehung um 90 Grad führt ebenso wie horizontales Spiegeln zu bereits bekannten Positionen, nämlich jenen, die aus der ursprünglichen Position durch Drehen um 180 Grad bzw. vertikales Spiegeln entstehen. □

Definition 6.3 (Symmetrie bei nichtquadratischen Spielfeldern) Seien A und B Spielpositionen eines $m \times n$-Bretts mit $m \neq n$. A heißt symmetrisch zu B, wenn A aus B durch Anwenden endlich vieler der folgenden Operationen hervorgeht:

- horizontales Spiegeln an der Mitte
- vertikales Spiegeln an der Mitte[42]

Schreibweise: $A \equiv B$.

Lemma 6.4 Für nichtquadratische Spielfelder gilt: A ist symmetrisch zu B genau dann, wenn es aus B durch genau eine der folgenden Operationen hervorgeht:

- Drehen um 0 Grad (Position bleibt unverändert)
- Drehen um 180 Grad
- horizontales Spiegeln an der Mitte
- vertikales Spiegeln an der Mitte

Das bedeutet, dass es zu jeder Spielposition 4 symmetrische Positionen gibt. Diese müssen nicht alle verschieden sein.

Beweis: Der Beweis lässt sich analog zum Fall eines quadratischen Spielfelds führen. Drehen um 180 Grad entspricht dem hintereinander Ausführen von horizontalem und vertikalem Spiegeln. Zweimaliges Drehen um 180 Grad führt auf die ursprüngliche Position zurück. Die beiden anderen Fälle folgen sofort aus der Definition. Umgekehrt gibt es nicht mehr als diese 4 zueinander symmetrischen Positionen, da jede weitere Anwendung einer Drehung oder Spiegelung wieder auf eine dieser Positionen führt. Die Details dieser Überlegungen sind einfach (Beweis durch Fallunterscheidung) und werden deshalb ausgelassen. □

[42] Analog zum Fall des horizontalen Spiegelns hängt es von der Anzahl der Spalten ab, ob an der mittleren Spalte oder an einer Kante zwischen den beiden mittleren Spalten gespiegelt wird. Vgl. Abbildungen 28 sowie 29.

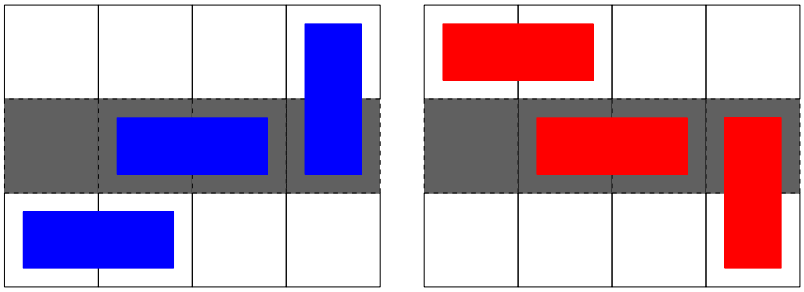


Abbildung 26: Horizontales Spiegeln (Anzahl der Zeilen ungerade)

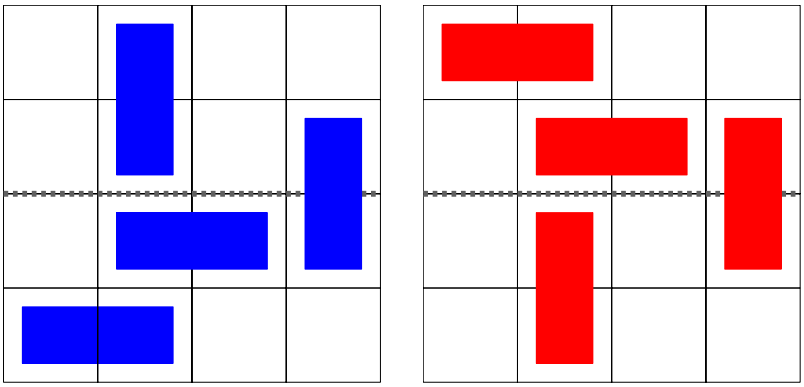


Abbildung 27: Horizontales Spiegeln (Anzahl der Zeilen gerade)

Bemerkung 6.5 Man kann die zueinander symmetrischen Spielpositionen eines quadratischen Spielfelds auch als Deckabbildungen eines Quadrats in der Ebene mit Mittelpunkt $M = (0, 0)$ betrachten. Bezeichne a die Drehung um 90 Grad, b die Spiegelung des Quadrats um die x-Achse und e die Identität. Dann ist

$$G = \{e, a, a^2, a^3, b, a \circ b, a^2 \circ b, a^3 \circ b\}$$

die Menge der Deckabbildungen dieses Quadrats. G bildet eine nichtabelsche Gruppe und wird als Diedergruppe der Ordnung 8 oder D_4 bezeichnet. Man beachte, dass a^2 einer Drehung um 180 und a^3 einer Drehung um 270 Grad entspricht. $a \circ b$ entspricht einer Spiegelung an der Nebendiagonalen, $a^3 \circ b$ einer an der Hauptdiagonalen. $a^2 \circ b$ schließlich entspricht vertikalem Spiegeln.

Eine analoge Betrachtung ist auch für den Fall eines nichtquadratischen Spielfeldes mit Mittelpunkt $M = (0, 0)$ möglich. Sei in diesem Fall a die Spiegelung um die x- und b die Spiegelung um die y-Achse. Dann bildet

$$H = \{e, a, b, a \circ b\}$$

ebenfalls eine Gruppe, die Diedergruppe D_2 der Ordnung 4. $a \circ b$ entspricht hier einer Drehung um 180 Grad.

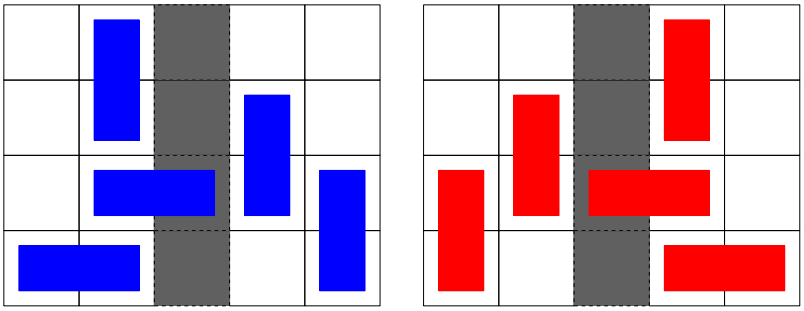


Abbildung 28: Vertikales Spiegeln (Anzahl der Spalten ungerade)

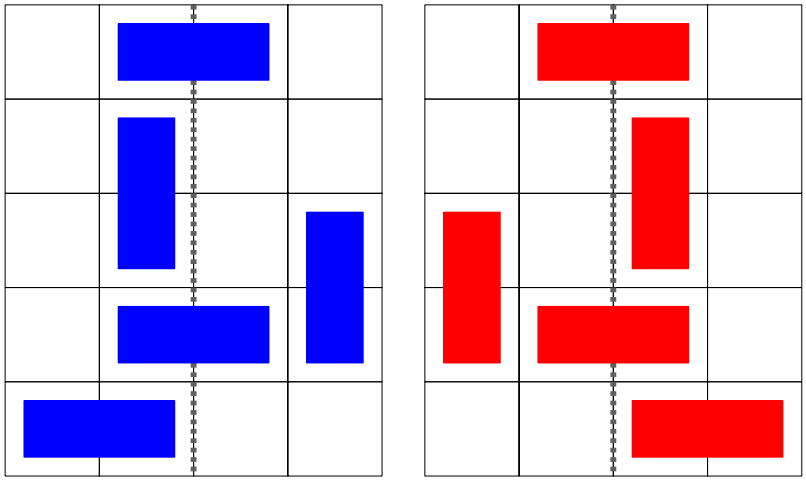


Abbildung 29: Vertikales Spiegeln (Anzahl der Spalten gerade)

Satz 6.6 Die Relation $\equiv$ ist eine Äquivalenzrelation. Das bedeutet, dass die zu einer Position A symmetrischen Spielpositionen eine Äquivalenzklasse bilden.

Beweis: Seien A, B und C Spielpositionen. Zu zeigen sind Reflexivität, Symmetrie und Transitivität:

- $A \equiv A \quad \forall A$
- $A \equiv B \Leftrightarrow B \equiv A \quad \forall A, B$
- $A \equiv B \wedge B \equiv C \Rightarrow A \equiv C \quad \forall A, B, C$

Die Reflexivität (jede Position ist zu sich selbst symmetrisch) haben wir bereits in den Lemmata gezeigt (Drehung um 0 Grad).[43]

Die Transitivität ist ebenfalls leicht einzusehen: Nach Voraussetzung ist $A \equiv B$ und $B \equiv C$. Es gibt daher nach Definition n Operationen $o_1, \ldots, o_n$, die A in B überführen

[43] Alternativ könnte man diese nahe liegende Eigenschaft auch schon per definitionem fordern.

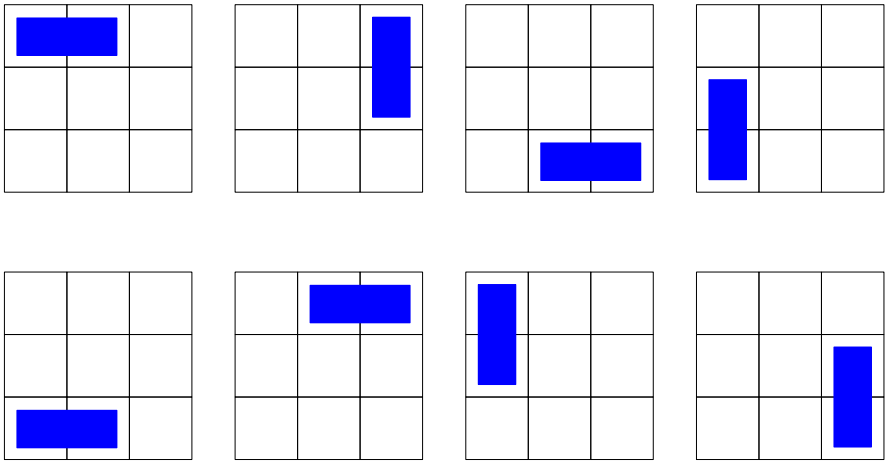


Abbildung 30: 8 zueinander symmetrische 3×3-Positionen bei Domino Juvavum

und m Operationen $o_{n+1}, \ldots, o_{n+m}$, die B in C überführen. Es folgt, dass die $n+m$ Operationen $o_1, \ldots, o_{n+m}$ von A nach C führen und daher $A \equiv C$.

Für die Symmetrie ist zu zeigen, dass es für jede der 4 (im nichtquadratischen) bzw. 8 (im quadratischen Fall) Operationen eine (Umkehr-)Operation gibt, die zur ursprünglichen Position zurückführt. Dies entspricht der Existenz eines inversen Elements, die durch die Gruppeneigenschaft (vgl. Kommentar 6.5) gesichert ist. Man kann die Umkehroperationen aber auch explizit angeben: Die Umkehrung einer Spiegelung ist die Spiegelung selbst (zweimaliges Spiegeln führt zur ursprünglichen Position). Die Umkehrung einer Drehung um 90 Grad ist eine Drehung um 270 Grad und umgekehrt. Das inverse Element zur Drehung um 180 Grad ist schließlich die Drehung um 180 Grad selbst. □

Ein gutes Computer-Analyseprogramm sollte über eine Möglichkeit verfügen, nicht alle Spielpositionen, sondern lediglich alle Äquivalenzklassen von zueinander symmetrischen Positionen zu betrachten. Es ist klar, dass die Zahl der Spielpositionen auch dann exponentiell wächst[44], dennoch lässt sich die Zahl der zu untersuchenden Positionen deutlich reduzieren.

Definition 6.7 Wir bezeichnen mit $poss(m,n)$ bzw. $dposs(m,n)$ die Anzahl der möglichen Spielpositionen bei $\mathcal{JUV}(m,n)$ bzw. $\mathcal{DJUV}(m,n)$, wobei zueinander symmetrische Positionen nur einmal gezählt werden.

Bemerkung 6.8 Da die Spielpositionen bei Cram die gleichen sind wie bei Domino-Juvavum, führen wir für deren Anzahl keine eigene Bezeichnung ein.

Natürlich interessieren wir uns nun für eine konkrete Darstellung von $poss(m,n)$ und $dposs(m,n)$ sowie für das Verhältnis $\frac{poss(m,n)}{pos(m,n)}$ bzw. $\frac{dposs(m,n)}{dpos(m,n)}$, das Aufschluss darüber gibt,

[44] Mit einer anderen Methode lässt sich die Anzahl der zu untersuchenden Positionen bei einfachem Juvavum noch weiter reduzieren. In manchen Fällen kann sogar lineares Wachstum erreicht werden. Dies wird im nächsten Abschnitt erläutert.

wie viele Positionen wir durch das Betrachten von Symmetrien einsparen können. Wir untersuchen zunächst den Fall $m = 1$.

Auf einem $1 \times n$-Feld kann man zu jeder Spielposition genau eine dazu symmetrische Position betrachten. Diese entsteht durch Drehung um 180 Grad. Eine zweite Möglichkeit wäre es, das Spielfeld vertikal an der Mitte zu spiegeln[45], was in diesem Fall allerdings stets das gleiche Resultat liefert. Intuitiv ist klar, dass diese Vorgehensweise die Anzahl der Spielpositionen auf circa die Hälfte reduzieren wird.

Satz 6.9

$$poss(1,n) = \left\{ \begin{array}{ll} 2^{n-1} + 2^{\frac{n}{2}-1} & \text{für } n \text{ gerade} \\ 2^{n-1} + 2^{\frac{n-1}{2}} & \text{für } n \text{ ungerade} \end{array} \right\} = 2^{n-1} + 2^{\lfloor \frac{n-1}{2} \rfloor}$$

$$dposs(1,n) = \left\{ \begin{array}{ll} \frac{1}{2}\left(F_{n+1} + F_{\frac{n+4}{2}}\right) & \text{für } n \text{ gerade} \\ \frac{1}{2}\left(F_{n+1} + F_{\frac{n+1}{2}}\right) & \text{für } n \text{ ungerade} \end{array} \right\} = \frac{1}{2}\left(F_{n+1} + F_{\lfloor \frac{n+3+(-1)^n}{2} \rfloor}\right)$$

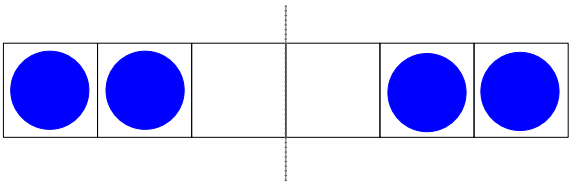


Abbildung 31: Beispiel einer $1 \times n$-Position, die sich bei vertikalem Spiegeln nicht verändert

Beweis: Wir betrachten zunächst einfaches Juvavum. Sei n gerade. Wir überlegen uns, welche der 2^n Spielpositionen auf einem $1 \times n$-Feld sich bei vertikalem Spiegeln nicht verändern. Es sind dies genau jene Positionen, bei denen die ersten $\frac{n}{2}$ Felder die gleiche Belegung haben wie die zweiten $\frac{n}{2}$, jedoch in umgekehrter Reihenfolge (für ein Beispiel siehe Abbildung 31). Davon gibt es $2^{\frac{n}{2}}$. Zu jeder der übrigen $2^n - 2^{\frac{n}{2}}$ Positionen gibt es genau eine symmetrische Position, die durch Spiegeln am Symmetriezentrum entsteht. Es folgt:

$$poss(1,n) = 2^{\frac{n}{2}} + \frac{1}{2}\left(2^n - 2^{\frac{n}{2}}\right) = 2^{n-1} + 2^{\frac{n}{2}-1}.$$

Sei n ungerade. Hier bleiben all jene Positionen bei Spiegelung unverändert, bei denen die ersten $\frac{n-1}{2}$ Felder die gleiche Belegung aufweisen wie die letzten $\frac{n-1}{2}$ in umgekehrter Reihenfolge und das mittlere Feld beliebig belegt ist. Für das mittlere Feld gibt es 2

[45] Das Symmetriezentrum liegt für gerades n zwischen den beiden Feldern in der Mitte, für ungerades n ist das mittlere Feld das Symmetriezentrum.

mögliche Belegungen (besetzt oder frei). Es sind dies daher $2 \cdot 2^{\frac{n-1}{2}} = 2^{\frac{n+1}{2}}$ Positionen. Wie im vorigen Fall folgt:

$$poss(1,n) = 2^{\frac{n+1}{2}} + \frac{1}{2}\left(2^n - 2^{\frac{n+1}{2}}\right) = 2^{n-1} + 2^{\frac{n-1}{2}}.$$

Der Beweis für Domino Juvavum verläuft im Wesentlichen analog. Sei n ungerade. Bei vertikalem Spiegeln verändern sich genau jene Positionen nicht, bei denen das mittlere Feld leer ist und die ersten $\frac{n-1}{2}$ Felder die gleiche Belegung haben wie die letzten $\frac{n-1}{2}$ in umgekehrter Reihenfolge. Davon gibt es $dpos(1, \frac{n-1}{2}) = F_{\frac{n-1}{2}}$. Es folgt analog zum Beweis für Juvavum: $dposs(1,n) = \frac{1}{2}(F_{n+1} + F_{\frac{n+1}{2}})$.

Sei n gerade. Dann müssen wir zwei Fälle unterscheiden. Für den Fall, dass ein Domino die beiden mittleren Felder bedeckt, gibt es $dpos(1, \frac{n-2}{2}) = F_{\frac{n}{2}}$ Positionen, die nicht gespiegelt werden können. Im anderen Fall (die beiden mittleren Felder werden nicht durch das gleiche Domino bedeckt) gibt es $dpos(1, \frac{n}{2}) = F_{\frac{n+2}{2}}$ solcher Positionen. Es folgt:

$$\begin{aligned} dposs(1,n) &= \frac{1}{2}(F_{n+1} + F_{\frac{n}{2}} + F_{\frac{n+2}{2}}) \\ &= \frac{1}{2}(F_{n+1} + [F_{\frac{n}{2}} + F_{\frac{n}{2}+1}]) \\ &= \frac{1}{2}(F_{n+1} + F_{\frac{n}{2}+2}) \\ &= \frac{1}{2}(F_{n+1} + F_{\frac{n+4}{2}}) \end{aligned}$$

Die Herleitung der geschlossenen Ausdrücke für allgemeines n sei dem Leser überlassen. □

Korollar 6.10

$$\lim_{n\to\infty} \frac{poss(1,n)}{pos(1,n)} = \lim_{n\to\infty} \left(\frac{1}{2} + \frac{1}{2^{\frac{n+1}{2}}}\right) = \frac{1}{2}$$

$$\lim_{n\to\infty} \frac{dposs(1,n)}{dpos(1,n)} = \lim_{n\to\infty} \left(\frac{1}{2} + \frac{1}{\varphi^{\frac{n-1}{2}}}\right) = \frac{1}{2}$$

Beweis: Die Aussagen folgen durch Einsetzen und elementare Umformungen. Im zweiten Fall verwenden wir die Formel von Binet zur Darstellung der Fibonacci-Zahlen. □

6.2 Normalform einer $\mathcal{JUV}$-Spielposition

Im Beweis von Lemma 3.8 haben wir uns bereits überlegt, dass wir bei Juvavum beliebige Zeilen- und Spaltenvertauschungen durchführen dürfen, ohne dass sich am spieltheoretischen Wert einer Position etwas verändert. Wir können diese Tatsache nun benutzen, um

die Anzahl der zu untersuchenden Positionen bei einfachem Juvavum weiter zu reduzieren. Dazu definieren wir die Normalform einer Spielposition.

Zunächst anschaulich: Um eine Juvavum-Position in Normalform zu bringen, werden sowohl die Zeilen als auch die Spalten nach der Anzahl der belegten Felder absteigend sortiert.

