Strategiesynthese für Paritätsspiele auf endlichen Graphen

Von der Fakultät für
Mathematik, Informatik und Naturwissenschaften der
Rheinisch-Westfälischen Technischen Hochschule Aachen zur
Erlangung des akademischen Grades eines Doktors der
Naturwissenschaften genehmigte Dissertation

vorgelegt von

Diplom-Informatiker

Jens Vöge

aus Schleswig

Berichter: Universitätsprofessor Dr. Wolfgang Thomas
Universitätsprofessor Dr. Erich Grädel

Tag der mündlichen Prüfung: 18. Dezember 2000

Diese Dissertation ist auf den Internetseiten der Hochschule online verfügbar.

Zusammenfassung

Paritätsspiele sind unendliche Zwei-Personen-Spiele, hier betrachtet auf endlichen Graphen. Eine Partie ist ein unendlicher Pfad durch den Graphen, der von den beiden Spielern durch wechselseitige Wahl von Knoten aufgebaut wird. Der Gewinner der Partie ist durch die in ihr unendlich oft auftretenden Knoten festgelegt.

Das Problem der Lösung von Paritätsspielen (d.h. die Bestimmung des Gewinners für Partien von einem gegebenen Anfangsknoten aus sowie einer Gewinnstrategie) gehört zur Komplexitätsklasse NP ∩ co-NP. Es ist eines der Kernprobleme der Theorie der Programmverifikation, denn viele Model-Checking-Probleme lassen sich in Polynomzeit auf die Lösung von Paritätsspielen reduzieren.

In der vorliegenden Arbeit wird ein neuer Algorithmus zur Lösung von Paritätsspielen vorgestellt. Im Gegensatz zu den bisherigen diskreten Verfahren folgt er einem Ansatz der Strategieverbesserung, wie er bereits für stochastische Spiele bekannt ist. Anders als für die Verfahren der Literatur ist noch kein Beispiel bekannt, welches eine exponentielle Laufzeit belegen würde. Der Algorithmus basiert auf einer neuartigen Bewertung unendlicher Partien.

Abstract

Parity games are infinite two person games, here considered on finite graphs. A play is an infinite path in the graph, whose vertices are chosen by the two players in alternation. The winner of the play is determined by the vertices that are visited infinitely often in the play.

The problem of solving a parity game (i.e., finding the winner for plays starting in a given vertex and the construction of a winning strategy) belongs to the complexity class NP ∩ co-NP. It is one of the core problems in the theory of program verification, because many model checking problems can be reduced to solving parity games by a polynomial time reduction.

In this thesis a new algorithm for solving parity games is presented. Contrary to known discrete procedures this one uses the method of strategy improvement as it is already known from stochastic games. Unlike for procedures in the literature, no example is known for the present algorithm which requires polynomial time. The algorithm is based on a new kind of valuation for infinite plays.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Ein großer Teil der heutigen Informatik-Systeme folgt nicht mehr dem klassischen Paradigma der Transformation von Daten (von Eingabe zu Ausgabe), sondern wird durch nichtterminierende reaktive Prozesse realisiert. Kennzeichen solcher Prozesse ist, dass ihre Berechnungen nicht mit einem Ausgabe-Ergebnis enden, sondern dass ihre Aktionen in andauerndem Wechselspiel mit der jeweiligen Umgebung ausgeführt werden, ohne dass in der Regel ein Endzustand erreicht wird. Milner hat diese Entwicklung durch das Motto ‚computing is interaction' beschrieben ([Mil89]).

Die klassischen Berechnungsmodelle (wie etwa Turingmaschine, endlicher Automat mit der üblichen Semantik) sind diesem neuen Paradigma nicht angepasst. Ausgehend von Arbeiten der sechziger Jahre von Church [Chu63], Büchi [Büc62], Rabin [Rab69] und anderen rückte in den letzten Jahren ein einfaches Modell reaktiver Systeme in den Vordergrund, das auch Gegenstand der vorliegenden Arbeit ist. Es handelt sich um die unendlichen Zweipersonen-Spiele auf endlichen Graphen. Hier wird die Reaktivität durch das Wechselspiel zwischen den zwei Spielern erfasst, die ‚Korrektheit' des unendlichen Verhaltens dagegen durch Gewinnbedingungen an unendliche Partien.

Spielgraphen

Ein endliches System (ein System mit endlich vielen Zuständen) wird dabei durch einen endlichen ‚Spielgraphen' repräsentiert. Seine Knoten sind die Zustände des betrachteten Gesamtsystems, dessen Verhalten durch zwei Parteien, die Spieler, bestimmt wird. Als Namen für die beiden Spieler verwenden wir 0 und 1. Jeder Zustand (Knoten) v ist einem der Spieler zugeordnet, und von v aus kann dieser Spieler durch Wahl einer ausgehenden Kante zum Zielpunkt dieser Kante übergehen. Man kann sich vorstellen, dass auf diese Weise eine Marke von Knoten zu Knoten verschoben wird, und zwar abwechselnd durch beide Spieler, wenn die Zuordnung der Knoten zu den Spielern dies vorschreibt. Die Partien sind unendlich lang. Eine Partie wird von einem festzulegenden Knoten aus begonnen, worauf die

Spieler entsprechend der Knotenzuordnung ihre Züge durchführen. Eine Partie lässt sich als unendliche Folge der besuchten Knoten ansehen. Spieler 0 gewinnt eine Partie, wenn eine vorgegebene Gewinnbedingung an diese Partie erfüllt ist. Für die Form dieser Gewinnbedingung gibt es viele Möglichkeiten.

Automatentheoretische Gewinnbedingungen

In der vorliegenden Arbeit verfolgen wir den Fall, dass der Gewinn einer Partie (etwa durch Spieler 0) von der Menge der unendlich oft besuchten Knoten der betrachteten Partie abhängt. In Kapitel 2 wird eine Reihe solcher Gewinnbedingungen genauer beschrieben.

Diese Gewinnbedingungen greifen auf die Theorie der ω-Automaten zurück. Dabei gewinnt Spieler 0 genau die Partien, die der betrachtete Automat akzeptiert. Die Betrachtung von Automaten mit unendlichen Läufen geht auf Büchi ([Büc62]) zurück. Später wurden weitere ω-Automatentypen eingeführt durch Muller ([Mul63]) Rabin ([Rab72]) und Streett ([Str82]). Eine Schlüsselstellung nimmt die so genannte Paritätsbedingung ein, die von Emerson und Jutla ([EJ91]) und vorher schon in anderer Formulierung von Mostowski ([Mos84]) eingeführt wurde. In diesem Fall sind den Knoten des Spielgraphen Farben zugeordnet und diese Farben sind nummeriert. Eine Partie erfüllt die Gewinnbedingung für Spieler 0, wenn die größte in ihrem Verlauf unendlich oft vorkommende Farbe gerade ist.