Definition 6.11 (Normalform einer Juvavum-Position) Sei $z(i)$ die Anzahl der belegten Felder in der i-ten Zeile und $s(j)$ die Anzahl der belegten Felder in der j-ten Spalte eines Juvavum-Spielfelds. Eine $m \times n$-Spielposition liegt in Normalform vor, wenn gilt:

1. für alle Zeilen $i, k : 0 \leq i < k \leq m \Rightarrow z(i) \geq z(k)$
2. für alle Spalten $j, k : 0 \leq j < k \leq n \Rightarrow s(j) \geq s(k)$

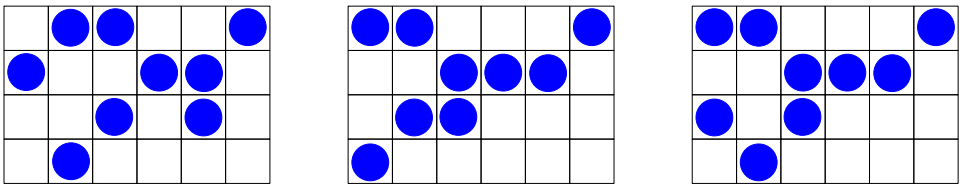


Abbildung 32: Eine 4×6-Position und zwei mögliche Normalformen

Bemerkung 6.12 Die Normaldarstellung einer Position ist im Allgemeinen nicht eindeutig (vgl. Abbildung 32), da es für Zeilen bzw. Spalten mit gleich vielen belegten Feldern kein Sortierkriterium gibt. Eine Möglichkeit wäre, zuerst nach der Anzahl der belegten Felder und dann nach der Binärcodierung einer Zeile bzw. Spalte zu sortieren. Die Binärcodierung einer Zeile ist jedoch nicht invariant gegenüber Spaltenvertauschungen. Ebenso kann sich die Codierung einer Spalte bei einer Zeilenvertauschung ändern. Die Binärcodierung einer Zeile bzw. Spalte ist daher kein gutes Sortierkriterium. Ein geeignetes Kriterium sollte ebenso wie die Anzahl der belegten Felder sowohl gegenüber Zeilen- als auch gegenüber Spaltenvertauschungen invariant sein.

Wir akzeptieren diesen Makel unserer Normalform, da sich auch ohne die Eigenschaft der Eindeutigkeit große Einsparungen erzielen lassen.

Wir betrachten erneut Juvavum auf einem $1 \times n$-Feld. Wir können die Felder einer Position so ordnen, dass die ersten k $(0 \leq k \leq n)$ Felder belegt und die restlichen $n - k$ frei sind. Es genügt also die Information über die Anzahl der belegten Felder. Deren Position spielt für die Analyse des Spiels keine Rolle.[46] Anstatt nun für jedes $0 \leq k \leq n$ alle $\binom{n}{k}$ Positionen mit k belegten Feldern untersuchen zu müssen, genügt es, jeweils nur eine Position zu untersuchen. Dies reduziert die Anzahl der Positionen von 2^n auf $n + 1$, was linearem anstelle von exponentiellem Wachstum in n entspricht.

[46] Dies entspricht auch genau den Überlegungen in Lemma 3.8.

Für allgemeine Spielfelder kann diese starke Reduktion der Anzahl der Positionen zwar nicht mehr erreicht werden, dennoch lassen sich auch hier teilweise große Einsparungen erzielen.

Wir bezeichnen die Anzahl der Normalformen, die unser JAVA-Programm (vgl. Kapitel A) zur Analyse von $\mathcal{JUV}(m,n)$ benötigt, mit $posn(m,n)$[47] und stellen sie zum Abschluss dieses Kapitels der Anzahl aller Positionen $pos(m,n)$ bzw. der Anzahl der Äquivalenzklassen von zueinander symmetrischen Positionen $poss(m,n)$ gegenüber.

m	n	$pos(m,n)$	$poss(m,n)$	$posn(m,n)$
1	1	2	2	2
1	2	4	3	3
1	3	8	6	4
1	4	16	10	5
1	5	32	20	6
1	6	64	36	7
1	7	128	72	8
1	8	256	136	9
1	9	512	272	10
2	2	16	6	6
2	3	64	24	14
2	4	256	76	33
2	5	1024	288	95
2	6	4096	1072	328
2	7	16384	4224	1266
3	3	512	102	75
3	4	4096	1120	394
3	5	32768	8640	2697
4	4	65536	8548	7760

[47] Die Vorgehensweise des Programms besteht darin, bei jeder neuen Position, die als Nachfolger einer bereits bekannten Position auftritt, zu überprüfen ob a.) die Normalform dieser Position oder b.) eine zu dieser Position symmetrische Spielposition bereits zu einem früheren Zeitpunkt im Programmablauf aufgetreten ist. In diesem Fall wird die bereits bekannte Position (und nicht die neu gefundene) als Nachfolger gespeichert. Auf Grund der Nichteindeutigkeit der Normalformen, muss dieses Vorgehen nicht unbedingt die optimale Lösung sein. Obwohl die Einsparungen groß sind, könnte man mit einer alternativen Normalform unter Umständen noch bessere Ergebnisse erzielen. Die Definition von *posn* ist daher streng genommen etwas willkürlich. Sie dient allerdings nur dazu, die Größenordnung der Einsparung darzustellen.

7 Allgemeine Spielstrategien

7.1 Die Spiegelungsstrategie

Eine wichtige Strategie, die nicht nur in allen Juvavum-Varianten, sondern auch in vielen anderen Spielen von großer Bedeutung ist, ist die Spiegelungsstrategie[48]. Dabei spiegelt ein Spieler jeden Zug seines Gegners an einem Symmetriezentrum, wodurch er den letzten Zug des Spiels machen wird. In diesem Kapitel nutzen wir diese Strategie, um für normales Juvavum, Domino Juvavum und Cram die Fälle *gerade* × *gerade* und *gerade* × *ungerade* (schwach) zu lösen.

Satz 7.1 $\mathcal{DJUV}(2m, 2n)$ kann $\forall m, n \in \mathbb{N}$ stets B gewinnen.

Beweis: Bei $\mathcal{DJUV}(2m, 2n)$ liegt das Symmetriezentrum zwischen den vier Feldern in der Mitte des Spielbretts. B kann jeden Zug von A an diesem Symmetriezentrum spiegeln und mit dieser Spielweise immer den letzten Zug machen. □

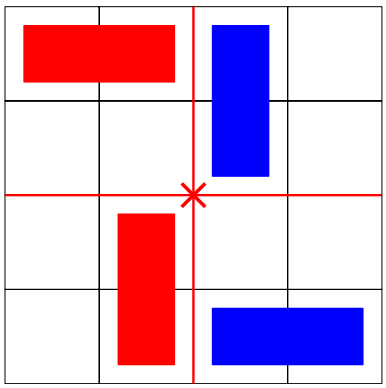


Abbildung 33: Symmetrie auf $2n \times 2m$-Feldern

Satz 7.2 $\mathcal{DJUV}(2m, 2n + 1)$ kann $\forall m, n \in \mathbb{N}$ stets A gewinnen.

Beweis: Bei $\mathcal{DJUV}(2m, 2n + 1)$ liegt das Symmetriezentrum auf einer Kante (siehe Abbildung 34). Belegt A jene Spalte komplett, in der das Symmetriezentrum liegt (das ist stets möglich, da die Anzahl der Zeilen gerade ist), kann er im Folgenden alle Züge von B am Symmetriezentrum spiegeln und so gewinnen. Die einzigen Zugmöglichkeiten, die sich nicht spiegeln lassen, sind durch den ersten Zug von A bereits eliminiert worden. □

Bemerkung 7.3 Durch Drehen um 90 Grad gilt dieses Ergebnis natürlich auch für $\mathcal{DJUV}(2m + 1, 2n)$.

[48] Diese Strategie wird in [BCG82] auch als Tweedledum & Tweedledee-Stratgie bezeichnet.

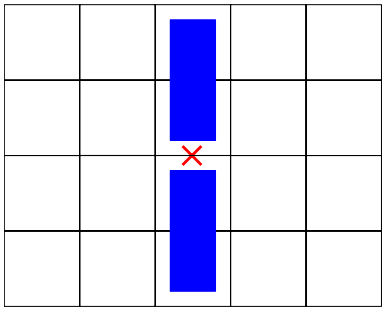


Abbildung 34: Symmetrie auf $2n \times (2m+1)$-Feldern

Analog gilt für Juvavum und Cram:

Satz 7.4 $\mathcal{JUV}(2m, 2n)$ kann $\forall m, n \in \mathbb{N}$ stets B gewinnen.
Beweis: Analog zu Satz (7.1). □

Satz 7.5 $\mathcal{JUV}(2m, 2n+1)$ und $\mathcal{JUV}(2m+1, 2n)$ kann $\forall m, n \in \mathbb{N}$ stets A gewinnen.
Beweis: Analog zu Satz (7.2). □

Satz 7.6 $\mathcal{CRAM}(2m, 2n)$ kann $\forall m, n \in \mathbb{N}$ stets B gewinnen.
Beweis: Analog zu Satz (7.1).[49] □

Satz 7.7 $\mathcal{CRAM}(2m, 2n+1)$ und $\mathcal{CRAM}(2m+1, 2n)$ kann $\forall m, n \in \mathbb{N}$ stets A gewinnen.
Beweis: A legt das erste Domino in die Mitte jener Zeile bzw. Spalte, in der das Symmetriezentrum liegt und kann fortan alle Züge von B an diesem Symmetriezentrum spiegeln.[50] □

Bemerkung 7.8 Das Wissen um diese Spielstrategien macht das Spielen auf $2m \times 2n$ bzw. $2m \times (2n+1)$-Spielbrettern schnell uninteressant. Anstatt sich völlig auf den Fall $(2m+1) \times (2n+1)$ zu konzentrieren, können diese Spiele jedoch durch Zusatzvorschriften wieder spannend gestaltet werden. Eine Möglichkeit besteht darin, im Fall $2m \times 2n$ dem 2. Spieler das Spiegeln des ersten Zuges nicht zu erlauben. Für jeden ersten Zug von A wird also genau eine Zugmöglichkeit für B aus dem Spielegraphen entfernt. Genauso wird im Fall $2m \times (2n+1)$ der in den obigen Beweisen beschriebene erste Zug für A nicht zugelassen. Eine zweite Möglichkeit besteht darin, für die Fälle *gerade* × *gerade* und *gerade* × *ungerade* nur Misère-Spiele zu betrachten, da sich die Spiegelungsstragie in diesem Fall nicht anwenden lässt.[51]

[49] Vgl. auch [Gar74]. [50] Vgl. auch [Gar74]. [51] Beide Varianten werden unter anderem von Gardner in [Gar74] vorgeschlagen.

7.2 Zerlegung in unabhängige Teilfelder

Wir haben in Kapitel 2 den Hauptsatz über Summenspiele bewiesen. Wir wollen nun zeigen, wie dieser bei der Bewertung mancher Juvavum-Positionen von Nutzen sein kann.

Definition 7.9 (Teilfeld) Sei $X := \{x_{ij}, i \leq n, j \leq m\}$ die Menge aller Felder einer $m \times n$-Spielposition. Sei weiters $x_{ij} = 1$, wenn das i-te Feld der j-ten Spalte belegt ist. Andernfalls sei $x_{ij} = 0$. Eine Menge T heißt Teilfeld von X, wenn gilt:

- T ist eine Teilmenge von X: $T \subseteq X$.
- Alle Felder von T sind frei: $x_{ij} = 0 \quad \forall i, j$ mit $x_{ij} \in T$.
- Der durch T repräsentierte Teil des Spielfelds ist horizontal und vertikal (links, rechts, oben und unten nicht notwendigerweise aber auch diagonal) durch belegte Felder oder den Spielfeldrand begrenzt.

Wir können nun alle Juvavum-Spiele auch auf Teilfeldern betrachten:

Definition 7.10 Sei T ein Teilfeld eines rechteckigen Spielfelds, dann bezeichnet $\mathcal{JUV}(T)$ das Spiel Juvavum[52] auf diesem Teilfeld.

Definition 7.11 (Unabhängigkeit von Teilfeldern) Zwei Teilfelder T_1 und T_2 einer Position x heißen unabhängig bezüglich S, wenn ein Zug (nach den Regeln von S) stets nur in einem der beiden Teilfelder möglich ist, das heißt, wenn es keinen legalen Zug gibt, der sowohl Felder von T_1 als auch Felder von T_2 belegt.

Korollar 7.12 Sei x eine Spielposition, die bzgl. eines Spiels S in k unabhängige Teilfelder $T_1, \ldots, T_k$ zerfällt. Dann gilt:

$$S(x) = \sum_{i=1}^{k} S(T_i).$$

Beweis: Diese Aussage folgt sofort aus der Definition der Unabhängigkeit bzw. der Definition von Summenspielen. □

Korollar 7.13 Sei x wie oben. Für den Grundywert (bzgl. des Spiels S) gilt:

$$g_S(x) = g_S(T_1) \dot{+} \ldots \dot{+} g_S(T_k).$$

Beweis: Diese Aussage folgt aus dem Hauptsatz über Summenspiele. □

Bemerkung 7.14 Die Einschränkung *bzgl. eines Spiels S* ist wichtig. Abbildung 35 zeigt ein Spielfeld, das in 3 Teilfelder zerfällt. Bzgl. Cram sind diese unabhängig, bzgl. Domino Juvavum allerdings nicht, wie die beiden eingezeichneten Zugmöglichkeiten verdeutlichen sollen.

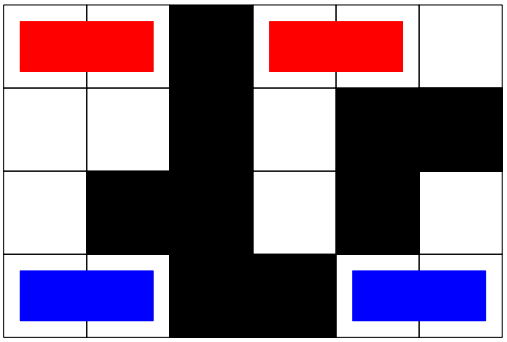


Abbildung 35: Beispiel für eine Position mit 3 Teilfeldern

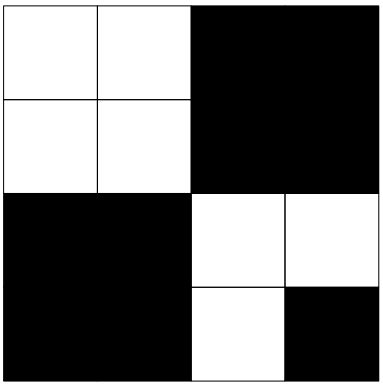


Abbildung 36: Beispiel für eine Juvavum-Position mit 2 unabhängigen Teilfeldern

Bemerkung 7.15 Bei einfachem Juvavum wird die Analyse von Teilfeldern nur selten zum Erfolg führen, da nur in wenigen Fällen unabhängige Teilfelder auftreten (für ein Beispiel vgl. Abbildung 36).

Bei Domino Juvavum und insbesondere bei Cram zerfällt ein Spielfeld oft schon sehr bald in einige wenige unabhängige Teilfelder. Ein (menschlicher) Spieler, der über die Grundywerte der am häufigsten auftretenden Teilfelder Bescheid weiß (und gute Kopfrechenfähigkeiten besitzt) oder zumindest ein Gefühl dafür entwickelt, welche Kombinationen von Teilfeldern günstig sind (d. h. einer Gewinnposition entsprechen), wird daher bei diesen beiden Spielen einen deutlichen Vorteil haben.

Wir führen nun für einige häufig vorkommende Teilfelder Bezeichnungen ein.

Definition 7.16 Wir definieren die Figuren $I(x)$, $L(x,y)$, $T(x,y,z)$, $U(x,y,z)$ und $O(x,y)$. Diese haben jeweils die in Abbildung 37 dargestellte Form.

Lemma 7.17

$$\mathcal{JUV}(I(x)) = \mathcal{JUV}(1,x)$$

[52] analog für alle anderen Juvavum-Varianten

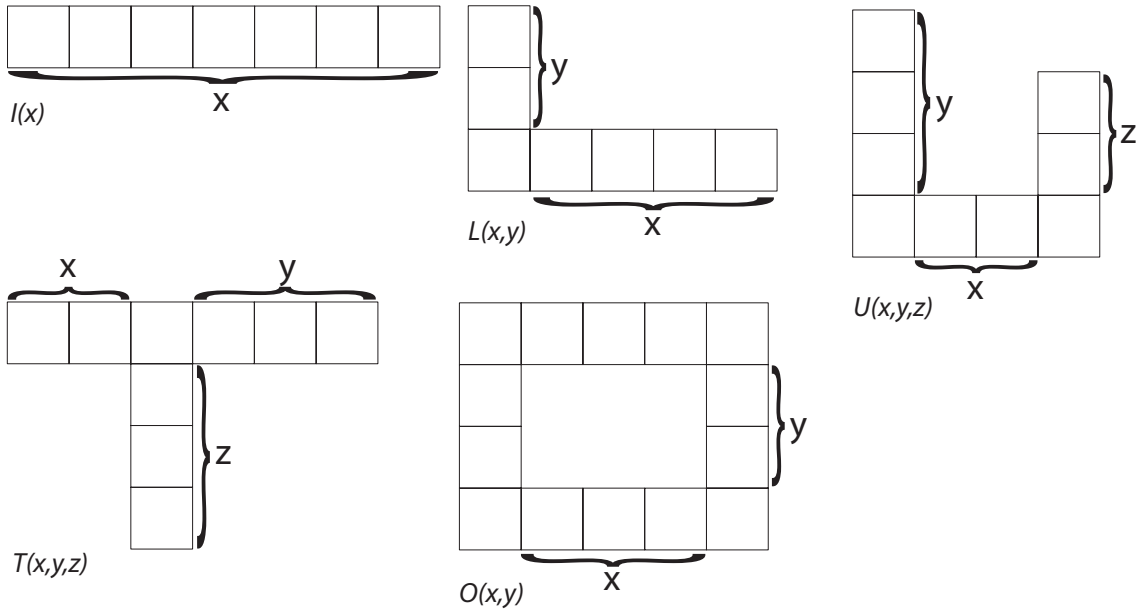


Abbildung 37: Ausgewählte Teilfelder von $m \times n$-Spielbrettern

Die Aussage gilt analog auch für alle anderen Juvavum-Varianten.
Beweis: Die Aussage ist trivial, da die Figur $I(x)$ gerade einem $1 \times x$-Feld entspricht.

Für kleine Werte macht es durchaus Sinn, die Grundywerte dieser Figuren zu tabellieren. Wir geben eine solche Tabelle beispielhaft für einige L-Figuren an. Die Grundywerte wurden mit Hilfe des Computerprogramms gewonnen, das in Kapitel A vorgestellt wird.

Beispiel 7.18 Wir betrachten die in Abbildung 39 gezeigte $\mathcal{DJUV}$-Position x auf einem 4×6-Spielbrett. Sie zerfällt in 3 paarweise unabhängige Teilfelder und es gilt:

$$x = L(1,3) + L(2,2) + I(2).$$

n	1	2	3	4	5	6
$g_{\mathcal{DJUV}}(L(1,n))$	1	2	2	3	3	4
$g_{\mathcal{DJUV}}(L(2,n))$	2	0	3	3	4	4
$g_{\mathcal{DJUV}}(L(3,n))$	2	3	3	4	4	5
$g_{\mathcal{DJUV}}(L(4,n))$	3	3	4	0	1	1
$g_{\mathcal{DJUV}}(L(5,n))$	3	4	4	1	1	6
$g_{\mathcal{DJUV}}(L(6,n))$	4	4	5	1	6	0

Abbildung 38: $\mathcal{DJUV}$-Grundywerte für L-förmige Teilfelder

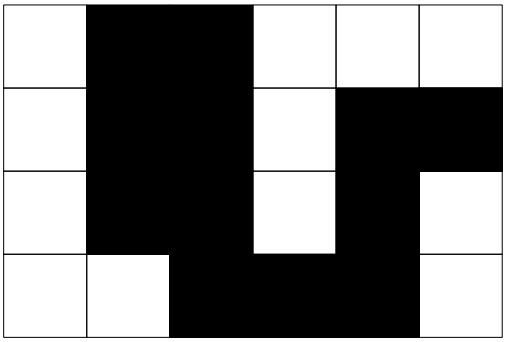


Abbildung 39: Beispiel für eine $\mathcal{DJUV}$-Position mit 3 unabhängigen Teilfeldern

Weiters folgt für den Grundywert der Spielposition:

$$g_{\mathcal{DJUV}}(x) = g_{\mathcal{DJUV}}(L(1,3)) \dot{+} g_{\mathcal{DJUV}}(L(2,2)) \dot{+} g_{\mathcal{DJUV}}(I(2)).$$

Wir entnehmen die Grundywerte der beiden L-Figuren aus der obigen Tabelle. Weiters stellen wir fest, dass $g_{DJUV}(I(2)) = 1$. Dies ist trivial, da es im Domino Juvavum-Spiel auf einem 1×2-Feld nur zwei mögliche Positionen gibt. Die Zielecke (beide Felder belegt) hat per Definition Grundywert 0, die Startecke (beide Felder leer) folglich Grundywert 1 (da sie die Zielecke als einzigen Nachfolger hat).

Es folgt:

$$g_{\mathcal{DJUV}}(x) = 2 \dot{+} 0 \dot{+} 1 = 3.$$

Ein guter erster Zug besteht darin, ein Domino (beliebig) in das $L(1,3)$-Teilfeld zu setzen. Dadurch entsteht je nach Lage dieses Dominos das Summenspiel $L(1,1)+L(2,2)+I(2)$ bzw. das Spiel $I(1)+I(2)+L(2,2)+I(2)$. Beide haben Grundywert 0. Jeder andere Zug führt auf eine Position mit Grundywert > 0.

8 Kleine Spielfelder: Die eindimensionale Version von $\mathcal{JUV}$, $\mathcal{DJUV}$ und $\mathcal{CRAM}$

Definition 8.1 (Lineares Juvavum) Wir nennen $\mathcal{JUV}(m,n)$ linear, wenn $m = 1$. Diese Bezeichnung gelte analog auch für $\mathcal{JUVM}$, $\mathcal{DJUV}$, $\mathcal{DJUVM}$, $\mathcal{CRAM}$ und $\mathcal{CRAMM}$.

Bemerkung 8.2 (Schreibweise) In diesem Kapitel schreiben wir $g(n)$ für den Grundywert von $\mathcal{JUV}(1 \times n)$ bzw. $gm(n)$ für den Grundywert von $\mathcal{JUVM}(1 \times n)$. Für Domino Juvavum und Cram definieren wir analog die Bezeichnungen $dg(n)$, $dgm(n)$, $cg(n)$ sowie $cgm(n)$.

8.1 Analyse von $\mathcal{CRAM}(1,n)$

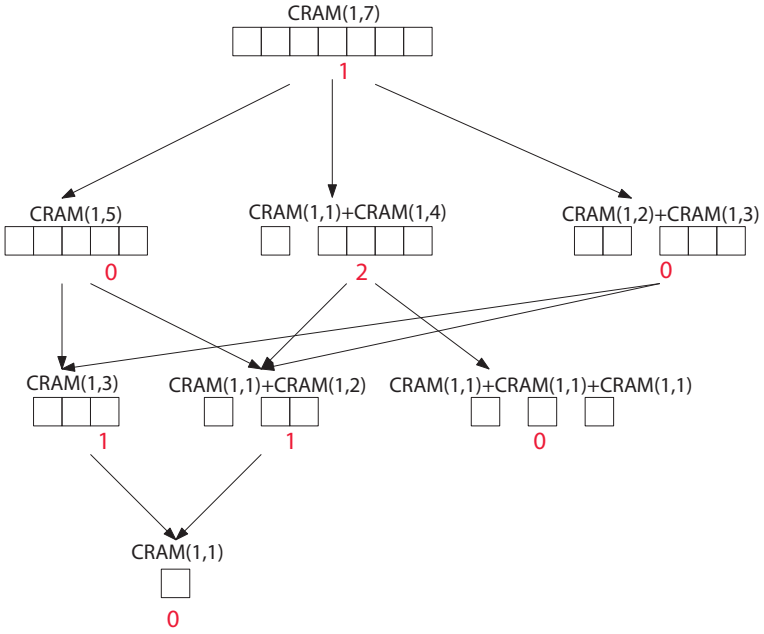


Abbildung 40: Spielegraph für $\mathcal{CRAM}(1,7)$

Martin Gardner erwähnt in [Gar74], dass David Singmaster 1973 mit Hilfe eines Computerprogramms $\mathcal{CRAM}(1,n)$ für $n \leq 1000$ untersucht und dabei 151 Spiele mit Grundywert 0 gefunden hat. Wir werden in Kapitel A.1 ein einfaches JAVA-Programm beschreiben, das diese Ergebnisse auf einem modernen PC in unter einer Sekunde bestätigt und zudem zur Berechnung weiterer Werte herangezogen werden kann.[53] Darüber hinaus ist es eine einfache und dennoch sehr effektive Anwendung des Satzes über Summenspiele (Satz 2.48) zur rekursiven Berechnung von Grundywerten.

[53] Für alle $n \leq 10^6$ dauert diese Berechnung auf einem modernen PC nur wenige Stunden.

Bei $\mathcal{CRAM}(1,n)$ unterteilt jeder erste Spielzug das Spielfeld in ein oder zwei Felder. Dadurch entsteht eine Situation, die als Summe zweier kleinerer Spiele angegeben werden kann.

Für $n = 5$ hat die Startposition beispielsweise 2 Nachfolger (Spiegelung des Spielbretts berücksichtigt): Entweder setzt man das Domino an den Rand des Spielbretts, dann bleibt ein $\mathcal{CRAM}(1,3)$-Spiel übrig, oder man lässt auf beiden Seiten des Dominos Felder frei (ein Feld auf der einen, zwei auf der anderen Seite), was das Spiel auf die Summe von $\mathcal{CRAM}(1,1)$ und $\mathcal{CRAM}(1,2)$ reduziert. Allgemein hat jeder Nachfolger der Startposition von $\mathcal{CRAM}(1,n)$ die Form

$$\mathcal{CRAM}(1,i) + \mathcal{CRAM}(1,n-2-i)$$

mit $i \in \mathbb{N}_0$.

Man sieht leicht, dass es bei $\mathcal{CRAM}(1,n)$ $\lfloor \frac{n}{2} \rfloor$ mögliche Positionen für das erste Domino gibt.[54]

Der Grundywert von $\mathcal{CRAM}(1,n)$ ist durch die Grundywerte dieser Nachfolger festgelegt und kann rekursiv bestimmt werden.

$$\begin{aligned} g(\mathcal{CRAM}(1,n)) &= mex\{g(\mathcal{CRAM}(1,i)) \dot{+} g(\mathcal{CRAM}(1,n-2-i)), \quad 0 \le i \le n-2\} \\ &= mex\{g(\mathcal{CRAM}(1,i)) \dot{+} g(\mathcal{CRAM}(1,n-2-i)), \quad 0 \le i \le \lfloor \frac{n}{2} \rfloor\} \end{aligned}$$

Die Tatsache, dass wir die Variable i nur im Bereich zwischen 0 und $\lfloor \frac{n}{2} \rfloor$ betrachten müssen, folgt aus der Möglichkeit, spiegelbildliche Positionen miteinander identifizieren zu können, die wir uns in Kapitel 6 überlegt haben.

Die Startwerte der Rekursion sind $g(\mathcal{CRAM}(1,0))$=$g(\mathcal{CRAM}(1,1)) = 0$, da auf diesen beiden Spielfeldern kein Zug möglich ist.

In Kapitel A wird der JAVA-Quellcode dieses Programms angegeben. Zwar können damit Grundywerte für sehr große n bestimmt werden, es zeigt sich jedoch, dass das gar nicht nötig ist, da die Folge $g(\mathcal{CRAM}(1,n))$ nach einer Vorperiode von 53 Gliedern eine Periode der Länge 34 aufweist.[55]

Dass sich diese Periode wirklich unendlich forsetzt, folgt aus Satz B.8.

n	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
$cg(n)$	0	0	1	1	2	0	3	1	1	0	3	3	2	2	4	0	5	2	2	3	3	0	1	1	3	0	2	1	1	0	4	5	2	7
$cg(n+34)$	4	0	1	1	2	0	3	1	1	0	3	3	2	2	4	4	5	5	2	3	3	0	1	1	3	0	2	1	1	0	4	5	3	7
$cg(n+68)$	4	8	1	1	2	0	3	1	1	0	3	3	2	2	4	4	5	5	9	3	3	0	1	1	3	0	2	1	1	0	4	5	3	7
$cg(n+102)$	4	8	1	1	2	0	3	1	1	0	3	3	2	2	4	4	5	5	9	3	3	0	1	1	3	0	2	1	1	0	4	5	3	7

Tabelle 2: Grundywerte für $\mathcal{CRAM}(1,n)$

Bemerkung 8.3 Spielt man die in Bemerkung 3.7 beschriebene Variante von Cram auf einem $(1 \times n)$-Feld, so erkennt man, dass $\mathcal{CRAM}(1,n) = DK(n)$.

[54] Berücksichtigt man die Spiegelung an der Mitte des Feldes nicht, gibt es $n-1$ Nachfolger. [55] Leider gibt Gardner keine Details über Singmasters Berechnungen an. Es ist interessant, dass auf die Periodizität der Grundywerte nicht eingegangen wird. Selbst wenn nur untersucht worden wäre, welcher der beiden Spieler für $n \le 1000$ eine Gewinnstrategie hat, müsste die Regelmäßigkeit aufgefallen sein: Der 2. Spieler kann stets gewinnen, wenn n von der Form $i + 34 \cdot j$ mit $i \in \{5,9,15,21,25,29\}$ und $j \ge 0$ ist sowie für $n \in \{0,1,15,35\}$.

8.2 Analyse von $\mathcal{CRAMM}(1, n)$

Die folgende Tabelle enthält Grundywerte für $\mathcal{CRAM}$ und $\mathcal{CRAMM}(1, n)$ für alle $n \leq 25$. Für weitergehende Resultate sei auf [Pla07b] verwiesen.[56]

n	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
$cg(n)$	1	1	2	0	3	1	1	0	3	3	2	2	4	0	5	2	2	3	3	0	1	1	3	0
$cgm(n)$	0	0	2	1	3	0	0	1	1	3	0	2	1	1	0	0	2	1	3	0	0	1	1	3

Tabelle 3: Grundywerte für lineares Cram und Misère-Cram

8.3 Analyse von $\mathcal{DJUV}(1, n)$

Satz 8.4 $\mathcal{DJUV}(1, n)$ kann $\forall n$ stets A gewinnen.

Beweis: A belegt bei geradem n alle Felder, bei ungeradem alle bis auf eines. Damit hat er gewonnen, da B keine Zugmöglichkeit mehr hat. □

Satz 8.5 $\mathcal{DJUV}(1, n)$ hat $\forall n \geq 2$ Grundywert $\lfloor \frac{n}{2} \rfloor$.

Beweis: Wir betrachten einen möglichen Spielverlauf (in Abbildung 41 dargestellt). In diesem Fall werden bis zum Spielende $\lfloor \frac{n}{2} \rfloor$ Dominos gesetzt. Weiters gilt: Die Zielposition mit $\lfloor \frac{n}{2} \rfloor$ Dominos ist von jeder anderen im Spielverlauf aufgetretenen Position aus auch in einem einzigen Zug erreichbar. Für jene Spielposition aus unserem vorgegebenen Spielverlauf mit $\lfloor \frac{n}{2} \rfloor - 1$ Dominos folgt, dass ihr Grundywert gleich 1 sein muss, da sie nur Zielpositionen (Grundywert 0) als Nachfolger hat.

Die Position mit $\lfloor \frac{n}{2} \rfloor - 2$ Dominos hat Nachfolger mit Grundywerten 0 und 1 und daher Grundywert 2. Allgemein gilt: Es gibt eine Position mit $\lfloor \frac{n}{2} \rfloor - k$ Dominos mit $0 \leq k \leq \lfloor \frac{n}{2} \rfloor$ und k ganz, die zumindest je einen Nachfolger mit Grundywert $0, 1, \ldots, k-1$ hat. Damit folgt: Diese Position hat Grundywert k. Für $k = \lfloor \frac{n}{2} \rfloor$ ist die einzig mögliche Position die Startposition. Daraus folgt die Behauptung. □

8.4 Analyse von $\mathcal{DJUVM}(1, n)$

Satz 8.6 $\mathcal{DJUVM}(1, n)$ kann A $\forall n \geq 4$ stets gewinnen.

Beweis: Bei geradem n zieht A so, dass 2 benachbarte Felder frei bleiben, bei ungeradem n lässt er 3 (davon zumindest 2 benachbarte) frei. So muss B stets den letzten Zug machen. □

[56] Dort befindet sich ein Archiv mit Lösungen diverser Misère-Spiele. Unter der oktalen Darstellung 0.07 findet man einige Ergebnisse für eindimensionales Cram (bzw. Dawson's Kayles).

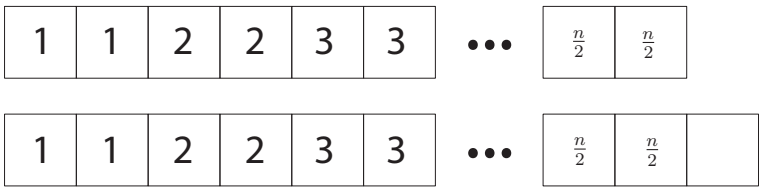


Abbildung 41: Längster möglicher Spielverlauf bei $\mathcal{DJUV}(1,n)$ für gerades und ungerades n

Satz 8.7 $\mathcal{DJUVM}(1,n)$ hat $\forall n \geq 4$ Grundywert $\lfloor \frac{n}{2} \rfloor$. Für $n = 2$ und $n = 3$ ist der Grundywert 0.

Beweis: Der Beweis lässt sich analog zum Beweis derselben Aussage für $\mathcal{DJUV}(1,n)$ führen. Wir betrachten den Spielegraphen des normalen $1 \times n$-Spiels und setzen den Grundywert aller Endecken gleich 1 (vgl. Bemerkung 2.41). Insbesondere haben alle Positionen mit $\lfloor \frac{n}{2} \rfloor$ Dominos Grundywert 1. Wir betrachten die Position mit $\lfloor \frac{n}{2} \rfloor - 1$ Dominos aus dem in Abbildung 41 dargstellten Spiel. Sie hat nur Nachfolger mit Grundywert 1 und daher Grundywert 0. Alle anderen Positionen haben die gleichen Grundywerte wie bei $\mathcal{DJUV}(1,n)$.

Für $n = 2$ und $n = 3$ ist der Grundywert gleich 0, da nur ein Zug möglich ist. □

n	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
$dg(n)$	1	1	2	2	3	3	4	4	5	5	6	6	7	7	8	8	9	9	10	10	11	11	12	12
$dgm(n)$	0	0	2	2	3	3	4	4	5	5	6	6	7	7	8	8	9	9	10	10	11	11	12	12

Tabelle 4: Grundywerte für $\mathcal{DJUV}(1,n)$ und $\mathcal{DJUVM}(1,n)$

8.5 Analyse von $\mathcal{JUV}(1,n)$

Satz 8.8 $\mathcal{JUV}(1,n)$ kann $\forall n$ stets A gewinnen.

Beweis: Wie wir bereits gesehen haben, ist $\mathcal{JUV}(1,n)$ das Gleiche wie Nim mit einem Haufen aus n Elementen. Daraus folgt: $g(\mathcal{JUV}(1,n)) = g(*n) = n$. Es kann also stets der 1. Spieler gewinnen. □

Der Gewinnzug ist offensichtlich: A belegt im ersten Zug das komplette Spielfeld. Mit jedem anderen Zug würde er bei gutem Spiel von B verlieren, da dieser dann den kompletten $1 \times n$-Streifen belegen kann.

8.6 Analyse von $\mathcal{JUVM}(1,n)$

Satz 8.9 $\mathcal{JUVM}(1,n)$ kann $\forall n \geq 2$ stets A gewinnen.

Beweis: A lässt in seinem ersten Zug genau ein Feld frei. □

Satz 8.10 $\mathcal{JUVM}(1,n)$ hat $\forall n \geq 2$ Grundywert n. Für $n = 1$ is der Grundywert 0.
Beweis: analog zu $\mathcal{DJUVM}(1,n)$. □

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
$g(n)$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
$gm(n)$	0	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Tabelle 5: Grundywerte für $\mathcal{JUV}(1,n)$ und $\mathcal{JUVM}(1,n)$

9 Etwas größer: Einige Resultate für m $\times$ n mit $m \geq 2$

In Kapitel 7.1 haben wir für die Fälle *gerade* $\times$ *gerade* sowie *gerade* $\times$ *ungerade* aller Juvavum-Formen (im normalen Spiel) eine Gewinnstrategie angegeben. Für den Fall *gerade* $\times$ *ungerade* gibt das dabei verwendete Symmetrieargument jedoch noch keinen genauen Aufschluss über die Grundywerte dieser Spiele (wir wissen nur, dass sie alle > 0 sind). Wir werden sie daher im Folgenden für einige Fälle angeben.

Ist sowohl die Spielfeldhöhe als auch die Spielfeldbreite ungerade, lassen sich noch keine allgemeinen Ergebnisse dafür angeben, welcher Spieler eine Gewinnstrategie besitzt. Wir geben im Folgenden Grundywerte für einige dieser Spiele an.

Da das Symmetrieargument bei Misère-Spielen nicht greift, bleibt hier vieles unklar. Im Folgenden werden wir die $\mathcal{DJUVM}$-Spiele $(2, n)$, $(3, 3)$ und $(3, 2n)$(schwach) lösen sowie für einige größere Spiele Grundywerte angeben.

Wir führen einige Abkürzungen ein. $g(m, n)$ sei der Grundywert von $\mathcal{JUV}(m \times n)$, $gm(m, n)$ jener von $\mathcal{JUVM}(m \times n)$. $pos(m, n)$ gibt die Anzahl der Spielpositionen von $\mathcal{JUV}(m, n)$ an, $poss(m, n)$ jene, wenn Symmetrien berücksichtigt werden. $avgb(m, n)$ ist der durchschnittliche Verzweigungsfaktor des Spielegraphen von $\mathcal{JUV}(m, n)$, $avgbs(m, n)$ jener des Graphen, der entsteht, wenn zwischen zueinander symmetrischen Positionen nicht unterschieden wird (vgl. Kapitel 6).

Für Domino Juvavum und Cram setzen wir ein d bzw. ein c an den Anfang der jeweiligen Abkürzungen. Zum Beispiel bezeichnet $dpos(m, n)$ die Anzahl der Positionen bei $\mathcal{DJUV}(m, n)$, $cg(m, n)$ ist der Grundywert von $\mathcal{CRAM}(m, n)$.

9.1 Juvavum

Satz 9.1

$$g(\mathcal{JUV}(2m, 2n)) = 0 \quad \forall m, n \geq 1.$$

Beweis: In Kapitel 7.1 haben wir mit Hilfe eines Symmetrieargumentes gezeigt, dass bei $\mathcal{JUV}(2m, 2n)$ stets der 2. Spieler gewinnen kann. Daraus folgt sofort die Aussage. □

Vermutung 9.2 Für alle $n \geq 2$ gilt:

$$g(\mathcal{JUV}(2, n)) = \begin{cases} 0, & \text{wenn } n \text{ gerade (folgt aus Satz 9.1 mit } m = 1) \\ 1, & \text{wenn } n \text{ ungerade} \end{cases}$$

Vermutung 9.3 Für alle $n \geq 2$ gilt:

$$g(\mathcal{JUVM}(2, n)) = \begin{cases} 0, & \text{wenn } n \equiv 2 \pmod 3 \\ 1, & \text{wenn } n \equiv 1 \pmod 3 \\ 2, & \text{wenn } n \equiv 0 \pmod 3 \end{cases}$$

Beide Vermutungen ergeben sich aus den Ergebnissen des Computerprogramms, die in einer Tabelle am Ende der Arbeit angegeben sind.

Für eine detaillierte Aufstellung aller Ergebnisse für einfaches Juvavum (Grundywerte, Anzahl der Positionen, Verzweigungsfaktor) verweisen wir ebenfalls auf die Tabelle am Ende der Arbeit.

9.2 Domino Juvavum

9.2.1 $\mathcal{DJUV}(2,n)$ und $\mathcal{DJUVM}(2,n)$

Satz 9.4 $\mathcal{DJUVM}(2,2k+1)$ kann $\forall k \geq 0$ stets B gewinnen.

Beweis: Wir zeigen eine leichte Verallgemeinerung. Gegeben sei eine Spielposition von $\mathcal{DJUVM}(2,n), n \in \mathbb{N}$, in der $m := 2k+1 \quad (0 \leq k \leq \frac{n-1}{2})$ Spalten frei und die anderen belegt sind (d. h. es gibt keine Spalte, in der nur eines der beiden Felder durch ein Domino bedeckt wird). Dann gilt: Von einer solchen Position kann B stets gewinnen.

Für $k = 0$ ist die Behauptung trivial, da auf einem solchen Spielfeld nur ein einziger Zug möglich ist und B stets gewinnen wird. Angenommen, die Behauptung stimmt für alle ungeraden $m \leq 2k-1$.