Aus den Arbeiten von Emerson, Jutla und Mostowski folgt, dass sich alle oben genannten Akzeptierbedingungen auf die Paritätsbedingung zurückführen lassen. Dabei ist jedoch eine Vergrößerung des Zustandsraumes der betrachteten Automaten notwendig. Insbesondere kann der aus einem Muller-Automaten konstruierte Paritätsautomat exponentiell größer sein kann als der gegebene.

Konstruktion von Gewinnstrategien

Ist ein Spielgraph mit automatentheoretischer Gewinnbedingung gegeben, stellen sich drei grundlegende algorithmische Fragen:

1. Kann man zu jedem Knoten v des Spielgraphen entscheiden, ob Spieler 0 (bzw. Spieler 1) von v aus eine Gewinnstrategie besitzt?

2. Welche Verfahren genügen zur Ausführung von Gewinnstrategien (z.B. realisiert durch bestimmte Typen abstrakter Automaten)?

3. Kann man solche Gewinnstrategien effektiv bestimmen?

Aus Ergebnissen der deskriptiven Mengenlehre folgt zur ersten Frage, dass von jedem Knoten des betrachteten Spielgraphen aus einer der beiden Spieler

eine Gewinnstrategie besitzt. Dies ergibt sich aus der Tatsache, dass die automatentheoretischen Gewinnbedingungen zu Spielen führen, die in der deskriptiven Mengenlehre zur zweiten Stufe der Borel-Hierarchie gehören, also determiniert sind.

Einen Durchbruch in der Beantwortung der genannten Fragen erzielten Büchi und Landweber [BL69], indem sie für endliche Spielgraphen mit Muller-Gewinnbedingung folgendes zeigten:

1. Man kann algorithmisch entscheiden, welcher der beiden Spieler von einem gegebenen Knoten eines Spielgraphen aus eine Gewinnstrategie besitzt.

2. Eine solche Gewinnstrategie ist durch einen endlichen Automaten mit Ausgabe realisierbar; man benötigt also nur einen Speicher fester Größe (für den jeweils bisherigen Verlauf einer Partie), um darauf aufbauend die richtige Zugwahl zu treffen.

3. Ein derartiger ‚Strategieautomat' ist aus dem Spielgraphen und der Gewinnbedingung effektiv konstruierbar.

Ein entscheidender weiterer Schritt in dieser Analyse wurde durch Emerson Jutla und Mostowski ([EJ91, Mos84]) erreicht: Ist die Gewinnbedingung als Paritätsbedingung gegeben, so genügen für beide Spieler jeweils Gewinnstrategien, die positional sind. Solche Strategien treffen die Kantenauswahl nur anhand der jeweils erreichten Knoten, benötigen also keinen Speicher für den vorangehenden Teil der Partie.

Wie Thomas in [Tho95] gezeigt hat, lässt sich auch die Lösung von Muller-Spielen auf die Analyse von Paritätsspielen zurückführen, wenn man den Spielgraphen entsprechend expandiert (allerdings um einen exponentiellen Faktor).

Eine positionale Strategie (etwa für Spieler 0), wie sie für Paritätsspiele genügt, ist durch eine Teilmenge der Kantenmenge definiert (in der für jeden Knoten von Spieler 0 genau eine ausgehende Kante aufgenommen ist). Ein triviales Verfahren zur Bestimmung einer Gewinnstrategie besteht also darin, eine Kantenmenge zu ‚raten' und dann die Eigenschaft zu prüfen, ob sie tatsächlich eine Gewinnstrategie repräsentiert, was sich in Polynomzeit durchführen lässt. Hieraus ergibt sich, dass das Problem der Bestimmung der Gewinnknoten von Spieler 0 zur Komplexitätsklasse NP gehört. Gemäß Symmetrie zwischen Spieler 0 und Spieler 1 ist das Problem sogar in der Komplexitätsklasse NP $\cap$ co-NP. Jurdziński hat in [Jur98] gezeigt, dass das Problem auch in der Klasse UP $\cap$ co-UP liegt. Eine wichtige Frage der aktuellen Forschung zielt auf die Klärung, ob das Problem in Polynomzeit lösbar ist.

Hauptresultat

In dieser Arbeit wird ein neues Verfahren zur Lösung von Paritätsspielen vorgestellt. Es verfolgt einen anderen Ansatz als die bisher bekannten diskreten

Konstruktionen von McNaughton ([McN93, Tho95]) und Jurdziński ([Jur00b]). Diese Verfahren diskutieren wir einleitend in Kapitel 2. Unser Ansatz folgt einer Idee von Puri ([Pur95]). In dieser Dissertation wurde ein Optimierungsverfahren beschrieben, welches eine gegebene Strategie durch sukzessive Modifikation der Kantenauswahl nach einem geeigneten Kriterium ‚optimiert'. Die Schwierigkeit in Puris Verfahren liegt in der Verwendung reeller Zahlen zur Definition des Optimalitätskriteriums, welche mit hoher Genauigkeit berechnet werden müssen. Die Bestimmung dieser Genauigkeit ist nicht trivial und macht eine praktische Durchführung des Verfahrens problematisch.

Kernpunkt des Beitrags der vorliegenden Arbeit ist die Einführung einer (recht elementaren) diskreten Bewertungsstruktur, die zweierlei erfüllt:

1. Die Durchführung der Optimierungsschritte ist in Polynomzeit möglich (und zwar in Zeit $O(n^3)$, wobei n die Anzahl der Knoten des Spielgraphen ist).

2. Die Ergebnisse der Optimierungsschritte lassen sich in Korrespondenz zu denen nach Puris Verfahren bringen.

Hierdurch wird einerseits eine Brücke zwischen dem ‚kontinuierlichen' Bewertungsschema nach Puri und diskreten Bewertungen geschlagen, andererseits erstmalig ein Strategieoptimierungsverfahren entwickelt, dessen Iterationsschritte effizient durchführbar sind.