Wir betrachten nun eine Spielposition von $\mathcal{DJUVM}(2,n), n$ beliebig, $n \geq 2k+1$ mit $2k+1$ freien Spalten. A ist am Zug. Egal wie er zieht, bietet sich B stets die Möglichkeit, eine Position mit $\leq 2k-1$ freien Spalten zu erreichen. Setzt A vertikal, d. h. er belegt eine weitere Spalte, besetzt B ebenfalls eine weitere Spalte, wodurch $2k-1$ Spalten frei bleiben. Zieht A horizontal, kopiert B dessen Zug in der jeweils anderen Zeile, wodurch ebenfalls eine Position mit $\leq 2k-1$ freien Spalten bleibt. Folglich kann B stets gewinnen. Die Startposition von $\mathcal{DJUVM}(2,2k+1)$ ist eine Spielposition von $\mathcal{DJUVM}(2,n)$ mit $2k+1$ freien Spalten ($n = 2k+1$). Damit ist der Satz bewiesen. □

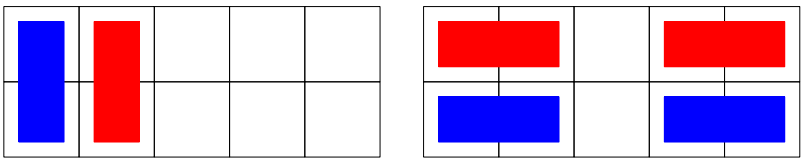


Abbildung 42: $\mathcal{DJUVM}(2,2k+1)$

Satz 9.5 $\mathcal{DJUVM}(2,2k)$ kann $\forall k \geq 0$ stets A gewinnen.

Beweis: A besetzt im ersten Zug eine beliebige Spalte und erreicht auf diese Weise eine Position, von der er nach Satz 9.4 stets gewinnen kann. □

Vermutung 9.6 Für alle $n \geq 2$ gilt:

$$g(\mathcal{DJUV}(2,n)) = \begin{cases} 0, & \text{wenn } n \text{ gerade (folgt aus Satz 9.13 mit } m=1) \\ 1, & \text{wenn } n \text{ ungerade} \end{cases}$$

Vermutung 9.7 Für alle $n \geq 2$ gilt:

$$g(\mathcal{DJUVM}(2,n)) = \begin{cases} 1, & \text{wenn } n \text{ gerade} \\ 0, & \text{wenn } n \text{ ungerade} \end{cases}$$

Die folgende Tabelle zeigt einige Resultate für $\mathcal{DJUV}(2,n)$ bzw. $\mathcal{DJUVM}(2,n)$.

n	2	3	4	5	6	7	8	9	10
$dg(2,n)$	0	1	0	1	0	1	0	1	0
$dgm(2,n)$	1	0	1	0	1	0	1	0	1
$dpos(2,n)$	6	18	54	162	486	1458	4374	13122	39366
$dposs(2,n)$	3	9	22	56	151	419	1198	3476	10219
$davgb(2,n)$	1.33	2.00	2.89	3.88	5.04	6.38	7.97	?	?
$davgbs(2,n)$	0.67	1.44	2.32	3.25	4.50	5.85	7.54	?	?

9.2.2 $\mathcal{DJUV}(3,n)$ und $\mathcal{DJUVM}(3,n)$

Satz 9.8 $\mathcal{DJUV}(3,3)$ kann stets B gewinnen.

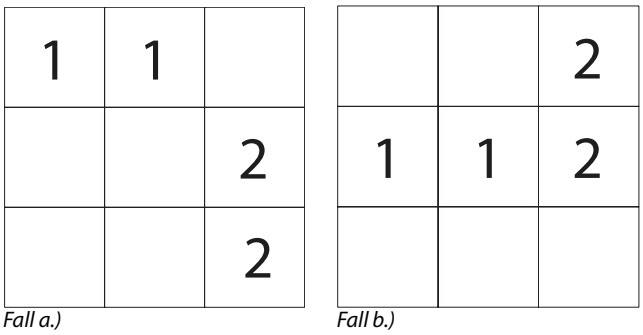


Abbildung 43: $\mathcal{DJUV}(3,3)$

Beweis: Bei $\mathcal{DJUV}(3,3)$ gibt es (modulo Symmetrien) genau 2 mögliche erste Züge (siehe Abbildungen). Für beide Situationen lässt sich leicht ein Zug für B angeben, der auf ein Spielfeld führt, in das noch genau 2 Dominos gelegt werden können. Folglich kann B stets gewinnen. □

Tatsächlich ist der Vorteil des 2. Spielers beim 3×3-Spiel so groß, dass er auch bei nichtperfekter Spielweise nur selten verlieren wird. Da das Spiel stets nach ≤ 4 Zügen endet, hat der 2. Spieler nur eine Möglichkeit, einen Fehler zu machen (seinen ersten Zug). Lässt man zwei Spieler gegeneinander $\mathcal{DJUV}(3,3)$ spielen, wobei jeder seinen Zug zufällig aus der Liste aller Nachfolgepositionen auswählt, beträgt die Wahrscheinlichkeit für einen Sieg des 2. Spielers knapp 92% ($p = \frac{190}{207}$). Sogar wenn der 1. Spieler perfekt spielt, wird er im Schnitt nur jedes 3. Spiel gegen einen zufällig ziehenden Gegner gewinnen können.

Diese Resultate lassen sich auch mit Hilfe des in Kapitel A beschriebenen Computerprogramms experimentell überprüfen. Für größere Spielfelder, für die eine vollständige Analyse (noch) nicht möglich ist, kann es eine Möglichkeit sein, zwei gleiche Computerprogramme wiederholt gegeneinander spielen zu lassen, um eventuell einen ersten Hinweis darauf zu erhalten, welcher Spieler prinzipiell im Vorteil ist. Dabei sollte das verwendete Computerprogramm natürlich deutlich besser als zufällig spielen.[57]

Satz 9.9 $\mathcal{DJUVM}(3,3)$ kann stets A gewinnen.

A belegt in seinem ersten Zug die ersten beiden Felder der ersten Zeile.[58] Dann lässt sich zu jeder möglichen Erwiderung von B leicht ein Gewinnzug angeben. Die Details sind einfach und werden daher ausgelassen. □

Satz 9.10 $\mathcal{DJUV}(3,2n+1)$ kann für $0 \leq n \leq 4$ A stets gewinnen.

Beweis: Die Fälle $n = 0$ und $n = 1$ haben wir bereits gezeigt. Wir beweisen nun noch die Aussage für $n = 2$. Dazu diskutieren wir alle möglichen Spielzüge. A belegt bei seinem ersten Zug die ersten beiden Felder der ersten Zeile. In den folgenden Abbildungen sind die 24 daraufhin möglichen Züge von B sowie jeweils eine gute Antwort für A angegeben. □

1	1	2	2	
				3
				3

Fall a.)

1	1		2	2
		3		
		3		

Fall b.)

1	1			
2	2			
3	3			

Fall c.)

1	1			
	2	2		
	3	3		

Fall d.)

1	1			
		2	2	
		3	3	

Fall e.)

1	1			
			2	3
			2	3

Fall f.)

1	1			
3	3			
2	2			

Fall g.)

1	1			
	3	3		
	2	2		

Fall h.)

1	1			
		3	3	
		2	2	

Fall i.)

[57] Ein Problem bei dieser Vorgehensweise könnte sein, dass es eine Gewinnstratgie für einen der beiden Spieler (oder allgemeiner eine gute Spielstragie) gibt, die auf Grund der Struktur der Computerspieler nicht erkannt und nie oder zu selten gespielt wird. [58] Ebenso möglich ist jeder dazu symmetrische Spielzug.

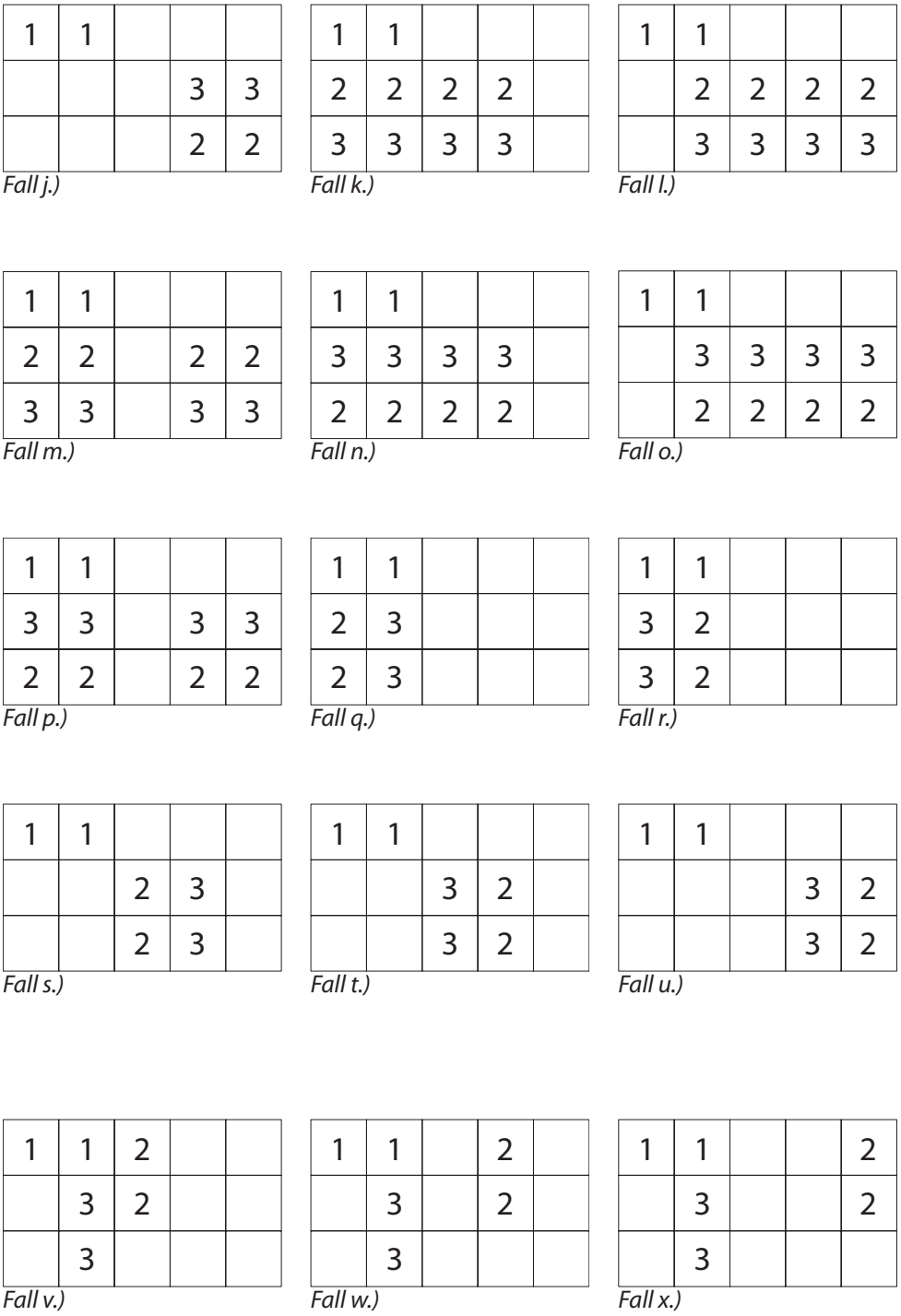


Abbildung 44: $\mathcal{DJUV}(3,5)$

- a.) entspricht $\mathcal{DJUV}(2,4)$, da das isolierte Feld rechts oben durch keinen Zug belegt werden kann und die übrigen freien Felder ein 2×4-Brett darstellen.

- b.) entspricht einem $\mathcal{DJUV}(2,4)$, aus dem einige Zugmöglichkeiten entfernt wurden. Es ist leicht zu zeigen, dass A gewinnen kann.
- c.) entspricht $\mathcal{DJUV}(3,3)$
- d.) entspricht einem $\mathcal{DJUV}(3,3)$-Spiel, aus dem einige Zugmöglichkeiten entfernt wurden. Auch hier sieht man schnell, dass A stets gewinnen kann.
- e.) In den beiden Teilfeldern sind jeweils genau 2 Dominozüge möglich
- f.) A kann stets erzwingen, dass 4 Züge bis zum Spielende nötig sind.
- g.)-j.) analog zu c.)-f.)
- k.)-p.) Es sind stets genau 2 Züge möglich
- q.) und r.) analog zu c.)
- s.) und t.) analog zu e.)
- u.) analog zu j.)
- v.)-x.) analog zu d.)

Für $n = 3$ und $n = 4$ wurde die Aussage mit Hilfe des Computerprogramms aus Kapitel A verifiziert.

Vermutung 9.11 $\mathcal{DJUV}(3, 2n + 1)$ kann $\forall n$ stets A gewinnen. Die Fälle $n = 0$ und $n = 1$ sind trivial. Sei $n \geq 2$. Ein guter erster Zug besteht dann darin, die ersten $2(n-1)$ Felder der ersten Zeile zu belegen. Zieht B nun horizontal in der ersten Zeile, kann A stets gewinnen, indem er durch ein vertikales Domino entweder ein $\mathcal{DJUV}(2,n)$-Spiel oder eine Position von $\mathcal{DJUV}(2,n+1)$ mit genau einer belegten Spalte erzeugt. Zieht B horizontal in der zweiten bzw. dritten Zeile, kopiert A diesen Zug in die jeweils andere Zeile. In beiden Fällen kann A gewinnen. Noch zu diskutieren bleiben die vertikalen Zugmöglichkeiten von B. Es bleibt weiters zu zeigen, dass diese Spielweise für alle n zum Sieg führt.

Satz 9.12 $\mathcal{DJUVM}(3, 2n)$ kann $\forall n \geq 2$ stets A gewinnen.

Beweis: Ein guter erster Zug besteht darin, die mittlere Zeile komplett zu belegen. Dadurch sind im restlichen Spiel nur noch Zeilenzüge möglich. A spielt nun nach folgenden Regeln:

- Wenn B in einer der beiden freien Zeilen noch Platz für genau ein Domino lässt, belege die andere Zeile komplett.
- Wenn B eine Zeile komplett belegt, lass in der anderen Zeile Platz für genau ein Domino.
- Ansonsten kopiere stets den Zug von B in der jeweils anderen Zeile.

Diese Spielweise führt dazu, dass B stets den letzten Zug macht und folglich das Misère-Spiel verliert. □

Die folgende Tabelle zeigt einige Resultate für $\mathcal{DJUV}(3,n)$ bzw. $\mathcal{DJUVM}(3,n)$.

n	3	4	5	6	7	8	9	10
$dg(3,n)$	0	1	3	5	4	4	≥ 1	≥ 1
$dgm(3,n)$	1	2	3	3	4	6	≥ 1	≥ 1
$dpos(3,n)$	98	550	3054	17014	?	?	?	?
$dposs(3,n)$	18	164	805	4414	23937	132763	?	?
$davgb(3,n)$	3.06	4.42	5.89	7.60	?	?	?	?
$davgbs(3,n)$	2.06	3.91	5.65	7.40	9.46	11.73	?	?

9.2.3 $\mathcal{DJUV}(4,n)$ und $\mathcal{DJUVM}(4,n)$

Die folgende Tabelle zeigt einige Resultate für $\mathcal{DJUV}(4,n)$ bzw. $\mathcal{DJUVM}(4,n)$.

n	4	5	6	7
$dg(4,n)$	0	2	0	≥ 1
$dgm(4,n)$	1	2	?	?
$dpos(4,n)$	5700	58830	?	?
$dposs(4,n)$	778	15021	?	?
$davgb(4,n)$	6.27	?	?	?
$davgbs(4,n)$	5.87	8.15	?	?

9.2.4 $\mathcal{DJUV}(5,n)$ und $\mathcal{DJUVM}(5,n)$

Die folgende Tabelle zeigt einige Resultate für $\mathcal{DJUV}(5,n)$ bzw. $\mathcal{DJUVM}(5,n)$.

n	5	6	7
$dg(5,n)$	1	≥ 1	?
$dgm(5,n)$	1	?	?
$dpos(5,n)$	?	?	?
$dposs(5,n)$	141363	?	?
$davgb(5,n)$	?	?	?
$davgbs(5,n)$	10.82	?	?

9.2.5 Allgemeine Resultate

Satz 9.13

$$g(\mathcal{DJUV}(2m,2n)) = 0 \quad \forall m,n \geq 1.$$

In Kapitel 7.1 haben wir gezeigt, dass bei $\mathcal{DJUV}(2m,2n)$ stets der 2. Spieler gewinnen kann. Daraus folgt sofort die Aussage. □

9.3 Cram

Satz 9.14 Für alle $n \geq 2$ gilt:

$$g(\mathcal{CRAM}(2,n)) = \begin{cases} 0, & \text{wenn } n \text{ gerade} \\ 1, & \text{wenn } n \text{ ungerade} \end{cases}$$

Beweis: Sei n gerade. Wir geben eine Gewinnstrategie für den 2. Spieler (B) an: B kopiert jeden Zeilenzug von A in einer Zeile in der jeweils anderen. Bei einem vertikalen Zug von A setzt er ein vertikales Domino neben das seines Gegners. Auf diese Weise kann er erzwingen, dass das Spielbrett komplett belegt wird, d. h. kein Feld frei bleibt. Es werden also genau n Dominos gesetzt. Da wir n als gerade vorausgesetzt haben, bedeutet das, dass das Spiel nach einer geraden Anzahl von Zügen endet. Folglich wird B stets gewinnen.

Sei n ungerade. A belegt in seinem ersten Zug die erste Spalte und gewinnt nach dem Argument von oben. □

Satz 9.15 Für alle $n \geq 2$ gilt:

$$g(\mathcal{CRAMMM}(2,n)) = \begin{cases} 1, & \text{wenn } n \text{ gerade} \\ 0, & \text{wenn } n \text{ ungerade} \end{cases}$$

Beweis: Der Beweis verläuft analog zur obigen Aussage. □

Die Ergebnisse unseres Analyseprogramms legen folgende Vermutung nahe.

Vermutung 9.16 Für alle $n \geq 1$ gilt:

$$g(\mathcal{CRAMMM}(3,n)) = \begin{cases} 0, & \text{wenn } n \equiv 0 \pmod 3 \\ 1, & \text{wenn } n \equiv 1 \pmod 3 \text{ oder } n \equiv 2 \pmod 3 \end{cases}$$

Satz 9.17

$$g(\mathcal{CRAM}(2m,2n)) = 0 \quad \forall m,n \geq 1.$$

Beweis: In Kapitel 7.1 haben wir gezeigt, dass bei $\mathcal{CRAM}(2m,2n)$ stets der 2. Spieler gewinnen kann. Daraus folgt sofort die Aussage. □

Für weitere Ergebnisse verweisen wir erneut auf die Tabelle am Ende dieser Arbeit.

10 Andere Spielbretter, höherdimensionales Juvavum und weitere Varianten

10.1 Treppen und der Diamant der Azteken

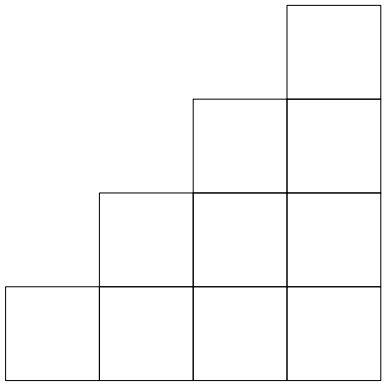


Abbildung 45: Treppe der Ordnung 4

Definition 10.1 (Treppe) Eine Treppe der Ordnung n besteht aus n Zeilen mit jeweils $1, 2, \ldots, n$ Feldern, die wie in Abbildung 45 angeordnet sind.

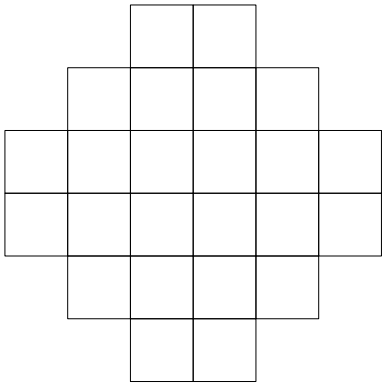


Abbildung 46: Aztekendiamant der Ordnung 3

Definition 10.2 (Aztekendiamant) Ein Aztekendiamant (*Aztec diamond*) der Ordnung n besteht aus 4 Treppen der Ordnung n, die wie in Abbildung 46 zusammengesetzt sind.

Satz 10.3 Juvavum, Domino Juvavum und Cram auf einem Aztekendiamant beliebiger Größe kann stets B gewinnen.

Beweis: Der Beweis verläuft analog zum Fall eines rechteckigen $2m \times 2n$-Felds (vgl. Kapitel 7.1). B spiegelt jeden Zug seines Gegners am Symmetriezentrum (in der Mitte zwischen den 4 Treppen). Dadurch endet das Spiel in jedem Fall nach einer geraden Anzahl von Zügen. Folglich kann er stets gewinnen. □

10.2 Juvavum auf einem Torus oder einem Möbius-Band

Man könnte alle in dieser Arbeit vorgestellten Juvavum-Spiele auch auf einem Torus betrachten. Das würde bedeuten, dass die Spielfelder keinen Rand hätten, sondern zyklisch wären. Wir werden diesen Fall nicht näher behandeln, verweisen jedoch auf [LMR], wo einige diesbezügliche Ergebnisse für das Spiel Domineering angegeben werden.

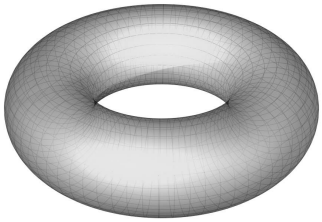


Abbildung 47: Torus

Ebenfalls denkbar wäre es, das Spielfeld auf ein Möbius-Band zu zeichnen, wodurch es zwar einen oberen und unteren Rand hat, horizontal jedoch zyklisch verläuft. Wir betrachten beispielhaft den Fall eines $1 \times n$-Spielbretts.

Satz 10.4 Möbius-$\mathcal{JUV}(1,n)$ und Möbius-$\mathcal{DJUV}(1,n)$ kann stets der 1. Spieler gewinnen.

Beweis: Der Gewinnzug besteht bei Juvavum für alle n und Domino Juvavum für gerades n darin, im ersten Zug sämtliche Felder zu belegen. Im Fall Domino Juvavum und n ungerade wird im ersten Zug genau ein beliebiges Feld freigelassen.[59] □

Satz 10.5 Möbius-$\mathcal{JUVM}(1,n)$ und Möbius-$\mathcal{DJUVM}(1,n)$ kann stets der 1. Spieler gewinnen.

Beweis: Der 1. Spieler zieht so, dass sein Gegner noch genau einen Stein setzen kann. Dadurch macht dieser den letzten Zug und verliert. □

[59] Bei Domino Juvavum gibt es noch andere Gewinnzüge. Solange man keine zwei benachbarten Felder freilässt, ist es egal, wie viele Felder im ersten Zug nicht belegt werden.

Satz 10.6 Möbius-$\mathcal{CRAM}(1,n)$ kann $\forall n \geq 2$ stets derjenige Spieler gewinnen, der normales $\mathcal{CRAM}(1,n-2)$ (bei perfektem Spiel seines Gegners) verliert. Der Grundywert des Spiels ist immer 0 oder 1.

Beweis: Jeder erste Zug bei Möbius-$\mathcal{CRAM}(1,n)$ führt auf die Startposition eines gewöhnlichen $CRAM(1,n-2)$-Spiels. Hat dieses Grundywert 0, ist der Grundywert des Möbius-Spiels 1 (alle Nachfolger haben Grundywert 0). Ist der Grundywert größer 0, besitzt das Möbius-Spiel Grundywert 0 (alle Nachfolger haben Grundywert > 0).

Satz 10.7 Möbius-$\mathcal{CRAMM}(1,n)$ kann $\forall n \geq 2$ stets derjenige Spieler gewinnen, der $\mathcal{CRAMM}(1,n-2)$ (bei perfektem Spiel seines Gegners) verliert. Der Grundywert des Spiels ist immer 0 oder 1.

Beweis: Analog.

n	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
$mcg(n)$	1	1	0	0	0	1	0	0	0	1	0	0	0	0	0	1	0	0	0
$mcgm(n)$	0	0	1	1	0	0	0	1	1	0	0	0	1	0	0	0	1	1	0

Tabelle 6: Grundywerte für lineares Möbius-Cram und Möbius-Misère-Cram

Bemerkung 10.8 Möbius-$\mathcal{CRAM}$ entspricht kreisförmigem Dawson's Kayles (vgl. Kapitel 2.5.4).

Um ein Gefühl dafür zu bekommen, was es heißt, ein Spiel auf einem Torus zu spielen, sei [Wee08] empfohlen. Diese Software enthält 8 einfache Spiele und simuliert wahlweise einen Torus oder eine Klein'sche Flasche als Spielfeld.

10.3 Mehrdimensionales Juvavum

Eine nahe liegende Erweiterung wäre es zudem, sämtliche Juvavum-Formen auch im höherdimensionalen (insbesondere 3-dimensionalen) Raum zu betrachten. Wir beschränken uns in dieser Arbeit allerdings auf den zweidimensionalen Fall.

10.4 Weitere Juvavum-Varianten

Neben Juvavum, Domino Juvavum und Cram sind natürlich etliche Varianten dieser Spiele denkbar. Ohne auf weitere Details einzugehen, möchten wir einige davon vorschlagen.

- **Triomino Juvavum (auch 3-Juvavum)**: Variante von Juvavum mit 1×3-Steinen
- **k-Juvavum**: Verallgemeinerung von Juvavum mit $1 \times k$-Spielsteinen
- **Triomino Cram**: Variante von Cram. Es wird mit 1×3-Steinen gespielt.

- **Hole Cram**: Interessante Variante: Ein *Spielstein* besteht aus zwei 1×1-Steinen und einem 1×1-Loch dazwischen.

In einem weiteren Schritt könnte man für ein Spiel auch verschiedene Spielsteine (z. B. 1×1- **und** 1×2-Steine) zulassen. Dies führt zu einer Vielzahl neuer Spiele, von denen wir abschließend mit Pentominoes und Blokus zwei bereits etablierte kurz vorstellen.

10.5 Pentominoes

Das Spiel *Pentominoes* wurde Mitte der 50er Jahre des 20. Jahrhunderts von Solomon W. Golomb vorgeschlagen und 1973 von Hallmark als Brettspiel auf den Markt gebracht. Dabei werden Pentominoes[60] von 2 Spielern abwechselnd auf ein 8×8-Spielbrett gelegt. Sieger ist, wer den letzten Zug macht. Mit Hilfe eines Computers lässt sich zeigen, dass der 1. Spieler stets gewinnen kann [Orm96].

10.6 Blokus

Das Spiel *Blokus*[61] (block-us) ist ein 1990 von Bernard Tavitian (Universität Yale) erfundenes Brettspiel. Jeder Spieler verfügt über 21 Spielsteine seiner Farbe (1 Monomino, 1 Domino, 2 (verschiedene) Triominos, 5 (verschiedene) Tetrominos und 12 (verschiedene) Pentominos), die nach bestimmten Regeln[62] auf ein 20×20-Brett gelegt werden. Eine Variante ist *Blokus Trigon*, bei dem die Spielsteine aus Dreiecken aufgebaut sind und auf ein hexagonales Spielfeld mit 486 Dreiecksfeldern gesetzt werden. In der Regel wird *Blokus* zu viert gespielt, es sind jedoch auch 2 oder 3 Spieler möglich. Zudem gibt es eine Variante für 2 Spieler auf einem 14×14-Brett (*Blokus Duo*). Das Spiel ist beendet, sobald kein Spieler mehr einen Stein setzen kann. Dann werden jedem Spieler für jedes von ihm belegte Feld Punkte gutgeschrieben. Wenn ein Spieler alle seine Steine verwenden konnte, erhält er zudem Bonuspunkte.

Mit Hilfe eines Symmetriearguments kann man sehr leicht zeigen, dass der zweite Spieler bei *Blokus Duo* stets den letzten Zug machen kann.[63] In Punkten gerechnet reicht diese Taktik allerdings nur für ein Unentschieden, da dem zweiten Spieler in diesem Fall genau gleich viele Steine bleiben wie seinem Gegner.

[60] Vgl. [Wei08] [61] http://www.blokus.com [62] Sobald bereits ein Stein einer Farbe gesetzt ist, darf jeder weitere nur so gelegt werden, dass zumindest ein Teilquadrat dieses Steins zu einem Teil eines Steins derselben Farbe *diagonal* benachbart ist und kein Teil des Spielsteins direkt (horizontal oder vertikal) an einen anderen Stein der selben Farbe angrenzt. [63] Das Argument ist das gleiche wie bei Juvavum auf einem $2m \times 2n$-Feld. Zwar gibt es bei *Blokus* die Einschränkung, dass der erste Zug jedes Spielers ein bestimmtes (auf dem Spielbrett markiertes) Feld belegen muss. Diese Felder sind jedoch so angelegt, dass die Spiegelungsstrategie trotzdem möglich ist.

A Programmbeschreibung

Im Folgenden werden die wichtigsten Teile des Analyseprogramms beschrieben und ausgewählte Teile des JAVA-Quellcodes angegeben. Für den kompletten Sourcecode sei auf die beiliegende CD sowie die Homepage des Autors[64] verwiesen.

A.1 Grundywerte von Cram

Das folgende Programm berechnet mit Hilfe des Hauptsatzes über Summenspiele Grundywerte für $\mathcal{CRAM}(1, n)$. Wichtig ist dabei, dass einmal berechnete Werte gespeichert werden (wir benutzen dazu ein Array) und im weiteren Programmverlauf nicht neu berechnet werden müssen. Auf diese Weise können die ersten 1000 Werte auf einem modernen PC in unter einer Sekunde gefunden werden.

```
import java.util.Enumeration;
import java.util.Vector;

public class LinearCramAnalyse {

  private static int[] gValues;

  private static int getGrundy(int n) {
    if (gValues[n]!=-1) // schon berechnet?
      return gValues[n];
    else {
      Vector values = new Vector();
      for (int i = 0; i <= n-2; i++) {
        int tmp=getGrundy(i)^getGrundy(n - 2 - i);
        values.add(tmp);
      }
      gValues[n]=mex(values);
      return gValues[n];
    }
  }

  private static int mex(Vector values){
    int i=0;
    int mex=-1;
    while (mex == -1) {
      boolean found = false;
      for (Enumeration e1 = values.elements(); e1.hasMoreElements();)  {
        if (((Integer)e1.nextElement()).intValue() == i) {
          found = true;
          break;
        }
      }
      if (!found)
        mex = i;
      i++;
```

[64] http://www.mschneider.cc/projects/juvavum

```
    }
    return mex;
  }

  public static void main(String args[]) {
    gValues=new int[1001];
    gValues[0]=0;
    gValues[1]=0; // Startwerte der Rekursion
    for (int i=2;i<=1000;i++) gValues[i]=-1; // alle anderen Grundywerte
        auf -1 setzen
    for (int i=0;i<=1000;i++)
      System.out.println("g(CRAM(1,"+i+"))="+getGrundy(i));
  }
}
```

Listing 1: LinearCramAnalyse.java

Bei allen anderen Spielen ist eine Anwendung des Haupsatzes nicht möglich, da dort Positionen im Allgemeinen nicht in eine einfache Summe kleinerer Spiele zerfallen. Zur Analyse dieser Spiele (sowie aller Misère-Formen, für die der Hauptsatz ohnehin nicht anwendbar ist), stellen wir im Folgenden einen anderen Ansatz vor, der den Hauptteil unseres Analyseprogramms ausmacht.

A.2 Darstellung eines Spielfelds

Ein Juvavum-Spielfeld wird durch die Klasse Board repräsentiert. Jedes Board-Objekt beinhaltet die Höhe und die Breite des Spielfelds als **int**-Werte sowie die Belegung in Form eines zweidimensionalen Arrays von **boolean**-Werten.

Daraus lässt sich auch die Binärcodierung einer Spielposition berechnen, die wir bereits in Kapitel 4.1 besprochen haben (vgl. die Methode flatten() in Listing 2).

```
/**
 * @author Martin Schneider
 *
 * Die Klasse Board stellt ein Juvavum-Spielfeld dar. Sie beinhaltet die
 * Größe des Spielbretts (width, height) sowie die Belegung der einzelnen
 * Felder. Die Klasse kann zur Analyse aller Juvavum-Varianten verwendet
 * werden.
 *
 */
public class Board {

  private boolean[][] board;

  private int h;

  private int w;

  /**
   * Erstellt ein leeres Spielfeld-Objekt (alle Felder werden als frei
```

```
   * initialisiert)
   *
   * @param w
   *            Breite des Spielfelds
   * @param h
   *            Höhe des Spielfelds
   */
  public Board(int h, int w) {
    if (h*w>63) throw new IllegalArgumentException("Spielfelder mit mehr
        als 63 Feldern sind in dieser Version noch nicht möglich.");
        this.board = new boolean[w][h];
    this.h = h;
    this.w = w;
    fill();
  }

  /**
   * Erstellt ein Spielfeld-Objekt aus der entsprechenden Binärkodierung.
   * Dabei wird der übergebene integer-Wert in eine Binärfolge umgewandelt
   * und die Felder des Spielbretts werden entsprechend der Werte dieser
     * Folge zeilenweise gesetzt.
   *
   * z.B.: 39 --> 1*2^0 + 1*2^1 + 1*2^2 + 1*2^5--> 1 1 1 0 0 1 0 0 0.
   *
   * Sollte die Binärkodierung ein ungültiger Wert für die übergebene
   * Spielfeldgröße sein, wird eine Exception geworfen.
   *
   * @param binary
   *            Binärkodierung des Spielfelds (als Dezimalzahl)
   * @param w
   *            Breite des Spielfelds
   * @param h
   *            Höhe des Spielfelds
   */
  public Board(long binary, int h, int w) {
    this.h = h;
    this.w = w;

    int x = 0;
    int y = 0;
    this.board = new boolean[w][h];
    String z = decimalToBinary(binary);
    int j = 0;
    for (int i = z.length() - 1; i >= 0; i--) {
      x = j / w;
      y = j % w;
      if (z.substring(i, i + 1).equals("1"))
        board[y][x] = true;
      else
        board[y][x] = false;
      j++;
    }
```

```
}

/**
 * Gibt das Spielfeld auf der Konsole aus.
   * X ... Feld belegt
   * - ... Feld frei
 */
public void output() {
  for (int i = 0; i < board[0].length; i++) {
    for (int j = 0; j < board.length; j++) {
      if (board[j][i] == true)
        System.out.print("X ");
      else
        System.out.print("- ");

    }
    System.out.println();
  }
  System.out.println();
}


/**
 * Gibt eine Binärdarstellung des Spielbretts (als Dezimalzahl) zurück.
 *
 * z.B.: 1 1 1 0 0 0 0 0 1 --> 1*2^0 + 1*2^1 + 1*2^2 + 1*2^8 = 263.
 *
 * @return Binärkodierung des Spielbretts
 */
public long flatten() {
  int length = w * h;
  long value = 0;
  int x, y = 0;
  for (int i = 0; i < length; i++) {
    x = i / w;
    y = i % w;
    if (board[y][x])
      value += Math.pow(2, i);
  }
  return value;
}

/**
 * @param x
 *             horizontale Koordinate
 * @param y
 *             vertikale Koordinate
 * @return true, wenn Feld(x,y) belegt ist false, wenn frei
 */
public boolean isSet(int x, int y) {
  boolean ret = false;
  try {
```

```
      ret = board[x - 1][y - 1];
    } catch (ArrayIndexOutOfBoundsException e) {
      return true;
    }
    return ret;
  }

  /**
   * @param x
   *            horizontale Koordinate
   * @param y
   *            vertikale Koordinate
   * @return true, wenn Feld(x,y) frei ist false, wenn belegt
   */
  public boolean isFree(int x, int y) {
    boolean ret = false;
    try {
      ret = !board[x - 1][y - 1];
    } catch (ArrayIndexOutOfBoundsException e) {
      return false;
    }
    return ret;
  }

  /**
   * Setzt das Feld an der Position (x,y) des Spielbretts auf true (belegt)
      .
   *
   * @param x
   *            horizontale Koordinate
   * @param y
   *            vertikale Koordinate
   */
  public void set(int x, int y) {
    board[x - 1][y - 1] = true;
  }

  /**
   * Setzt das Feld an der Position (x,y) des Spielbretts auf false (nicht
   * belegt).
   *
   * @param x
   *            horizontale Koordinate
   * @param y
   *            vertikale Koordinate
   */
  public void clear(int x, int y) {
    board[x - 1][y - 1] = false;
  }

  /**
   * @return Breite des Spielfelds
```

```
 */
public int getWidth() {
  return w;
}

/**
 * @return Höhe des Spielfelds
 */
public int getHeight() {
  return h;
}

/**
 * Hilfmethode: Gibt die Binärdarstellung einer Dezimalzahl zurück.
 */
private String decimalToBinary(long zahl) {
  String z = "";
  long r;
  do {
    r = zahl % 2;
    z = r + z;
    zahl = zahl / 2;
  } while (zahl != 0);
  return z;
}

/**
 * Hilfmethode: Füllt das 2-elementige Array, welches das Spielbrett
 * darstellt mit false (nicht belegt) auf.
 */
public void fill() {
  for (int i = 0; i < board.length; i++) {
    for (int j = 0; j < board[i].length; j++) {
      board[i][j] = false;
    }
  }
}
```

Listing 2: Board.java

Bemerkung A.1 Zur Identifikation einer Spielposition x innerhalb des Programms verwenden wir meist die Binärdarstellung (vgl. Kapitel 4.1). Der Wert $B(x)$ wird dabei als Variable vom Typ **long** definiert, was gewisse Einschränkungen nach sich zieht, da dadurch nur Werte bis $2^{63}-1$ möglich sind. Wir müssen uns also in der aktuellen Version des Programms auf Spielfelder mit ≤ 63 Feldern beschränken, was jedoch für unsere Zwecke völlig ausreicht.

In einer zukünftigen Programmversion könnte man diese Einschränkung zum Beispiel dadurch umgehen, dass man die Binärcodierung zeilen- oder spaltenweise (anstatt über das gesamte Spielfeld) bildet. Dies würde Spielfelder mit bis zu 63 Feldern pro Zeile bzw. Spalte erlauben, hätte aber auch den Nachteil, dass die Darstellung der Spielpositionen

komplizierter wird (man bräuchte für die Binärcodierung eine Liste von Werten anstatt eines einzelnen Werts).

A.3 Drehen und Spiegeln von Spielfeldern

In Kapitel 6 haben wir festgestellt, dass es zu jeder Spielposition 8 symmetrische Positionen gibt. Dadurch kann die Anzahl der zu untersuchenden Spielfelder stark reduziert werden. Die dafür notwendigen Methoden befinden sich ebenfalls in der Klasse Board.

```
/**
 * Spiegelt das Spielbrett vertikal
 */
public Board fliplr() {
  int dy = board.length;
  int dx = board[0].length;
  boolean help;
  for (int xi = 0; xi < dx; xi++) {
    for (int yi = 0; yi < dy / 2; yi++) {
      help = board[yi][xi];

      board[yi][xi] = board[dy - 1 - yi][xi];
      board[dy - 1 - yi][xi] = help;
    }
  }
  return this;
}

/**
 * Spiegelt das Spielbrett horizontal
 */
public Board flipud() {
  int dy = board.length;
  int dx = board[0].length;
  boolean help;
  for (int xi = 0; xi < dx / 2; xi++) {
    for (int yi = 0; yi < dy; yi++) {
      help = board[yi][xi];
      board[yi][xi] = board[yi][dx - 1 - xi];
      board[yi][dx - 1 - xi] = help;
    }
  }
  return this;
}

  /**
 * Dreht das Spielbrett um 180 Grad
 */
public Board rotate180() {
  flipud();
  fliplr();
  return this;
```

```
}

/**
 * Dreht das Spielbrett um 270 Grad
 */
public Board rotate270() {
  boolean[][] help = new boolean[w][h];
  for (int i = 1; i <= h; i++) {
    for (int j = 1; j <= w; j++) {
      help[j - 1][w - i] = isSet(i, j);
    }
  }
  board = help;
  return this;
}

/**
 * Dreht das Spielbrett um 90 Grad (im Uhrzeigersinn)
 */
public Board rotate90() {
  boolean[][] help = new boolean[w][h];
  for (int i = 1; i <= h; i++) {
    for (int j = 1; j <= w; j++) {
      help[h - j][i - 1] = isSet(i, j);
    }
  }
  board = help;
  return this;
}

/**
 * Spiegelt das Spielfeld an der Hauptdiagonalen
 */
public Board flipd1() {
  boolean help;
  for (int i = 1; i < board.length; i++) {
    for (int j = 0; j < i; j++) {
      help = board[i][j];
      board[i][j] = board[j][i];
      board[j][i] = help;
    }
  }
  return this;
}

/**
 * Spiegelt das Spielfeld an der Nebendiagonalen
 */
public Board flipd2() {
  boolean help;
  for (int i = 0; i < board.length; i++) {
    for (int j = 0; j < w - 1 - i; j++) {
```

```
        help = board[i][j];
        board[i][j] = board[h - 1 - j][w - 1 - i];
        board[h - 1 - j][w - 1 - i] = help;
      }
    }
    return this;
  }
}
```

Listing 3: Methoden zum Spiegeln und Drehen von Spielfeldern

A.4 Das Position-Objekt

Das zweite wichtige Objekt zur Verwaltung der Spielpositionen ist das Position-Objekt und beinhaltet neben der Spielfeldbelegung (Board) die Informationen über alle Nachfolger (in Form einer Liste).