Die Anzahl der Iterationsschritte ist allerdings (bisher) nicht durch ein Polynom abschätzbar, so dass die allgemeine Frage nach der Polynomzeit-Lösung für Paritätsspiele offen bleibt. Allerdings ist es nun möglich, diese Frage anhand des in dieser Arbeit vorgestellten konkreten Algorithmus, des diskreten Strategieverbesserungsalgorithmus (dSVA), zu studieren: Gelingt insbesondere eine Beschränkung der Iterationsschritte durch ein Polynom, so ist das Paritätsspiel-Problem gelöst und als polynomial lösbar nachgewiesen.

Anwendungen 1: Controller-Synthese

Der Anwendungshorizont dieser algorithmischen Resultate zur Strategiesynthese zerfällt in zwei Bereiche. Zunächst bedeutet die algorithmische Konstruktion von Gewinnstrategien in Paritätsspielen, dass man in den entsprechenden reaktiven Systemen für einen der beiden Spieler den Gewinn durch eine algorithmisch bestimmte Strategie garantieren kann. Trifft dies auf Spieler 0 zu, so kann man ihn als Controller in seiner Umgebung sehen, der bei beliebigen Aktionen seiner Umgebung sicherstellt, dass die entstehenden Partien der vorgelegten Gewinnbedingung genügen. Damit ist also die algorithmische Konstruktion von Gewinnstrategien ein Zugang zur automatischen Synthese von Controllern.

Anwendungen 2: Model-Checking (Verifikation)

Ein zweites Anwendungsfeld ist in den vergangenen Jahren intensiv studiert worden: das Model-Checking-Problem für den μ-Kalkül. Dieses wird in der vorliegenden Arbeit über eine Implementierung des diskreten Strategieverbesserungs-Algorithmus näher untersucht.

Ziel der Verifikation ist es zu zeigen, dass ein zustandsbasiertes System unter Einhaltung einer Spezifikation arbeitet (Zustandsübergänge durchführt). Eine Spezifikation ist eine Beschreibung der gewünschten Eigenschaften von Abläufen (Zustandsfolgen) des Systems.

Wir sehen ein zustandsbasiertes System als ein Transitionsystem an, das sich zu jeder Zeit in genau einem Zustand befindet und zu gewissen Zeitpunkten in einen anderen Zustand übergeht. Dabei abstrahieren wir von den Zeitabständen zwischen solchen Änderungen. Ein solches Transitionssystem lässt sich als ein Graph darstellen, dessen Knoten die Zustände des Systems sind und dessen Kanten die Zustandsübergänge (Transitionen) des Systems repräsentieren.

Zur Beschreibung der Eigenschaften eines Systems existieren verschiedene Formalismen. Ein Formalismus, in den sich viele Spezifikationssprachen einbetten lassen (wie z.B. temporale Logiken), ist der μ-Kalkül ([Koz83]). Die Frage, ob ein gegebenes System eine in einer Logik L formulierte Eigenschaft besitzt, nennt man das Model-Checking-Problem ([CES86]) für L.

Es existieren verschiedene Verfahren zur Lösung des Model-Checking-Problems für den μ-Kalkül (Tableau-Verfahren, Umformulierung in boolesche Gleichungsysteme und deren Lösung durch Gaußelimination [Mad97]). Von Emerson, Jutla und Sistla wurde in [EJS93] gezeigt, dass das Model-Checking-Problem des μ-Kalküls polynomzeit-äquivalent zur Bestimmung der Gewinnbereiche eines unendlichen Spiels mit Paritätsgewinnbedingung ist. Damit liefert jeder Strategiesynthesealgorithmus für Paritätsspiele auch ein Model-Checking-Verfahren.

Die in [EJS93] beschriebene Transformation liefert zu einem Model-Checking-Problem ein Paritätsspiel, dessen Größe linear in dem Produkt der Größen von Systembeschreibung und zu verifizierenden Eigenschaften ist. Dabei entsteht das Paritätsspiel aus dem Transitionssystem und der die Systemeigenschaft beschreibenden Formel. Wir stellen ein alternatives Verfahren zur Konstruktion des Paritätsspiels vor, das ebenfalls Spielgraphen linearer Größe erzeugt, aber ohne den in [EJS93] verwendeten Zwischenschritt über ‚Paritätsbaumautomaten' auskommt. Dadurch bleiben die Struktur des Transitionssystems der Systembeschreibung und der Formel besser in dem entstehenden Paritätsspiel erhalten.

Gliederung

Im zweiten Kapitel werden theoretische Grundlagen unendlicher Spiele beschrieben und notwendige Notationen eingeführt. Weiterhin werden zwei bereits bekannte Algorithmen zur Strategiekonstruktion für Paritätsspiele vorgestellt.

Das dritte Kapitel enthält das Hauptergebnis dieser Arbeit. Es wird ein Algorithmus zur Strategiesynthese für Paritätsspiele, der diskrete Strategieverbesserungs-Algorithmus (dSVA), vorgestellt. Weiterhin wird auf den Zusammenhang zum Strategiesynthesealgorithmus für Discounted-Payoff-Spiele von Puri eingegangen.

Im vierten Kapitel wird auf den Zusammenhang zwischen dem modalen μ-Kalkül und Paritätsspielen eingegangen. Es wird gezeigt, wie sich ein Model-Checking-Problem, das mit Hilfe des μ-Kalküls spezifiziert ist, auf das Ermitteln der Gewinnbereiche in einem Paritätsspiel zurückführen lässt.

Das fünfte Kapitel geht auf die Implementierung des dSV-Algorithmus und der Transformation von Model-Checkingproblemen in Paritätsspiele ein. Es wird die automatentheoretische Experimentierplattform `omega` beschrieben sowie die Anwendung dieser Implementierung anhand kleiner Fallstudien erläutert.

Dank

Ich möchte Wolfgang Thomas danken, der mir die Gelegenheit zu dieser Arbeit gegeben und mich mit gutem Rat und vielfältigen Anregungen unterstützt hat. Viele wichtige Anregungen verdanke ich auch den Gesprächen mit Marcin Jurziński während seines Aufenthaltes in Aachen und meines in Århus. Thomas Wilke hat mir nicht nur fachlich mit gutem Rat weitergeholfen. Meinen Kollegen Dietmar Berwanger, Christof Löding, P. Madhusudan und Oliver Matz danke ich für erhellende Diskussionen und Hinweise. Dominik Schmitz möchte ich danken, der bei der Implementierung unermüdlich meine Optimierungsvorschläge umsetzte. Meiner Freundin Bettina Kamps danke ich für das sorgsame Korrekturlesen und die unterstützende Aufmunterung. Ich danke meinen Eltern Elisabeth und Kuno Vöge, die schon früh mein mathematisches Interesse gefördert haben.