Wichtig ist in diesem Zusammenhang, dass die Liste der Nachfolger nur Verweise auf die entsprechenden Position-Objekte (und nicht die Daten selbst) enthält, da es sonst bereits bei kleinen Spielfeldern zu erheblichen Speicherproblemen kommt. Weiters wird darauf geachtet, dass jede Spielposition nur einmal abgespeichert wird.

```
public class Position implements Comparable {
  long value;
  int h;
  int w;
  int special;
  int grundy = -1;
  private BufferedWriter out;
  Position parent = null;
  Vector children = new Vector();

  final String FILENAME = "Ergebnisse.txt";

  public Position(int value) {
    special = -1;
  }

  public Position(Board b) {
    value = b.flatten();
    h = b.getHeight();
    w = b.getWidth();
  }

  public Position(Board b, int h, int w) {
    value = b.flatten();
    this.h = b.getHeight();
    this.w = b.getWidth();
  }

  public Position(Board b, int special) {
```

```
    value = b.flatten();
    h = b.getHeight();
    w = b.getWidth();
  }

  public void addChild(Position pos) {
    children.add(pos);
  }

  public Vector getChildren() {
    return children;
  }

  public Board getBoard() {
    return new Board(value, h, w);
  }

  public boolean hasChildren(Position p) {
    long value = p.getValue();
    for (Enumeration e1 = children.elements(); e1.hasMoreElements();) {
      if (((Position) e1.nextElement()).getValue() == value)
        return true;
    }
    return false;
  }

  public int getHeight() {
    return h;
  }

  public int getWidth() {
    return w;
  }

  public int succCount() {
    if (children.isEmpty())
      return 0;
    int count = 0;
    for (Enumeration e1 = children.elements(); e1.hasMoreElements();) {
      count += ((Position) e1.nextElement()).succCount();
    }
    return children.size() + count;
  }

  public int directSuccCount() {
    return children.size();
  }

  public String toString() {
    return value + " | " + grundy + " | " + children.size() + " | " +
    succCount();
  }
```

```
  public void setGrundy(int grundy) {
    this.grundy = grundy;
  }

  public int getGrundy(int fileOutput) {
    [...]
  }

  public int getGrundyMisere(Position winEdge, int fileOutput) {
    [...]
  }

  public long getValue() {
    return value;
  }

  public void sortChildren() {
    Collections.sort(children);
  }
}
```

Listing 4: Position.java

A.5 Alle Nachfolger einer Position finden

Wie bereits erwähnt, bietet die Repräsentierung einer Spielposition die Möglichkeit, Nachfolger zu speichern. Im Folgenden sollen nun Algorithmen beschrieben werden, um diese zu finden.

Eine Möglichkeit besteht darin, mit Hilfe einer Schleife das nächste freie Feld (bei Juvavum) bzw. das nächste Loch der Länge ≥ 2 (bei Domino Juvavum) zu finden (zuerst zeilen-, dann spaltenweise), es zu füllen und die neue Position als Nachfolger zu speichern, d. h. einen Verweis auf die gefundene Postion zur Liste der Nachfolger der ursprünglichen Position hinzuzufügen. Von dieser Position kann nun rekursiv weitergesucht werden. Dabei muss allerdings berücksichtigt werden, dass eventuell noch weitere Züge in derselben Zeile beziehungsweise Spalte möglich sind, die dann ebenfalls als Nachfolger der ursprünglichen Position vermerkt werden müssen (dies geschieht in den Methoden moveCurrentRow sowie moveCurrentColumn).[65]

Im Folgenden finden sich nun die wesentlichen Teile dieses Algorithmus im Quellcode:

```
// finde Nachfolger
private static void move(Board b, int caller) {
    moveRow(b, caller);    // suche Zeilenzüge
    moveColumn(b, caller); // suche Spaltenzüge
}
```

[65] Bei Cram darf stets nur ein Domino pro Zug gesetzt werden. Zur Analyse dieses Spiels sind diese beiden Methoden daher nicht notwendig.

```
// finde Nachfolger mit Zug in Zeile
private static void moveRow(Board b, int caller) {
    int posNr = 0;
    Position tmp = null;
    int i = 0;
    int j = 0;
    int foundPos = -1;

    while (i < b.getHeight()) {
        j = 0;
        while (j + 1 < b.getWidth()) {

            // es wurde eine freie Position
            // (Loch der Länge >= 2) gefunden

            if (b.isFree(i, j) && b.isFree(i, j + 1)) {
                // in dieses Loch wird ein Domino gesetzt

                b.set(i, j);
                b.set(i, j + 1);

                // diese neue Position wird nun in der Liste
                // der bereits erzeugten Positionen gesucht



               ... symmetrische Positionen in Liste suchen ...



                foundPos = pos.search(b.flatten());
                tmp = new Position(b);

                // wird die Position
                // (oder eine symmetrische) nicht gefunden

                if (foundPos == -1) {
                    if (!pos.get(caller).hasChildren(tmp)) {

                        // neue Position (tmp) der Liste
                        // aller Positionen hinzufügen

                        posNr = pos.add(tmp);


                        // der Position, von der aus
                        // gesucht wurde (caller)
                        // die neue Position als
                        // Nachfolger hinzufügen (addChild)

                        pos.get(caller).addChild(pos.get(posNr));
```

```
                        // in dieser Zeile weitersuchen
                        // (sind Züge mit mehr als einem
                        // Domino möglich?)

                        moveCurrentRow(b, caller, i);


                        // weitere Züge suchen (rekursiver Aufruf)

                        move(b, posNr);

                    }


                } else {
                // die gefundene Position wurde bereits erzeugt


                    // wenn sich die neue Position noch nicht in der
                    // Nachfolgerliste der ursprünglichen Position
                    // befindet, wird sie dort hinzugefügt

                    if (!pos.get(caller).hasChildren(tmp))
                        pos.get(caller).addChild(pos.get(foundPos));


                    // in dieser Zeile weitersuchen (sind Züge mit
                    // mehr als einem Domino möglich)

                    moveCurrentRow(b, caller, i);
                }


                // Domino wieder entfernen

                b.clear(i, j);
                b.clear(i, j + 1);

            }
            j++;
        }
        i++;
    }
}


// finde Nachfolger mit Zug in Spalte
private static void moveColumn(Board b, int caller) {

    ... analog zu Zeilenzug ...
```

```
}


// finde alle Nachfolger mit Zug in selber Zeile
private static void moveCurrentRow(Board b,
    int caller, int rowNr) {

    ...

}


// finde alle Nachfolger mit Zug in selber Spalte
private static void moveCurrentColumn(Board b,
    int caller, int columnNr) {

    ...

}
```

Listing 5: Methoden zum Finden von Nachfolgern

Wendet man diesen Algorithmus rekursiv auf eine Ausgangsposition an, erhält man schließlich einen kompletten Spielegraphen. Es handelt sich dabei um keinen Baum, da jede Spielposition nur einmal vorkommt und folglich Ebenen übersprungen werden können. Zum Beispiel kann eine Position mit zwei Dominos in einer Zeile beziehungsweise Spalte von der Startecke sowohl in einem Zug als auch in zwei Zügen erreicht werden.

Ein anderer Ansatz bestünde darin, alle möglichen Domino-Belegungen für Löcher der Länge l mit $2 \leq l \leq max(h, w)$ zu erzeugen und jede dieser Belegungen dann in die Löcher einer Spielposition zu kopieren, um auf diese Art Nachfolger zu erzeugen. Auch hier muss die Möglichkeit beachtet werden, dass ein Zug auch in mehreren Löchern gleichzeitig möglich sein kann (ab $w \geq 5$ oder $h \geq 5$). Da sich die Laufzeit beider Varianten nicht wesentlich unterscheidet (und das Setzen mehrerer Dominos in einem Zug mit dem zweiten Ansatz nicht ohne weiteres realisierbar ist), beschränken wir uns auf die erste Variante.

A.6 Berechnung der Grundywerte

Haben wir nun ausgehend von einer Startposition einen kompletten Spielegraphen erzeugt (d. h. alle Blätter des Graphen sind Zielecken), können wir uns folgenden Algorithmus zur Berechnung der Grundywerte überlegen.

Dazu rufen wir uns erneut die Definition des Grundywertes einer Ecke in Erinnerung: Der Grundywert einer Spielposition x ist die kleinste Zahl aus $\mathbb{N}_0$, die bei keinem direkten Nachfolger von x als Grundywert vorkommt.

Diese lässt sich mittels folgendem Algorithmus finden:

```
/**
 * Berechnet den Grundywert der Spielposition für das
 * gewöhnliche Spiel (rekursiv)
```

```
 * @return Grundywert der aktuellen Spielposition
 */
public int getGrundy() {
    if (grundy != -1)
        // Grundywert bereits bekannt (gerechnet)
        return grundy;

    if (children.isEmpty()) { // Zielecke
        this.grundy = 0;
        return 0;

    } else {
        int i = 0;
        int grundy = -1;
        while (grundy == -1) {
            boolean found = false;
            for (Enumeration e1 = children.elements(); e1.hasMoreElements()
                ;) {
                if (((Position) e1.nextElement()).getGrundy() == i) {
                    found = true; // Grundywert i kommt bei einem
                        Nachfolger
                                  // bereits vor
                    break;
                }
            }
            if (!found) // i ist die kleinste Zahl, die bei keinem
                Nachfolger
                          // vorkommt
                grundy = i;
            i++;
        }
        this.grundy = grundy;
        return grundy;
    }

public int getGrundyMisere(Position winEdge, int fileOutput) {
  if (grundy != -1)
    return grundy;
  if (children.isEmpty() || children.contains(winEdge)) { // Zielecke
    this.grundy = 1;
    if (fileOutput==0 || (fileOutput==1 && grundy==0))
      outputFile();
    if (!children.contains(winEdge))
      children.add(winEdge);
    return 1;
  } else {
    int i = 0;
    int grundy = -1;
    while (grundy == -1) {
      boolean found = false;
      for (Enumeration e1 = children.elements(); e1.hasMoreElements();) {
        if (((Position) e1.nextElement()).getGrundyMisere(winEdge,
```

```
            fileOutput) == i) {
          found = true;
          break;
        }
      }
      if (!found)
        grundy = i;
      i++;
    }
    this.grundy = grundy;
      if (fileOutput==0 || (fileOutput==1 && grundy==0))
      outputFile();
    return grundy;
  }
}
```

Listing 6: Methode zur Berechnung von Grundywerten

A.7 Die Computergegner

Unser JAVA-Programm bietet die Möglichkeit, gegen den Computer anzutreten. Die verschiedenen Computergegner sind durch Klassen realisiert, die alle das Interface Player implementieren.

```
package juvavum.game.players;

import juvavum.analyse.Board;

public interface Player {
  public Board getSucc();
  public void setBoard(Board b);
  public String getName();
}
```

Listing 7: Das Interface Player

Kernteil dieser Klassen ist die Methode getSucc(), die zum aktuellen Spielfeld einen gültigen Nachfolgezug zurückgibt. In der Implementierung dieser Methode liegt der Hauptunterschied der einzelnen Computergegner.

A.7.1 Random Player

Dieser Spieler zieht stets auf eine beliebige Nachfolgeposition.

A.7.2 Symmetric Player

Dieser Spieler versucht mit Hilfe einer Bewertungsfunktion stets auf eine möglichst symmetrische Spielposition zu ziehen. Dadurch wird er als 2. Spieler auf $2m \times 2n$-Feldern stets gewinnen. Auch in allen übrigen Fällen stellt er (zumindest anfangs) einen ernst

zu nehmenden Gegner dar. Da diese symmetrische Spielweise für Misère-Juvavum keinen Sinn macht, steht der Symmetric Player für diese Spiele nicht zur Verfügung.

```
  private int getValue(Position pos){
    Board b=pos.getBoard();
    int w=b.getWidth();
    int h=b.getHeight();
    int symm=0;
    for (int i=1;i<=w;i++){
      for (int j=1;j<=h;j++){
        if (b.isSet(i,j)==b.isSet(w+1-i,h+1-j))symm++;
      }
    }
    return symm;  // je höher, der Wert, desto "besser" die Position
  }
}
```

Listing 8: Die Bewertungsfunktion des Symmetric Players

A.7.3 Better Symmetric Player

Dieser Spieler stellt eine Weiterentwicklung des Symmetric Players dar: Er gewinnt als 1. Spieler zusätzlich Juvavum, Domino Juvavum und Cram in der Normalform auf allen $2m \times (2n+1)$ bzw. $(2m+1) \times 2n$-Feldern $(m, n \geq 0)$. Dazu verwendet er die in Kapitel 7.1 angegebenen Gewinnstrategien. Als erster Spieler auf einem $(2m+1) \times (2n+1)$-Brett belegt er zudem die Spielfeldmitte, was sich in vielen Fällen als guter erster Zug erweist.

A.7.4 Perfect Endgame Player

Der Perfect Endgame Player berechnet einige Züge vor Ende des Spiels (genauer gesagt, sobald noch eine gewisse Anzahl an Feldern frei ist) den kompletten Spielegraphen und kann ab diesem Zeitpunkt optimal spielen.[66] Das bedeutet: Wenn es eine Gewinnmöglichkeit gibt, wird er sie nutzen. Bis zu diesem Zeitpunkt setzt er wie der Random Player zufällig.

A.7.5 Perfect Player

Wie der Name bereits vermuten lässt, spielt dieser Spieler (in gewissem Sinne) perfekt. Mit Hilfe von zuvor berechneten Grundywerttabellen (in Form von Textdateien) kann er immer auf eine Nachfolgeposition mit Grundywert 0 ziehen, falls es eine solche gibt. Kurz: Wenn er gewinnen kann, wird er gewinnen. Das bedeutet, dass ein menschlicher Spieler, um ihn zu schlagen, erstens eine Gewinnstrategie besitzen muss und zweitens über den gesamten Spielverlauf keinen Fehler (d. h. Zug auf eine Position mit Grundywert

[66] In der aktuellen Version startet die Berechnung, sobald nur noch 7 Felder frei sind. Diese Zahl könnte mit etwas mehr Aufwand variabel gestaltet werden und sollte insbesondere von der Rechenleistung des Computers sowie der Zeit, die man dem Computergegner für seinen Zug einräumen möchte, abhängen. Zudem sollte man den unterschiedlichen Rechenaufwand bei den verschiedenen Juvavum-Versionen berücksichtigen.

> 0) machen darf.[67] Unser Programm enthält bereits einige Grundywerttabellen. Weitere können damit erstellt werden.

A.7.6 Allround Player

Dieser Spieler greift auf die Funktionen zweier anderer zurück. In den Fällen, in denen der Better Symmetric Player garantiert gewinnen wird (1. Spieler auf $gerade \times ungerade$ und $ungerade \times gerade$-Feldern bzw. 2. Spieler im Fall $gerade \times gerade$ jeweils im normalen Spiel), befragt er diesen Spieler, andernfalls den Perfect Endgame Player. Auf Spielfeldern, für die man noch keine Grundywerttabellen berechnet hat, stellt dieser Spieler die beste Wahl dar und ist ohne das Wissen über eine Gewinnstrategie von einem menschlichen Spieler bereits sehr schwer zu schlagen.[68]

A.8 Die grafische Benutzeroberfläche

Abbildung 48: Startbildschirm

Das Programm ist in zwei Teile geteilt, die bereits am Startbildschirm (Abbildung 48) ersichtlich sind: den Spielmodus und den Analysemodus.

[67] Im Gegensatz zum Perfect Endgame Player, der in manchen Fällen bis kurz vor dem Spielende durchaus den einen oder anderen Fehler verzeiht. [68] Der interessierte Leser versuche den Allround Player und in weiterer Folge den Perfect Player als 1. Spieler bei $\mathcal{DJUV}(5,5)$ zu schlagen. Aus den Resultaten des Analyseprogramms wissen wir, dass es für ihn eine Gewinnstrategie gibt (da der Grundywert der Startposition > 0 ist). Wie diese aussieht, ist allerdings alles andere als klar, und es wird in der Regel einige Zeit vergehen, bevor man den Computer das erste Mal besiegt hat. Wer diese Hürde genommen hat und den Computer regelmäßig besiegt, versuche sich an der Misère-Form und in weiterer Folge an größeren Spielfeldern.

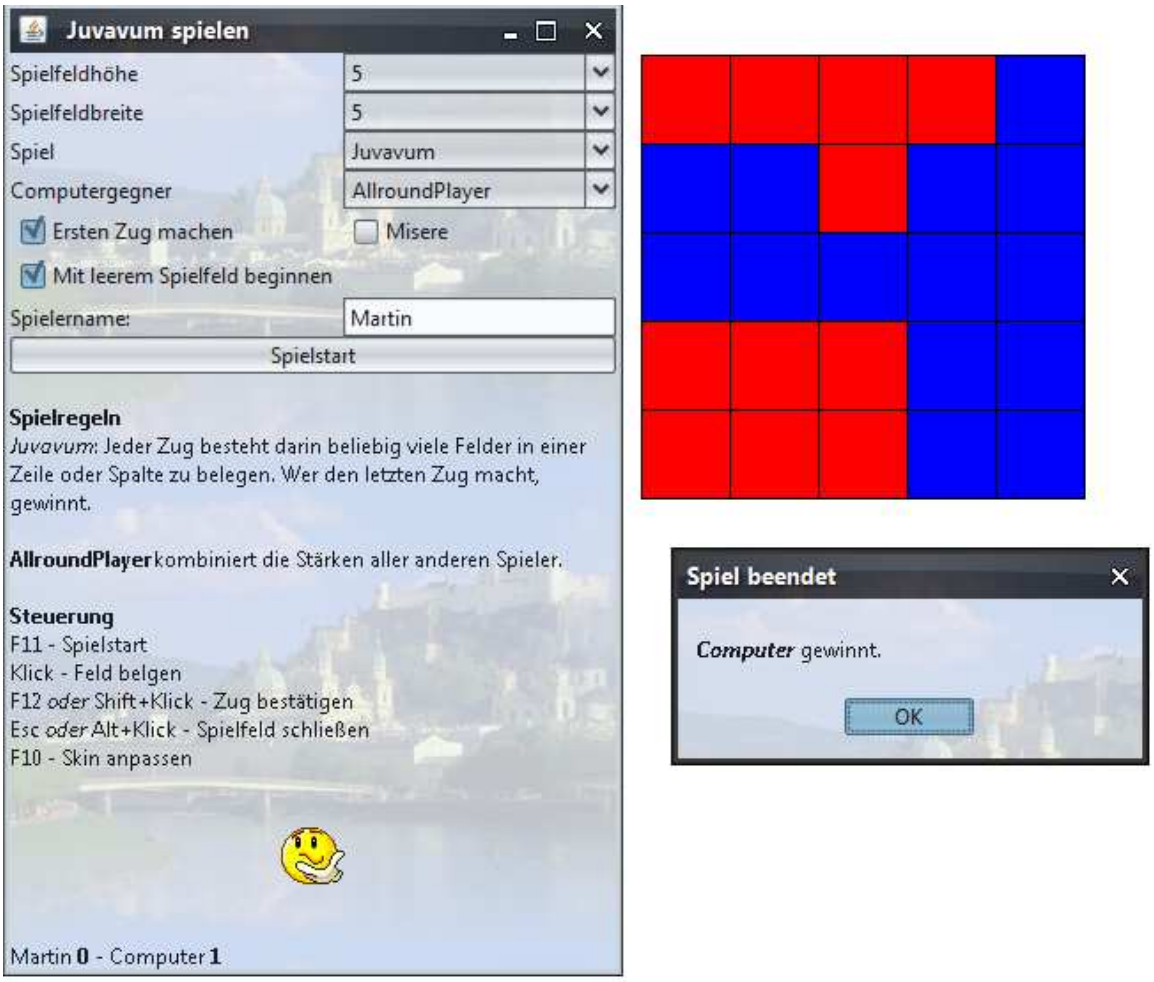


Abbildung 49: Spielbildschirm

A.8.1 Juvavum spielen

Im Spielmodus (Abbildung 49) kann gegen die im vorigen Abschnitt beschriebenen Computergegner angetreten werden. Unterstützt werden dabei folgende Juvavum-Varianten (jeweils in der normalen sowie in der Misère-Form):

- Juvavum
- Domino Juvavum
- Cram
- Hole Cram
- 3-Juvavum

A.8.2 Juvavum Analyse Tool (JAT)

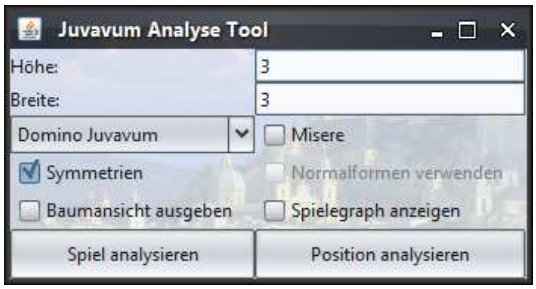


Abbildung 50: JAT: Hauptbildschirm

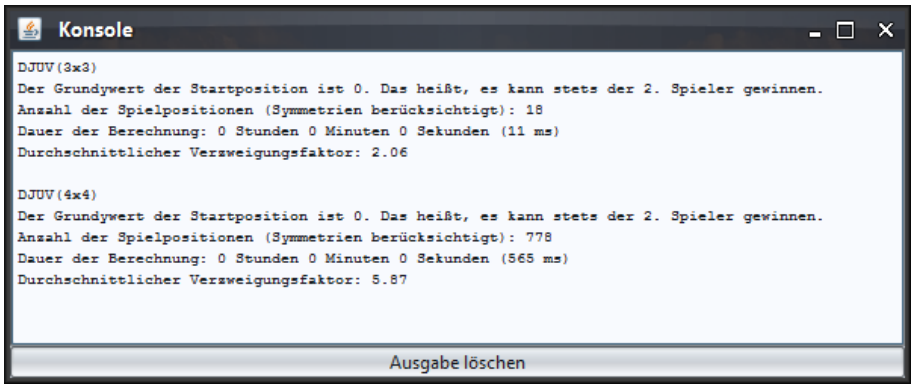


Abbildung 51: JAT: Ergebnisausgabe in Textform

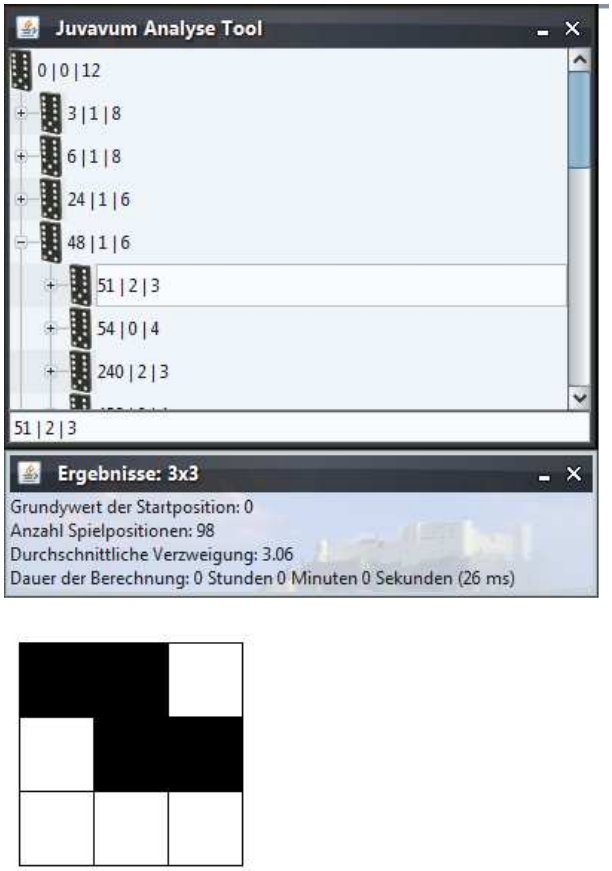


Abbildung 52: JAT: Spielegraph in Baumform

Das JAT dient der Analyse der in dieser Arbeit präsentierten Juvavum-Varianten. Das Programm unterstützt dabei verschiedene Ausgabe-Modi. Nach jeder Berechnung wird eine kurze Zusammenfassung in Textform angezeigt (Abbildung 51). Diese enthält neben dem Grundywert des untersuchten Spiels die Anzahl der Spielpositionen, den durchschnittlichen Verzweigungsfaktor sowie die Dauer der Berechnung. Optional gibt es die Möglichkeit, die Spielpositionen in Form eines Baums auszugeben (Abbildung 52) oder den Spielegraph anzeigen zu lassen (Abbildungen 53 und 54). Beides empfiehlt sich nur für kleinere Spielfelder. Vor allem die Anzeige des Spielegraphen macht nur für sehr kleine Spielfelder (bis ca. 3×3) Sinn, da die resultierenden Ausgaben ansonsten zu unübersichtlich sind.

Für diese Spielfeldgrößen ist die Ausgabe des Spielegraphen aber durchaus interessant. Das Programm bietet deshalb eine Reihe von Optionen, die Darstellung dieser Graphen anzupassen.

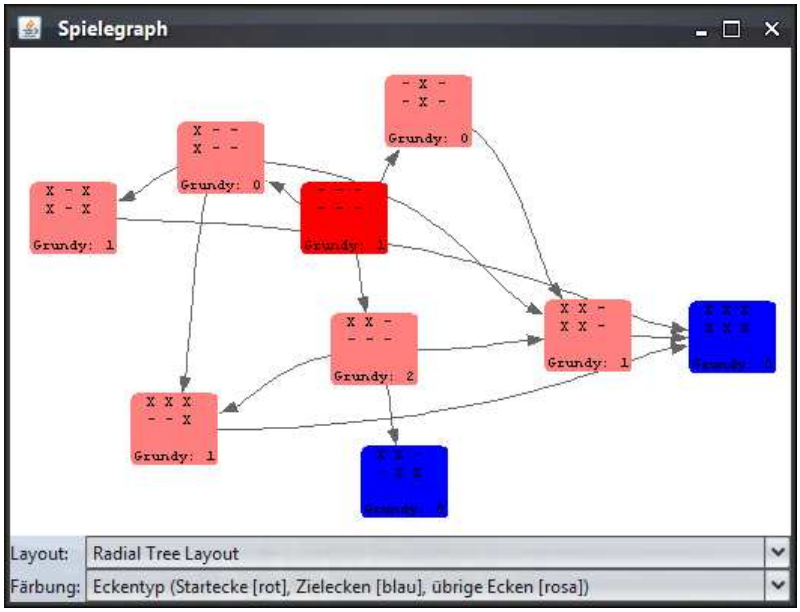


Abbildung 53: JAT: Spielegraph (Radial Tree View und Färbung nach Eckentyp)

So lassen sich z. B. die Ecken nach drei verschiedenen Kriterien einfärben.

- **Grundywerte**: Die Ecken werden nach den Grundywerten der Spielposition, die sie repräsentieren, in unterschiedlichen Rottönen gefärbt. Je höher der Grundywert, desto heller die Darstellung.
- **Eckentyp 1**: Gewinnecken werden rot, Verlustecken blau dargestellt.
- **Eckentyp 2**: Startecken werden rot, Zielecken blau, alle übrigen rosa gefärbt.

Auch die Anordnung der Ecken kann mit Hilfe verschiedener Algorithmen erfolgen.

- **Random Layout**: Die Ecken werden zufällig angeordnet.
- **Tree Layout**: Die Ecken werden in Baumform angeordnet, wobei die Wurzel links und die Blätter rechts liegen.
- **Radial Tree Layout**: Die Ecken werden in Baumform angeordnet, wobei die Wurzel innen liegt und die Blätter außen.
- **Fruchterman Reingold Layout**: Dieses Layout verwendet einen Algorithmus von Fruchterman und Reingold[69]. Vgl. dazu [FR91].

Darüber hinaus lassen sich einzelne Ecken oder der gesamte Graph per Drag & Drop (mit der linken Maustaste) verschieben. Ziehen mit gedrückter rechter Maustaste vergrößert oder verkleinert den Graphen.

Bei jeder Berechnung fragt das JAT zusätzlich nach, ob die Ergebnisse in eine Textdatei gespeichert werden sollen. Es besteht die Möglichkeit, Resultate für alle Spielpositionen des untersuchten Spiels zu speichern oder nur für solche mit Grundywert 0. Im ersten Fall kann die resultierende Datei als Grundywerttabelle für den *Perfect Player* (vgl. den vorigen Abschnitt über die Computergegner) verwendet werden.

[69] Der Algorithmus ist in erster Linie für das Zeichnen ungerichteter Graphen gedacht, liefert aber auch in unserem Fall gute Ergebnisse.

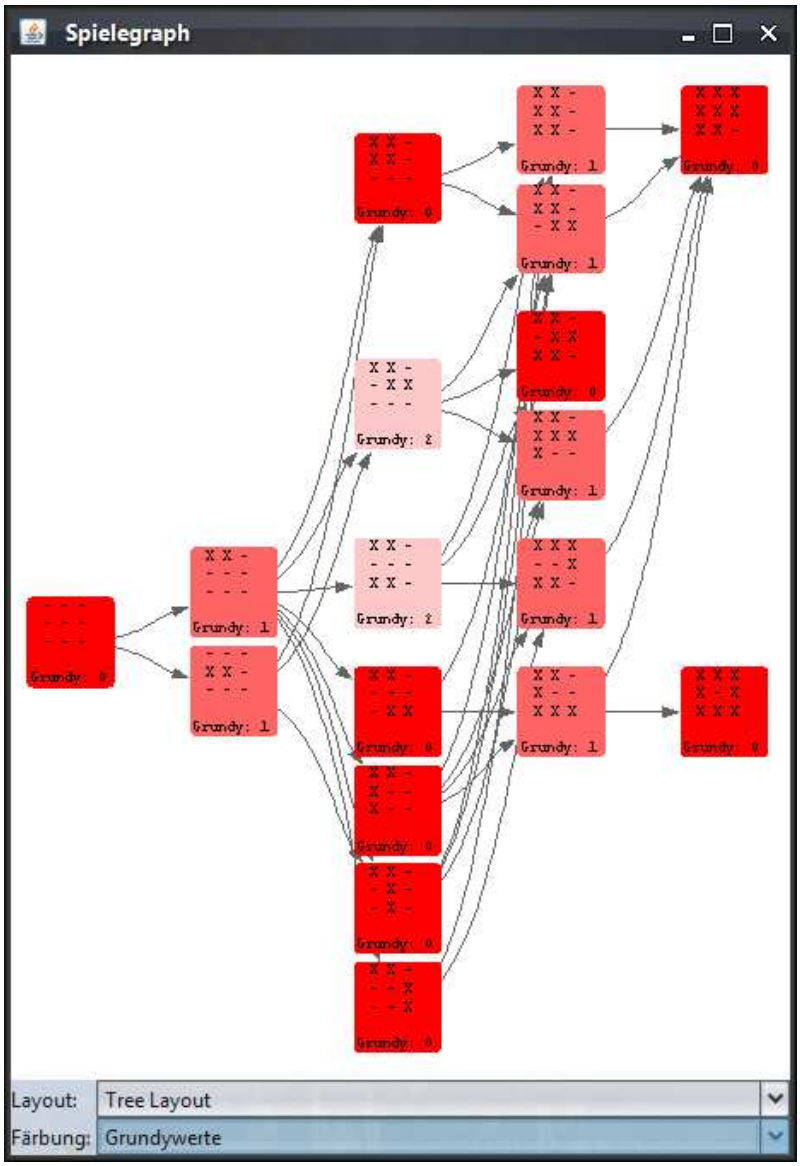


Abbildung 54: JAT: Spielegraph (Tree View und Färbung nach Grundywerten)

B Take and Break-Spiele und die oktale Darstellung eines Spiels

Eine Klasse von Spielen, die in der Literatur häufig betrachtet wird, ist jene der Take and Break-Spiele. Wir sind darauf in unserer Arbeit nicht näher eingegangen, da nur einige Spezialfälle von Juvavum in diese Kategorie von Spielen fallen. Im Folgenden bieten wir einen kurzen Überblick über die Theorie der Take and Break-Spiele.

Definition B.1 (Take and Break-Spiel) Unter einem Take and Break[70]-Spiel bezeichnen wir ein Spiel mit folgenden Eigenschaften:

- Es gibt $m > 0$ Haufen aus jeweils $n_i > 0 \quad (1 \leq i \leq m)$ Spielsteinen.
- Jeder Zug besteht darin, einen dieser Haufen zu verändern, indem man nach vorgegebenen Regeln Steine wegnimmt und/oder den Haufen in mehrere kleinere teilt.
- Jeder Spielzug verändert stets nur einen der Haufen.

Nim und Kayles sind einfache Beispiel für Take and Break-Spiele. Bei Kayles betrachte man dazu einfach die Kegel als Spielsteine und Reihen von Kegeln als Haufen von Spielsteinen.

Definition B.2 (Darstellung eines Take and Break-Spiels) Take and Break-Spiele können als Zahlenfolge in der Form $d_0.d_1d_2d_3\ldots$ dargestellt werden, wobei $d_i \geq 0$ angibt, in wie viele kleinere, nichtleere Haufen der Haufen, in dem gespielt wird, geteilt werden darf, wenn i Steine entfernt werden. Die Zahl d_i wird dabei in der Form $d_i = a_0 \cdot 2^0 + a_1 \cdot 2^1 + \ldots$ betrachtet, wobei $a_k = 1$ genau dann, wenn der Haufen in k nichtleere Haufen unterteilt werden darf. Ansonsten ist $a_k = 0$.

Beispiel B.3 (Nim) Bei Nim können beliebig viele Steine von einem Haufen entfernt werden. Daraus folgt: $d_i > 0 \quad \forall i > 0$. Die Haufen werden dabei nicht in kleinere geteilt, es bleibt also nach jedem Zug ein oder (wenn alle Steine entfernt wurden) kein Haufen übrig. Daher sind für alle $i \geq 1$ alle $d_i = 2^0 + 2^1 = 3$. Da in jedem Zug zumindest ein Stein weggenommen werden muss, ist zudem $d_0 = 0$. Nim besitzt folglich die Darstellung $0.\dot{3}$.

Beispiel B.4 (Kayles) Bei Kayles dürfen stets ein beliebiger oder zwei beliebige benachbarte Spielsteine (= Kegel) entfernt werden. Je nach Position (am Rand oder in der Mitte) bleiben ein oder zwei Haufen (= Reihen von Kegeln) übrig. Im Fall, dass der letzte bzw. die zwei letzten Kegel entfernt werden, bleibt nichts übrig. Es folgt:

$$d_0 = 0$$

$$d_1 = d_2 = 2^0 + 2^1 + 2^2 = 7$$

Kayles hat also die Darstellung 0.77. Man kann sich weiters leicht davon überzeugen, dass Dawson's Kayles die Darstellung 0.07 hat.

[70] Nehmen und teilen

Beispiel B.5 (Lineares Cram) Wir haben in Kapitel 8 die lineare Form von $\mathcal{CRAM}$ behandelt. Dieses Spiel entspricht Dawson's Kayles und hat daher ebenfalls die oktale Darstellung 0.07.

Eine gängige Konvention besteht darin, die Größe der d_i nach oben zu beschränken. In der Literatur werden z. B. oft oktale Take and Break-Spiele betrachtet.

Definition B.6 (Oktales Spiel) Ein Take and Break-Spiel heißt oktal[71], wenn $\forall i \quad d_i < 8$.

Bemerkung B.7 Die Grundywerte eines oktalen Spiels weisen in vielen Fällen eine Periode auf. Der folgende Satz gibt an, wie viele Werte untersucht werden müssen, um sicherzustellen, dass sich eine (zum Beispiel mittels eines Computerprogramms) entdeckte bzw. vermutete Periode stets fortsetzt.

Satz B.8 (Guy Smith-Periodizität) Sei G ein oktales Spiel mit endlich vielen Ziffern ungleich 0 in der oktalen Darstellung. Sei weiters k der größte Index, für den $d_k \neq 0$ gilt. H_n bezeichne einen Haufen mit n Elementen. Für $n_0 > 0$ und $p > 0$ gelte:

$$g(H_{n+p}) = g(H_n) \quad \forall n : n_0 \leq n < 2n_0 + p + k.$$

Dann folgt:

$$g(H_{n+p}) = g(H_n).$$

Beweis: Induktion nach n. Wir verweisen dazu auf [Sie08].

Beispiel B.9 (Kayles: Periodizität der Folge $g(K(n))$) Tabelle 7 zeigt die Grundywerte für $K(n)$ (Kayles mit n Kegeln) für alle $n \leq 175$. Nach den ersten 71 Werten scheint es eine Periode der Länge 12 zu geben. Es stellt sich die Frage, ob wir aus den vorhandenen Daten bereits schließen können, dass sich diese Periode unendlich fortsetzt. Wir wenden den Guy Smith-Periodizitätssatz an: In unserem Fall ist n_0 (der Index, an dem die Periode beginnt) gleich 71, p (die Länge der vermuteten Periode) ist gleich 12. Da Kayles die oktale Darstellung 0.77 besitzt, ist $k = 2$. Wir müssen die Periodizität ($g(K_{n+12}) = g(K_n)$) also lediglich für alle $72 \leq n \leq 156$ überprüfen. Das können wir anhand der Werte unserer Tabelle tun. Die ersten 167 Werte der Folge $g(K(n))$ genügen also bereits, um jeden beliebigen Wert dieser Folge bestimmen zu können: Sei $m(n) := n \pmod{12}$.

$$\forall n \geq 84 : K(n) = K(m(n) + 72)$$

[71] Oktale (und viele andere) Spiele lassen sich unter anderem sehr gut mit dem Computerprogramm [Sie07] von Aaron Siegel analysieren.

	0	1	2	3	4	5	6	7	8	9	10	11
n=0+	0	1	2	3	1	4	3	2	1	4	2	6
n=12+	4	1	2	7	1	4	3	2	1	4	6	7
n=24+	4	1	2	8	5	4	7	2	1	8	6	7
n=36+	4	1	2	3	1	4	7	2	1	8	2	7
n=48+	4	1	2	8	1	4	7	2	1	4	2	7
n=60+	4	1	2	8	1	4	7	2	1	8	6	7
n=72+	**4**	**1**	**2**	**8**	**1**	**4**	**7**	**2**	**1**	**8**	**2**	**7**
n=84+	4	1	2	8	1	4	7	2	1	8	2	7
n=96+	4	1	2	8	1	4	7	2	1	8	2	7
n=108+	4	1	2	8	1	4	7	2	1	8	2	7
n=120+	4	1	2	8	1	4	7	2	1	8	2	7
n=132+	4	1	2	8	1	4	7	2	1	8	2	7
n=144+	4	1	2	8	1	4	7	2	1	8	2	7
n=156+	4	1	2	8	1	4	7	2	1	8	2	7
n=168+	4	1	2	8	1	4	7	2				

Tabelle 7: Die Grundywerte von $K(n)$

C Dominokachelungen und dimer coverings

Wir haben in Kapitel 5.2 unter anderem die Anzahl der Dominobelegungen eines $m \times n$-Felds betrachtet und bemerkt, dass sie gerade der Anzahl aller Matchings eines geeigneten Graphen entspricht. Ein damit verwandtes Problem ist unter dem Stichwort *domino tiling* bekannt. Gesucht wird dabei die Anzahl der Möglichkeiten, ein Gitter so mit nichtüberlappenden Dominos zu füllen, dass kein Feld frei bleibt. Repräsentiert man das Spielfeld wie in Kapitel 5.2 beschrieben durch einen Graphen G, entspricht ein *domino tiling* gerade einem perfekten Matching von G.