Notationen

Das Symbol $\mathbb{N}$ bezeichnet die Menge der nicht-negativen, ganzen Zahlen (d.h. insbesondere $0 \in \mathbb{N}$). Zur Bezeichnung eines endlichen Anfangsstücks der natürlichen Zahlen verwenden wir für alle $n \in \mathbb{N}$ die Abkürzungen:

$$[n] = \{i \in \mathbb{N} | i < n\}.$$

Bei Verwendung der Quantorenlogik folgen wir der Konvention, dass Quantoren am schwächsten binden.

Ist M eine Menge und $R \subseteq M \times M$ eine Relation auf M, so verwenden wir für alle $a \in M$ und $A \subseteq M$ die Abkürzungen:

$$\begin{aligned} aR &= \{m \in M \mid aRm\}, \\ AR &= \{m \in M \mid \exists a \in A\colon aRm\}, \\ Ra &= \{m \in M \mid mRa\}, \\ RA &= \{m \in M \mid \exists a \in A\colon mRa\}. \end{aligned}$$

Kapitel 2

Unendliche Spiele

Dieses Kapitel führt in die für diese Arbeit notwendigen Grundlagen der Theorie der unendlichen Spiele über endlichen Graphen ein. Es werden zunächst Spielgraphen mit gebräuchlichen Gewinnbedingungen für endliche und unendliche Partien vorgestellt. Danach wird genauer auf Spiele mit Paritätsgewinnbedingung eingegangen, und es werden verschiedene Ansätze zur Lösung dieser Spiele (d.h. zur Gewinnstrategiekonstruktion für diese Spiele) beschrieben. Im letzten Abschnitt wird ein Verfahren zur Gewinnstrategiekonstruktion für Paritätsspiele beschrieben, welches sich auf unendliche Spiele mit reellen Bewertungen (Payoff-Spiele) bezieht.

2.1 Spielgraphen und Gewinnbedingungen

Ein *Spielgraph* $G = (V, E)$ besteht aus einer endlichen Menge V von Knoten, die sich in zwei Mengen V_0 und V_1 aufteilt, und einer Menge von Kanten $E \subseteq V \times V$, wobei $vE \neq \emptyset$ für alle $v \in V$ ist. Eine Partie ist eine unendliche Folge $\pi = \pi_0\pi_1\pi_2 \ldots \in V^\omega$ mit $\pi_k E \pi_{k+1}$ für alle $k \in \mathbb{N}$. Eine Partie wird von zwei Spielern, Spieler 0 und Spieler 1, gespielt. Die Knoten in V_0 gehören Spieler 0 und die in V_1 Spieler 1: Die Fortsetzung einer Partie von einem Knoten u in V_0 (bzw. V_1) erfolgt durch Spieler 0 (bzw. 1), der eine Kante (u, v) wählt und damit den nächsten Knoten v der Partie bestimmt. Die Menge $\Pi = \{\pi \in V^\omega \mid \forall k \in \mathbb{N}\colon \pi_k E \pi_{k+1}\}$ bezeichne die Menge aller Partien (über dem zugrundeliegenden Graphen). Wir bezeichnen mit *Pref*(Π) die Menge aller endlichen Partiepräfixe von Partien in Π.

Beispiel

Ein Spielgraph, der auch im folgenden Kapitel zur Illustration verwendet wird, ist in Abbildung 2.1 angegeben. Bei der graphischen Darstellung von Spielgraphen werden Knoten von Spieler 0 stets als Kreise dargestellt und Knoten von Spieler 1 als Quadrate.

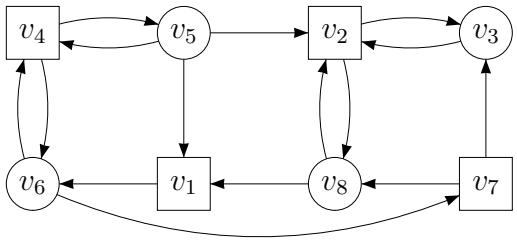


Abbildung 2.1: Spielgraph G_1.

Eine *Strategie* für Spieler i von der Knotenmenge $S \subseteq V$ aus ist eine Funktion

$$\rho : \mathit{Pref}(\Pi) \cap SV^* \cap V^*V_i \to V,$$

für die für alle $w \in \mathit{Pref}(\Pi) \cap SV^*$ auch $w\rho(w) \in \mathit{Pref}(\Pi)$ gilt. Dies bedeutet: Eine Strategie für Spieler i von $S \subseteq V$ aus liefert zu jedem Partiepräfix, das in S beginnt und in einem Knoten von Spieler i endet, einen Knoten, der an das gegebene Partiepräfix angefügt ein neues Partiepräfix ergibt.

Eine Strategie ρ von S aus heißt *positional* (oder auch *gedächtnislos* oder *memoryless*), falls für jeden Knoten $v \in V_i$ gilt:

$$\rho(w) = \rho(w') \text{ für alle } w, w' \in \mathit{Pref}(\Pi) \cap SV^* \cap V^*v.$$

Die Menge der von S erreichbaren Knoten ist:

$$R = \{v \in V | \exists w \in \mathit{Pref}(\Pi) \cap SV^* \cap V^*v\}.$$

Die Strategiefunktion einer positionalen Strategie ρ schreiben wir verkürzend als:

$$\bar{\rho} : R \cap V_i \to V$$

definiert durch:

$$\bar{\rho}(v) = \rho(w) \text{ für } w \in \mathit{Pref}(\Pi) \cap SV^* \cap V^*v.$$

Man beachte, dass $\bar{\rho}$ wohldefiniert ist: Weil ρ positional ist, kann jeweils ein beliebiger unter den möglichen Werten für w ausgewählt werden.

Da die Funktionen $\bar{\rho}$ und ρ miteinander verträglich sind (d.h. $\bar{\rho}(v) = \rho(v)$ für alle $v \in S \cap V_i$), werden wir im Folgenden keine abweichende Bezeichnung für die verkürzende Schreibweise der Strategie mehr verwenden (d.h. wir schreiben auch ρ anstelle von $\bar{\rho}$).

Eine positionale Strategie auf einer Menge $R \subseteq V$ bezeichnen wir auch als *uniforme Strategie*, um hervorzuheben, dass sich der Spieler unabhängig vom Anfangsknoten der Partie in jedem einzelnen erreichten Knoten stets gleich verhält.

Durch eine *Gewinnbedingung* wird jeder Partie ein Gewinner zugeordnet. Trifft die Gewinnbedingung auf eine Partie zu, so gewinnt Spieler 0 diese Partie und anderenfalls Spieler 1.

Existiert für einen Spieler eine Strategie, mit der er von einem Knoten v aus jede nach diese Strategie gespielte Partie gewinnt, so heißt dieser Knoten v *Gewinnknoten* für diesen Spieler. Wir bezeichnen die Menge der Gewinnknoten von Spieler 0 mit Win_0 und von Spieler 1 mit Win_1. Diese beiden Mengen heißen auch die *Gewinnbereiche* des Spiels.

2.1.1 Theorie der Spiele und automatentheoretischer Hintergrund

Die Theorie der unendlichen Spiele wurde erstmalig im Rahmen der „deskriptiven Mengenlehre“ eingeführt ([Kec94, Mos80]).

Für diese Theorie betrachtet man spezielle Spielgraphen: Als Spielgraph (V, E) dient hier ein Baum, dessen Knotenmenge $V = \{0,1\}^*$ ist und die sich auf $V_0 = \{w \in \{0,1\}^* \mid |w| \text{ gerade}\}$ für Spieler 0 und $V_1 = \{w \in \{0,1\}^* \mid |w| \text{ ungerade}\}$ für Spieler 1 aufteilt. Die Kantenmenge ist $E = \{(w, wi) \mid w \in \{0,1\}^* \wedge i \in \{0,1\}\}$.

Mit einer Menge $M \subseteq \{0,1\}^\omega$ von Pfaden lässt sich auf diesem Spielgraphen, wie in [GS53] beschrieben, ein *Gale-Stewart-Spiel* $\Gamma(M)$ definieren, dessen Gewinnbedingung genau die Pfade aus M als Gewinnpartien von Spieler 0 festlegt. Bei diesem Spiel ist man üblicherweise nur an Partien interessiert, die im Wurzelknoten ϵ des Baumes beginnen.

Ein Spiel heißt *determiniert*, wenn die Existenz einer Gewinnstrategie für einen der beiden Spieler garantiert werden kann. Eine zentrale Frage der deskriptiven Mengenlehre ist, welche Spiele determiniert sind. Ein Ansatz zur Untersuchung dieser Frage liegt in der Einteilung der Spiele $\Gamma(M)$ anhand der zugrundeliegenden Pfadmengen M. Dazu betrachtet man Klassifikationen der mengentheoretischen Topologie, so insbesondere die Borel-Hierarchie über der Cantor-Topologie auf $\{0,1\}^\omega$. Damit teilt man die Spiele nach der Platzierung ihrer Gewinnpfadmengen in der Borel-Hierarchie ein.

Gale und Stewart [GS53] haben gezeigt:

$$M \textit{ abgeschlossen } \Rightarrow \Gamma(M) \textit{ determiniert.}$$

Die daran anschließenden Untersuchungen kulminierten in dem Satz von Martin [Mar75]:

$$M \text{ Borel-Menge } \Rightarrow \Gamma(M) \text{ determiniert.}$$

Andererseits folgt aus dem Auswahlaxiom, dass es nicht-determinierte Spiele gibt. In der axiomatischen Mengenlehre hat man daher das „Axiom der Determiniertheit“ (das die Determiniertheit von $\Gamma(M)$ für jede Menge $M \subseteq \{0,1\}^\omega$ fordert) als Alternative zum Auswahlaxiom studiert.

In den Spielen, die in der Informatik betrachtet werden, kommen vorwiegend Gewinnbedingungen auf der ersten und zweiten Ebene der Borel-Hierarchie vor. Die Gewinnbedingungen sind motiviert durch Akzeptierbedingungen der ω-Automatentheorie. Diese wurde von Büchi, McNaughton, Rabin, u.a. ab etwa 1960 entwickelt.

Akzeptierbedingungen beziehen sich hier auf die Zustandsfolgen, die ein ω-Automat auf einem gegebenen Eingabewort annimmt.

Betrachtet man den Graphen eines Automaten als Spielgraphen, teilt also die Zustände auf beide Spieler auf, so kann man den Gewinn einer Partie durch die Akzeptierbedingung des Automaten definieren. Entsprechend kann auch die Gewinnbedingung für einen Spielgraphen als Akzeptierbedingung eines Automaten aufgefasst werden.

Die automatentheoretischen Gewinnbedingungen lassen sich in zwei Arten unterteilen. Die erste stellt Bedingungen an die Menge $Occ(\pi) = \{v \in V \mid \pi \in V^*vV^\omega\}$ der in einer Partie π vorkommenden Knoten. Die zweite Art von Gewinnbedingungen stellt Bedingungen an die Menge $Inf(\pi) = \{v \in V \mid \pi \in (V^*v)^\omega\}$ der in einer Partie π unendlich oft vorkommenden Knoten.

Um die Verbindung zur klassischen deskriptiven Mengenlehre zu schlagen, betrachten wir einen Baum, der entsteht, wenn man einen (bipartiten) Spielgraphen von einem Knoten q_0 aus abgewickelt. Dann definiert eine Gewinnbedingung über $Occ(\pi)$ eine Menge von Pfaden, die zur Booleschen Hülle der abgeschlossenen Mengen (Π_1-Mengen der Borel-Hierarchie) gehört. Eine Gewinnbedingung über $Inf(\pi)$ bestimmt eine Menge von Pfaden, die zur Booleschen Hülle der Π_2-Mengen der Borel-Hierarchie gehört.

Unter den Bedingungen an das Vorkommen $Occ(\pi)$ von Knoten fallen als Spezialfälle folgende Gewinnbedingungen:

- Garantiebedingung: Für eine gegebene Knotenmenge $F \subseteq V$ gilt: Spieler 0 gewinnt die Partie π, wenn $Occ(\pi) \cap F \neq \emptyset$.
- Sicherheitsbedingung: Für eine gegebene Knotenmenge $F \subseteq V$ gilt: Spieler 0 gewinnt die Partie π, wenn $Occ(\pi) \subseteq F$.

Für jede Partie π und jede Menge $F \subseteq V$ gilt, dass sie die Garantiebedingung für F genau dann erfüllt, wenn sie die Sicherheitsbedingung für $V \setminus F$ nicht erfüllt.

Unter den Bedingungen an unendlich häufiges Vorkommen $Inf(\pi)$ von Knoten finden sich folgende besondere Fälle:

- Büchi-Bedingung: Für eine gegebene Knotenmenge $F \subseteq V$ gilt: Spieler 0 gewinnt die Partie π, wenn $Inf(\pi) \cap F \neq \emptyset$.
- Paritätsbedingung: Für eine gegebene Färbung der Knoten $c : V \to [|V| + 1]$, gilt:[1] Spieler 0 gewinnt π, wenn $\max c(Inf(\pi))$ gerade ist. (Dabei sei $[n] = \{i \in \mathbb{N} \mid 0 \leq i < n\}$.)