Definition C.1 (Dominokachelung, *domino tiling*) Sei S ein (nicht notwendigerweise rechteckiges) Spiefeld, das aus 1×1-Feldern besteht. Dann ist eine Dominokachelung (*domino tiling*) von S eine vollständige Belegung von S durch nichtüberlappende 1×2-Spielsteine (Dominos).

Korollar C.2 Dominokachelungen existieren nur auf Spielfeldern mit einer geraden Anzahl von Feldern. Für $m \times n$-Felder bedeutet dies, dass mindestens eine der Zahlen m und n gerade sein muss.[72]

Eine weitere äquivalente Formulierung stammt aus der Physik. Dazu definieren wir zunächst (stark vereinfacht) folgenden Begriff:

Definition C.3 (Dimer) Ein Dimer ist ein Polymer, das aus zwei Atomen besteht.

Man stelle sich die Ecken eines Graphen G als einwertige Atome vor. Gesucht sind nun Möglichkeiten, ein durch den Graphen G repräsentiertes Gitter (*lattice*) vollständig mit Dimeren zu bedecken. Eine solche Kachelung heißt *dimer covering* von G. Es ist leicht zu sehen, dass jedes *dimer covering* einem perfekten Matching von G enstpricht und umgekehrt.

Das *dimer*-Modell ist in der Physik von großem Interesse. Kasteleyn, Fisher und Temperley lieferten unabhängig voneinander Anfang der sechziger Jahre des 20. Jahrhunderts exakte Ergebnisse über die Anzahl der *dimer coverings* auf $m \times n$-Gittern und später auch auf allgemeinen Graphen. Aussagen über die Anzahl von Dominokachelungen folgen sofort aus diesen Resultaten.

Eine detaillierte Beschreibung dieser Arbeiten würde hier zu weit führen. Wir geben daher nur einige Ergebnisse an und verweisen für weiterführende Betrachtungen auf die Literatur: [KO05], [Kas61], [Fis61], [FT61], [AS04].

Satz C.4 (Dominokachelungen eines $m \times n$-Felds) Die Anzahl der Dominokachelungen eines leeren $m \times n$-Spielfelds ist

$$\prod_{l=1}^{m} \prod_{k=1}^{n} \left| 2 \cdot \cos \frac{\pi l}{m+1} + 2i \cdot \cos \frac{\pi k}{n+1} \right|^{\frac{1}{2}}$$

	n=1	2	3	4	5	6	7	8
m=1	0	1	0	1	0	1	0	1
m=2	1	2	3	5	8	13	21	34
m=3	0	3	0	11	0	41	0	153
m=4	1	5	11	36	95	281	781	2245
m=5	0	8	0	95	0	1183	0	14824
m=6	1	13	41	281	1183	6728	31529	167089
m=7	0	21	0	781	0	31529	0	1292697
m=8	1	34	153	2245	14824	167089	1292697	12988816

Tabelle 8: Anzahl der Dominokachelungen eines $m \times n$-Spielfelds

Satz C.5 (Dominokachelungen des Aztekendiamants) Die Anzahl der Dominokachelungen eines Aztekendiamants der Ordnung n ist $2^{n \cdot \frac{n+1}{2}}$.

n	$2^{n \cdot \frac{n+1}{2}}$
1	2
2	8
3	64
4	1024
5	32768
6	2097152
7	268435456
8	68719476736
9	35184372088832
10	36028797018963968

Tabelle 9: Anzahl der Dominokachelungen eines Aztekendiamants der Ordnung n

Tabelle 8 zeigt die Anzahl der Dominokachelungen eines $m \times n$-Spielfelds, Tabelle 9 die Anzahl der Dominokachelungen eines Aztekendiamants der Ordnung n.

Eine Verallgemeinerung des *dimer*-Problems stellt die Frage nach *monomer-dimer coverings* dar. Dabei lassen wir bei der Kachelung eines Gitters neben Dimeren (Molekülen aus 2 Atomen) auch Monomere (einzelne Atome) zu. Übersetzt auf Domino Juvavum entspricht ein Dimer einem Dominostein und ein Monomer einem freien Feld. Jedes *monomer-dimer covering* eines $m \times n$-Felds entspricht daher genau einer Dominobelegung bei $\mathcal{DJUV}(m, n)$ bzw. $\mathcal{CRAM}(m, n)$.

[72] Eine gerade Anzahl von Feldern, stellt eine notwendige, nicht aber eine hinreichende Bedingung für die Existenz einer Dominokachelung auf allgemeinen Spielfeldern dar. Auf (leeren) rechteckigen Spielbrettern mit einer geraden Anzahl von Feldern existiert allerdings immer eine Dominokachelung.

Gesucht ist nun die Anzahl aller möglichen Kachelungen eines $m \times n$-Felds mit k Dimeren und $mn - 2k$ Monomeren. Summation über alle k würde schließlich die Anzahl aller möglichen Dominobelegungen eines $m \times n$-Spielfelds ergeben. Leider gibt es für dieses Problem noch keine exakte Lösung.

Eine Approximation der Anzahl der *monomer-dimer coverings* wäre unter anderem mit Hilfe einer Markov-Ketten-Monte-Carlo (MCMC)-Simulation möglich. Ein passender Algorithmus sowie einige Resultate werden unter anderem in [JS89], [Ran94], [KRS96], [BOS01] und [Wie02] angegeben.

D Lektürevorschläge

Die drei Grundlagenartikel zum Thema stammen von Bouton [Bou02], Sprague [Spr36] und Grundy [Gru39]. Bouton analysiert das Spiel Nim, Sprague und Grundy legen den Grundstein zur Analyse allgemeinerer Zwei-Personen-Spiele. Die Definition des Grundywerts einer Spielposition (vgl. Kapitel 2.4) geht auf die Arbeiten dieser beiden Mathematiker zurück.

Als modernes Standardwerk im Bereich der kombinatorischen Spieltheorie kann das Buch *Winning Ways* von Elwyn R. Berlekamp, John H. Conway und Richard K. Guy [BCG82] gesehen werden. Ebenfalls zur Grundlagenlektüre sollte *On Numbers And Games* von John H. Conway [Con83] gehören. Ein neueres Buch stammt von Michael Albert, Richard Nowakowski und David Wolfe [ANW07].

Eine gute Einführung in die Analyse von Misère-Spielen liefern Aaron N. Siegel in [Sie08] sowie die Internetseiten [Pla07b] und [Pla07a] von Thane Plambeck.

D.1 Anmerkung zum Literaturverzeichnis

Bei einigen Internetquellen, die im Literaturverzeichnis angegeben sind, ließ sich kein Erscheinungsdatum eruieren. Die Gültigkeit dieser Internetseiten wurde zuletzt Ende 2008 überprüft.

		JUVAVUM								DOMINO JUVAVUM						CRAM					
		Grundywert		#(Pos.)			branching			Grundywert		#(Pos.)		branching		Grundywert		#(Pos.)		branching	
Höhe	Breite	normal	misere		Symm.	Normal		Symm.	Normal	normal	misere		Symm.		Symm.	normal	misere		Symm.		Symm.
1	1	1	0	2	2	2	0,50	0,50	0,50												
1	2	2	2	4	3	3	1,25	1,00	1,00	1	0	2	2	0,50	0,50	1	0	2	2	0,50	0,50
1	3	3	3	8	6	4	2,38	2,00	1,50	1	0	3	2	0,67	0,50	1	0	3	2	0,67	0,50
1	4	4	4	16	10	5	4,06	3,30	2,00	2	2	5	4	1,20	1,00	2	2	5	4	1,00	0,75
1	5	5	5	32	20	6	6,59	5,45	2,50	2	2	8	5	1,63	1,40	0	1	8	5	1,25	1,00
1	6	6	6	64	36	7	10,39	8,67	3,00	3	3	13	9	2,31	2,00	3	3	13	9	1,54	1,22
1	7	7	7	128	72	8	16,09	13,50	3,50	3	3	21	12	3,05	2,67	1	0	21	12	1,81	1,58
1	8	8	8	256	136	9	24,63	21,15	4,00	4	4	34	21	4,03	3,48	1	0	34	21	2,09	1,76
1	9	9	9	512	272	10	37,44	32,23	4,50	4	4	55	30	5,20	4,70	0	1	55	30	2,36	2,17
1	10	10	10	1.024	528	11	56,67	50,03	5,00	5	5	89	51	6,67	5,88	3	1	89	51	2,64	2,35
1	11	11	11	2.048	1.056	12		75,55	5,50	5	5	144	76	8,48	7,80	3	3	144	76	2,92	2,76
1	12	12	12	4.096	2.080	13			6,00	6	6	233	127	10,72	9,61	2	0	233	127	3,19	2,96
1	13	13	13	8.192	4.160	14			6,50	6	6	377	195	13,49	12,61	2	2	377	195	3,47	3,35
1	14	14	14	16.384	8.256	15			7,00	7	7	610	322	16,91	15,44	4	1	610	322	3,75	3,57
1	15	15	15	32.768	16.512	16			7,50	7	7	987	504	21,13	20,01	0	1	987	504	4,02	3,94
1	16	16	16	65.536	32.896	17			8,00	8	8	1597	826	26,36	24,45	5	0	1597	826	4,30	4,17
1	17	17	17	131.072	65.792	18			8,50	8	8	2584	1309	32,82	31,38	2	0	2584	1309	4,58	4,52
1	18	18	18	262.144	131.328	19			9,00	9	9	4181	2135			2	2	4181	2135	4,85	4,76
1	19	19	19	524.288	262.656	20			9,50	9	9	6765	3410			3	1	6765	3410	5,13	5,09
1	20	20	20	1.048.576	524.800	21			10,00	10	10	10946	5545			3	3	10946	5545	5,40	5,34
1	21	21	21	2.097.152	1.049.600	22			10,50	10	10	17711	8900			0	0	17711	8900	5,68	5,65
1	22	22	22	4.194.304	2.098.176	23			11,00	11	11	28657	14445			1	0	28657	14445	5,96	5,91
1	23	23	23	8.388.608	4.196.352	24			11,50	11	11	46368	23256			1	1	46368	23256	6,23	6,21
1	24	24	24	16.777.216	8.390.656	25			12,00	12	12	75025	37701			3	1	75025	37701	6,51	6,48
1	25	25	25	33.554.432	16.781.312	26			12,50	12	12	121393	60813			0	3	121393	60813	6,79	6,77
2	2	0	0	16	6	6	3,00	1,50	1,50	0	1	6	3	1,33	0,67	0	1	6	3	1,33	0,67
2	3	1	2	64	24	14	5,50	4,21	3,21	1	0	18	9	2,00	1,44	1	0	18	9	2,00	1,44
2	4	0	1	256	76	33	9,13	7,71	5,21	0	1	54	22	2,89	2,32	0	1	54	22	2,70	2,14
2	5	1	0	1.024	288	95	14,44	12,97	8,24	1	0	162	56	3,88	3,25	1	0	162	56	3,41	2,88
2	6	0	2	4.096	1072	328	22,28	21,13	12,39	0	1	486	151	5,04	4,50	0	1	486	151	4,11	3,68
2	7	1	1	16.384	4224	1266	33,92	32,69	17,95	1	0	1458	419	6,38	5,85	1	0	1458	419	4,81	4,48
2	8	0	0	65.536		5140			25,36	0	1	4374	1198	7,97	7,54	0	1	4374	1198	5,52	5,27
2	9	≥ 0		262.144						1	0	13122	3476			1	0	13122	3476	6,22	6,05
2	10	0		1.048.576						0	1	39366	10219			0	1	39366	10219	6,93	6,80
3	3	0	1	512	102	75	9,75	7,11	6,39	0	1	98	18	3,06	2,06	0	1	98	18		2,06
3	4	4	0	4.096	1120	394	15,69	14,73	11,15	1	2	550	164	4,42	3,91	1	0	550	164		3,65
3	5	2	1	32.768	8640	2697	24,16	23,26	17,87	3	3	3054	805	5,89	5,65	1	0	3054	805		5,04
3	6	5	0	262.144		21004			27,00	5	3	17014	4414	7,60	7,40	4	1	17014	4414		6,19
3	7			2.097.152						4	4		23937		9,46	1	0		23937		7,42
3	8	≥ 0		16.777.216						4	6		132763		11,73	3	0		132763		
3	9			134.217.728												1	1				
4	4	0	1	65.536	8548	7760	24,50		21,85	0	1	5700	778	6,27	5,87	0	0	5700	778	5,70	5,29
4	5	4	5	1.048.576		86564			31,07	2	2	58830	15021		8,15	2	0	58830	15021	7,20	7,08
4	6	0		16.777.216						0						0	0				
5	5			33.554.432						1	1		141363		10,82	0	2		141363		
5	6	≥ 0		1.073.741.824						≥ 0						≥ 0					
5	7			34.359.738.368												1					
6	6	0		68.719.476.736						0						0					
7	7			562.949.953.421.312																	

Abbildungsverzeichnis

Tabellenverzeichnis

Quellcodeverzeichnis

Literaturverzeichnis

[ABMP91] Alt, H. ; Blum, N. ; Mehlhorn, K. ; Paul, M.: Computing a maximum cardinality matching in a bipartite graph in time $o(n^{1.5}\sqrt{m/log(n)})$. In: *Inf. Proc. Letters* 37 (1991), Februar, S. 237–240

[ANW07] Albert, Michael ; Nowakowski, Richard ; Wolfe, David: *Lessons in Play. An Introduction to Combinatorial Game Theory*. Wellesley, MA : A K Peters, 2007. – ISBN 978–1–56881–277–9

[AS04] Ardila, Federico ; Stanley, Richard P.: Pflasterungen. (2004), Juli

[BCG82] Berlekamp, Elwyn R. ; Conway, John H. ; Guy, Richard K.: *Winning Ways*. London : Academic Press, 1982

[Bog07] Bogomolny, Alexander: *Date Game*. `http://www.cut-the-knot.org/Curriculum/Games/Date.shtml`, 2007

[BOS01] Beichl, Isabel ; O'Leary, Dianne P. ; Sullivan, Francis: Approximating the number of monomer-dimer coverings in periodic lattices. In: *Physical Review E* 64 (2001), Jun, Nr. 1, S. 016701. `http://dx.doi.org/10.1103/PhysRevE.64.016701`. – DOI 10.1103/PhysRevE.64.016701

[Bou02] Bouton, Charles L.: Nim, a game with a complete mathematical theory. In: *Annals of Mathematics* (1901-02), Nr. 3, S. 35–39

[Bro] Brouwer, Andries E.: *The game of Chomp*. `http://www.win.tue.nl/~aeb/games/chomp.html`,

[Bul02] Bullock, Nathan: *Domineering: Solving Large Combinatorical Search Spaces*. `http://http://www.nathanbullock.org/download/obsequi.pdf`, 2002

[CD98] Cowen, Robert ; Dickau, Robert: Playing Games with Mathematica. In: *Mathematica in Education and Research* 7 (1998), Nr. 3, S. 5–10

[Con83] Conway, John H.: *Über Zahlen und Spiele*. Braunschweig : Vieweg, 1983. – ISBN 3–528–08434–0

[CW08] Chandra, Pravin ; Weisstein, Eric W.: *Fibonacci Number*. `http://mathworld.wolfram.com/FibonacciNumber.html`, 2008

[Dit00] Ditmarsch, Hans P.: *Knowledge Games*. `http://www.cs.otago.ac.nz/staffpriv/hans/ILLCthesis.pdf`, November 2000

[FF62] Ford, Lester Randolph J. ; Fulkerson, D.R.: Flows in Networks. In: *Princeton U. Press* (1962)

[Fis61] Fisher, Michael E.: Statistical Mechanics of Dimers on a Plane Lattice. In: *Phys. Rev.* 124 (1961), Dec, Nr. 6, S. 1664–1672. http://dx.doi.org/10.1103/PhysRev.124.1664. – DOI 10.1103/PhysRev.124.1664

[FR91] Fruchterman, Thomas M. J. ; Reingold, Edward M.: *Graph drawing by Force-directed placement.* November 1991

[Fra07] Fraenkel, Aviezri S.: Combinational Games: Selected Bibliography with a Succinct Gourmet Introduction. In: *Electronic Journal of Combinatorics* DS2 (2007)

[FT61] Fisher, Micheal E. ; Temperley, H.: Dimer problem in statistical mechanics. An exact result. In: *Philos. Mag.* (1961), Nr. 6, S. 1061–1063

[Gar74] Gardner, Martin: Mathematical Games. Cram, crosscram and quadrapage: new games having elusive winning strategies. In: *Scientific American* (1974), Februar, Nr. 230, S. 106–108

[Ger05] Gerl, Peter: *Vorlesungsunterlagen zum Projektpraktikum Spiele (WS 2004/05 und SS 2005).* 2004-05

[GKP94] Graham, Ronald L. ; Knuth, Donald E. ; Patashnik, Oren: *Concrete Mathematics.* Addison-Wesley Professional, 1994

[Gru39] Grundy, P. M.: Mathematics and games. In: *Eureka* 2 (1939), S. 6–8

[HK73] Hopcroft, J. ; Karp, R.: An $n^{\frac{5}{2}}$ algorithm for maximum matchings in bipartite graphs. In: *SIAM Journal of Computing* 2 (1973), Dezember, Nr. 4, S. 225–231

[JS89] Jerrum, Mark ; Sinclair, Alistair: Approximating the Permanent. In: *SIAM J. Comput.* 18 (1989), Nr. 6, S. 1149–1178

[Kas61] Kasteleyn, P.W.: The statistics of dimers on a lattice. In: *Physica* 27 (1961), S. 1209–1225

[Knu97] Knuth, Donald E.: *Art of Computer Programming, Volume 1: Fundamental Algorithms.* Addison-Wesley Professional, 1997

[KO05] Kenyon, Richard ; Okounkov, Andrei: *What is a Dimer?* http://www.ams.org/notices/200503/what-is.pdf, März 2005

[KRS96] Kenyon, Claire ; Randall, Dana ; Sinclair, Alistair: Approximating the number of monomer-dimer coverings of a lattice. In: *Journal of Statistical Physics* 83 (1996), S. 637–659

[Kry07] Kryukov, Kirill: *Endgame Tablebases Online. 6-men endgame analysis free for everyone. (http: //kirr.homeunix.org/ chess/ tablebases-online).* 2007

[LMR] Lachmann, Michael ; Moore, Cristopher ; Rapaport, Ivan: *Who Wins Domineering on Rectangular Boards?* http://citeseer.ist.psu.edu/314065.html,

[LP86] Lovász, L. ; Plummer, M.D.: *Matching Theory*. North Holland, 1986. – ISBN 0–444–87916–1

[Orm96] Orman, Hilarie K.: Pentominoes: A First Player Win. In: *MSRI Publications* 29 (1996), S. 340–344

[Pla06] Plambeck, Thane E.: *Advances in Loosing*. http://arxiv.org/abs/math/0603027, 2006

[Pla07a] Plambeck, Thane E.: *Advances in Loosing*. http://miseregames.wordpress.com, 2007

[Pla07b] Plambeck, Thane E.: *miseregames.org*. http://www.miseregames.org, 2007

[PS08] Plambeck, Thane E. ; Siegel, Aaron N.: *Misère Quotients for Impartial Games*. http://arxiv.org/abs/math/0609825, 2008

[Ran94] Randall, Dana: *Counting in Lattices: Combinatorical Problems from Statistical Mechanics*. http://www.math.gatech.edu/~randall/r-thesis.pdf, Oktober 1994

[Sie07] Siegel, Aaron N.: *Combinatorical Game Suite*. http://www.cgsuite.org, 2007

[Sie08] Siegel, Aaron N.: Misere Games and Misere Quotients. (2008), February

[SL93] Saip, Herbert Alexander B. ; Lucchesi, Cláudio L.: *Matching Algorithms for Bipartite Graphs*. http://citeseer.ist.psu.edu/176173.html, März 1993

[Spr36] Sprague, R. P.: Über mathematische Kampfspiele. In: *Tohoku Mathematical Journal* 41 (1935-36), S. 438–444

[Wee08] Weeks, Jeff: *Torus Games*. http://www.geometrygames.org/TorusGames/, 2008

[Wei08] Weisstein, Eric W.: *Pentomino*. http://mathworld.wolfram.com/Pentomino.html, 2008

[Wie02] Wiebe, Eduard: *Markov-Ketten Monte-Carlo Methode*. wwwcs.uni-paderborn.de/cs/ag-madh/vorl/Perlen/Wiebe.ps.gz, Januar 2002

[Wik07] Wikipedia: *Match des Jahrhunderts*. http://de.wikipedia.org/wiki/Match_des_Jahrhunderts, 2007