[1] Eine übliche Variante der Färbung ist den Knoten, denen hier die Farbe 0 zugeordnet wird,

- Streett-Bedingung: Für eine gegebene Menge von Paaren von Knotenmengen $(L_0, U_0), \ldots, (L_{r-1}, U_{r-1})$ mit $L_j, U_j \subseteq V$ gilt: Spieler 0 gewinnt π, wenn

$$\forall k \in [r] : \mathit{Inf}(\pi) \cap U_k \neq \emptyset \implies \mathit{Inf}(\pi) \cap L_k \neq \emptyset.$$

- Rabin-Bedingung: Für eine gegebene Menge von Paaren von Knotenmengen $(L_0, U_0), \ldots, (L_{r-1}, U_{r-1})$ mit $L_j, U_j \subseteq V$ gilt: Spieler 0 gewinnt π, wenn

$$\exists k \in [r] : \mathit{Inf}(\pi) \cap U_k \neq \emptyset \ \wedge \ \mathit{Inf}(\pi) \cap L_k = \emptyset.$$

- Muller-Bedingung: Für eine gegebene Menge von Knotenmengen $\mathcal{F} \subseteq 2^V$ gilt: Spieler 0 gewinnt π, wenn $\mathit{Inf}(\pi) \in \mathcal{F}$.

Wie schon Garantie- und Sicherheitsbedingungen, so weisen auch Rabin- und Streettbedingungen komplementäres Verhalten auf: Für jede Partie π und eine gegebene Menge von Paaren von Knotenmengen $(L_0, U_0), \ldots, (L_{r-1}, U_{r-1})$ mit $L_j, U_j \subseteq V$ gilt, dass π die Rabin-Bedingung genau dann erfüllt, wenn π die Streett-Bedingung nicht erfüllt.

Für eine detaillierte Diskussion dieser Gewinnbedingungen und ihrer Beziehungen sei auf [Tho97] verwiesen.

2.1.2 Verbindung zu logisch definierten Gewinnbedingungen

Die Motivation für die heute studierten automatentheoretischen Gewinnbedingungen stammt aus der Theorie von Büchi, McNaughton und Rabin, in der Automaten und monadische Logik verknüpft werden.

Automaten über unendlichen Eingabeworten verwendete erstmals Büchi in [Büc62], um die Entscheidbarkeit der monadischen Logik zweiter Stufe mit einem Nachfolger (S1S) zu zeigen. Wir setzen die übliche Terminologie über Automaten und Logiken voraus, wie sie etwa in [Tho97] bereitgestellt wird. Insbesondere betrachtet man in diesem Kontext statt der obengenannten Gewinnbedingungen entsprechende Akzeptierbedingungen von ω-Automaten; in dieser Weise erhält man für ω-Sprachen die Begriffe: ‚Büchi-erkennbar', ‚Muller-erkennbar', ‚Streett-erkennbar', ‚Rabin-erkennbar' und ‚Parität-erkennbar'.

Die Äquivalenz von ω-Automaten zu Logiken beschreibt der folgende Satz:

keine Farbe zuzuordnen. Man erhält eine partielle Färbungsfunktion $V \dashrightarrow \{1, \ldots, |V|\}$. Legt man wie üblich fest, dass Spieler 0 die Partie gewinnt, wenn in ihr keine Farbe unendlich oft vorkommt ($c(\mathit{Inf}(\pi)) = \emptyset$), so erhält man dieselbe Gewinnbedingung.

Satz 2.1 (Büchi, McNaughton, Rabin) *Für jede ω-Sprache L gilt:*

$$\begin{aligned} & L \text{ ist S1S-definierbar} \\ \iff\ & L \text{ ist nicht-deterministisch Büchi-erkennbar} \\ \iff\ & L \text{ ist deterministisch Muller-erkennbar} \\ \iff\ & L \text{ ist deterministisch Streett-erkennbar} \\ \iff\ & L \text{ ist deterministisch Rabin-erkennbar} \\ \iff\ & L \text{ ist deterministisch Parität-erkennbar} \end{aligned}$$

Die Bedeutung dieses Satzes liegt darin, dass er die Konstruktion von Gewinnstrategien in beliebigen Sprachen über endlichen Graphen ermöglicht, in denen die Gewinnbedingung durch eine S1S-Formel beschrieben wird. Diese Formeln umfassen insbesondere die Formeln der propositionalen temporalen Logik LTL (vermöge einer Übersetzung von LTL nach S1S [Tho90]). Aufgrund von Satz 2.1 gilt nämlich:

Satz 2.2 [**Tho00**] *Sei φ eine in LTL oder S1S formulierte Gewinnbedingung für Spieler 0 über dem Spielgraphen $G = (Q, E)$ (mit atomaren Formeln $q \in Q$), und sei S die Zustandsmenge eines zu φ äquivalenten deterministischen ω-Automaten mit Muller- bzw. Paritätsbedingung; über $Q \times S$ werde die Kantenrelation E' durch Produktbildung aus E und der Transitionsrelation des Automaten definiert.*

Dann gilt für den endlichen Spielgraphen $G' = (Q \times S, E')$ mit Muller- bzw. Paritäts-Gewinnbedingung: Spieler 0 gewinnt von q aus das Spiel über G mit Gewinnbedingung φ $\iff$ Spieler 0 gewinnt von (q, s_0) aus das Spiel über G' mit Muller- bzw. Paritäts-Gewinnbedingung.

Wie in der ω-Automatentheorie gezeigt wird, wird diese Reduktion von LTL-Formeln auf Paritätsspiele mit einer hohen Zustandskomplexität erkauft: Die bekannten Konstruktionen transformieren LTL-Formeln der Länge n in nicht-deterministische Büchi-Automaten mit 2^n Zuständen, Büchi-Automaten mit n Zuständen in deterministische Muller-Automaten mit $2^{O(n \log n)}$ Zuständen und deterministische Muller-Automaten mit n Zuständen in deterministische Paritätsautomaten mit $2^{O(n \log n)}$ Zuständen. Im Folgenden gehen wir von einer Präsentation von Spielen mit Paritätsgewinnbedingung aus.

2.2 Paritätsspiele und Ansätze zur Strategiekonstruktion

In diesem Abschnitt werden zunächst übliche Varianten des Paritätsspiels angegeben und Eigenschaften von Gewinnstrategien in Paritätsspielen formuliert. Danach werden zwei Verfahren zur Konstruktion von Strategien für Paritätsspiele vorgestellt. Das erste, von McNaughton entwickelte Verfahren ist in [McN93,

Tho95] beschrieben. Es basiert darauf, rekursiv die Strategiekonstruktion für Teilspiele vorzunehmen und die Ergebnisse geeignet zusammenzusetzen. Das zweite, von Jurdziński entwickelte Verfahren ist in [Jur00b] beschrieben und verwendet Fortschrittsmaße, wie sie von Klarlund und Kozen in [KK91] eingeführt werden. Wenn es möglich ist, ein Fortschrittsmaß für einen Spieler auf einem Paritätsspielgraphen zu konstruieren, so lassen sich daraus eine Gewinnstrategie für ihn und die Gewinnbereiche beider Spieler bestimmen.

2.2.1 Varianten des Paritätsspiels

Im Folgenden sollen gebräuchliche Varianten in der Definition von Paritätsspielen beschrieben werden.

Knoten ohne ausgehende Kanten. Die Anforderung an Spielgraphen, die festlegt, dass jeder Knoten mindestens eine ausgehende Kante haben muss, kann aufgehoben werden. Dazu legt man fest, dass Partien, die in einem Knoten ohne ausgehende Kanten eines Spielers enden, für diesen Spieler verloren sind.

Diese Art von Spiel lässt sich in die ursprüngliche transformieren, indem man zwei *Senken* genannte Knoten hinzufügt. Beide Senken besitzen jeweils nur eine ausgehende Kante zu sich selbst. Die Senken sind so gefärbt, dass in einer Spieler 0 gewinnt und in der anderen Spieler 1. Weiterhin fügt man von jedem Knoten aus, der keine ausgehenden Kanten hat, eine Kante ein, die zur Senke mit gegnerischem Gewinn führt. Damit erhält man auf dem neuen Spielgraphen zu den endlichen Partien des ursprünglichen Graphen korrespondierende Partien, die unendlich sind, jeweils schließlich in einer der beiden Senken bleiben und von demselben Spieler gewonnen werden.

Rabinkettenbedingung In dieser Variante wird die Gewinnbedingung als eine spezielle Rabin-Gewinnbedingung formuliert. Sie besteht aus einer Menge von Paaren $\{(E_0, F_0), \dots (E_{r-1}, F_{r-1})\}$ mit $E_0 \subseteq F_0 \subseteq E_1 \subseteq F_1 \subseteq \dots \subseteq E_{r-1} \subseteq F_{r-1} \subseteq V$. Spieler 0 gewinnt eine Partie, wenn es ein $i \in [r]$ gibt, so dass die unendlich oft vorkommenden Knoten nicht alle in E_i, aber alle in F_i enthalten sind.

Dieselbe Gewinnbedingung erhält man mit der Färbung $c : V \to [2r]$, wobei für alle $v \in E_0 : c(v) = 2r - 1$ und für alle $v \in F_i \setminus E_i : c(v) = 2r - 2 - 2i$ für alle $i \in [r]$ und $v \in E_{i+1} \setminus F_i : c(v) = 2r - 3 - 2i$ für alle $i \in [r-1]$. Umgekehrt erhält man aus einer gegebenen Färbung $c : V \to [2d]$ folgende Paare mit

$$E_i = \{v \in V | c(v) \geq 2d - 1 - 2i\} \text{ und } F_i = \{v \in V | c(v) \geq 2d - 2 - 2i\}$$

für $i \in [d]$.

Relevanzordnung. Bei dieser Variante wird an Stelle der Färbung der Knoten eine totale Ordnung (die *Relevanzordnung*) der Knoten verwendet, bei der jedes Knotenpaar vergleichbar ist und, wenn beide Knoten verschiedenen Spielern gehören, einer der Knoten echt relevanter als der andere ist. Der Gewinner einer Partie ergibt sich aus einem bezüglich der Relevanz maximalen unter den unendlich oft erreichten Knoten. Der Spieler, dem dieser Knoten gehört, ist Gewinner der Partie. Diese Definition ist eindeutig, da Knoten, die nach der Relevanzordnung äquivalent sind, alle demselben Spieler gehören.

Aus einer gegebenen Relevanzordnung lässt sich leicht eine Färbung konstruieren, die dieselbe Gewinnbedingung liefert. Um umgekehrt aus einer Färbung eine Relevanzordnung zu konstruieren, kann es notwendig sein, Hilfsknoten einzufügen. Hat ein Knoten p eine gerade Farbe und gehört aber Spieler 1, so wird in dem neuen Spielgraph ein Hilfsknoten mit p' für Spieler 1 eingeführt. Von diesem Knoten gehen alle Kanten aus, die im ursprünglichen Graphen von p ausgingen. Im neuen Graphen gehört der Knoten p Spieler 0 und besitzt genau eine ausgehende Kante zu p'. Entsprechend wird bei einem Knoten p mit ungerader Farbe von Spieler 0 ein zusätzlicher Knoten p' für Spieler 0 eingeführt und p gehört im neuen Graphen Spieler 1. Entsprechend wird auch mit den Kanten verfahren. Die Relevanzordnung ist genau die $<$-Ordnung auf den Farben. Die zusätzlichen Hilfsknoten erhalten dabei die kleinste Relevanz (entsprechend der Farbe 0). Lässt man die Hilfsknoten außer acht, so können auf beiden Graphen dieselben Partien gespielt werden und sie werden jeweils von denselben Spielern gewonnen.

2.2.2 Eigenschaften des Paritätsspiels

Im Folgenden werden grundlegende Eigenschaften von Paritätsspielen angegeben.

Uniforme Strategien

Durch Emerson, Jutla und Mostowski ([EJ91, Mos84]) wurde gezeigt, dass ein Spiel von einem Knoten aus determiniert ist und es eine positionale Gewinnstrategie gibt. Betrachtet man alle Knoten als mögliche Anfangsknoten von Partien, so folgt, dass sich die Knoten des Spielgraphen in die zwei Gewinnbereiche der beiden Spieler aufteilen, und für jeden der Knoten existiert eine positionale Gewinnstrategie für einen der beiden Spieler. Im Folgenden zeigen wir, dass jeweils eine uniforme Gewinnstrategie auf den beiden Gewinnbereichen existiert.

Eine einfache aber wichtige Eigenschaft von Paritätsspielen ergibt sich unmittelbar daraus, dass die Gewinnbedingung nur von den unendlich oft vorkommenden Zuständen abhängt:

Bemerkung 2.3 *Sei ein Spielgraph (V, E) mit einer Aufteilung der Knoten $V = V_0 \dot\cup V_1$ auf beide Spieler und einer Paritätsgewinnbedingung gegeben. Dann gilt*

für zwei Partien $\pi_1, \pi_2 \in V^\omega$ mit $\pi_1 = \pi_1'\pi$ und $\pi_2 = \pi_2'\pi$, wobei $\pi_1', \pi_2' \in V^*$ und $\pi \in V^\omega$, dass sie vom gleichen Spieler gewonnen werden.

Das bedeutet, dass zwei Partien, die sich nur in einem endlichen Anfangsstück unterscheiden, denselben Gewinner haben.

Ist R eine Menge von Knoten, die in einer Partie besucht werden, die nach einer positionalen Strategie und von einem Knoten v aus gespielt ist, dann ist die Strategie offensichtlich eine uniforme Strategie auf R. Der folgende Hilfssatz sagt, dass, wenn diese Strategie eine Gewinnstrategie von v aus ist, sie auch eine uniforme Gewinnstrategie auf R ist.

Hilfssatz 2.4 *Sei ein Spielgraph (V, E) mit einer Aufteilung der Knoten $V = V_0 \dot\cup V_1$ auf beide Spieler und einer Paritätsgewinnbedingung gegeben und eine Gewinnstrategie $\rho : V_i \to V$ für Spieler i ($i \in [2]$) von einem Knoten $v \in V$ aus. Sei $R \subseteq V$ die Menge der Knoten, die in von v ausgehenden nach ρ gespielten Partien erreicht werden können. Dann ist ρ eine Gewinnstrategie für Spieler i auf der Knotenmenge R.*

Beweis: Sei ein Spielgraph (V, E) mit einer Aufteilung der Knoten $V = V_0 \dot\cup V_1$ auf beide Spieler und einer Paritätsgewinnbedingung gegeben und eine Gewinnstrategie $\rho : V_i \to V$ für Spieler i ($i \in [2]$) von einem Knoten $v \in V$ aus. Sei $R \subseteq V$ die Menge der Knoten, die in von v ausgehenden Partien erreicht werden können. Wir betrachten eine beliebige Partie von einem beliebigen Knoten in R aus, in der Spieler i nach ρ spielt, und zeigen, dass Spieler i diese Partie gewinnt.

Sei $v' \in R$ gegeben und $\pi' \in V^\omega$ eine von v' aus nach der Strategie ρ gespielte Partie. Es ist zu zeigen, dass Spieler i diese Partie π' gewinnt.

Da $v' \in R$ ist, existiert eine von v aus nach ρ gespielte Partie $\pi \in V^\omega$, in der v' erreicht wird; d.h. es existieren $\pi_0 \in V^*$ und $\pi_1 \in V^\omega$ mit $\pi = \pi_0 v \pi_1$. Somit ist auch die zusammengesetzte Partie $\pi_0\pi'$ eine Partie von v aus, die nach der Gewinnstrategie ρ gespielt ist. Da sich die Partien π' und $\pi_0\pi'$ nur in einem endlichen Anfangsstück unterscheiden, haben sie nach Bemerkung 2.3 denselben Gewinner. Die von v ausgehende Partie gewinnt Spieler i nach Voraussetzung. Also gewinnt er auch die von v' ausgehende Partie π'. □

Hilfssatz 2.5 *Sei ein Spielgraph (V, E) mit einer Aufteilung der Knoten $V = V_0 \dot\cup V_1$ auf beide Spieler und einer Paritätsgewinnbedingung gegeben und eine Gewinnstrategie $\rho : V_i \to V$ für Spieler i ($i \in [2]$) für Partien, die von Knoten aus $R \subseteq V$ ausgehen. Sei $\rho' : V_i \to V$ eine Gewinnstrategie für Spieler i von Knoten $v' \in V$ aus. Dann existiert eine uniforme Gewinnstrategie $\rho'' : V_i \to V$ für Partien, die von Knoten aus $R \cup \{v'\}$ ausgehen, und für diese Strategie gilt:*

$$\rho''(v) = \rho(v) \text{ für alle } v \in R \cap V_i$$

Die bereits zum Beweis des vorangehenden Hilfssatzes verwendete Methode lässt sich auch für diesen Hilfssatz verwenden.

Beweis: Sei ein Spielgraph (V, E) mit einer Aufteilung der Knoten $V = V_0 \dot\cup V_1$ auf beide Spieler und einer Paritätsgewinnbedingung gegeben und eine Gewinnstrategie $\rho : V_i \to V$ für Spieler i ($i \in [2]$) für Partien, die von Knoten aus $R \subseteq V$ ausgehen. Sei $\rho' : V_i \to V$ eine Gewinnstrategie für Spieler i von Knoten $v' \in V$ aus. Sei $R' \subseteq V$ die Menge der Knoten, die in Partien, die von R ausgehen, erreicht werden können. Wähle

$$\rho'' = \rho|_{R'} \cup \rho'|_{V_I \setminus R'}.$$

Damit ist

$$\rho''(v) = \rho(v) \text{ für alle } v \in R \cap V_i$$

erfüllt, da nach Definition $R' \supseteq R$.

Nach Hilfssatz 2.4 gewinnt Spieler i, wenn er nach ρ'' spielt, alle Partien, die von R' ausgehen, und somit die, die von R ausgehen, da $R \subseteq R'$ gilt. Partien, in denen Spieler i nach ρ'' spielt und die in v' beginnen, verlaufen, wenn sie einen Knoten aus R' enthalten, ab diesem Knoten nur noch in R'. Da diese Partien sich nur in einem endlichen Anfangsstück von in R' beginnenden Partien unterscheiden, werden sie ebenfalls von Spieler i gewonnen.

Es bleiben Partien zu betrachten, die von v' aus nach ρ'' gespielt werden aber nie in R' führen. Auf den Knoten $V \setminus R'$ ist die Strategie ρ'' nach Definition identisch mit ρ'. Da Spieler i von v' aus nach der Gewinnstrategie ρ' gespielte Partien gewinnt, gewinnt er somit auch Partien, die von v' aus nach ρ'' gespielt werden, aber nie in R' führen. □

Lemma 2.6 *Sei ein Spielgraph (V, E) mit einer Aufteilung der Knoten $V = V_0 \dot\cup V_1$ auf beide Spieler und einer Paritätsgewinnbedingung gegeben. Sei $R \subseteq V$ eine Menge von Knoten für die Spieler i für jeden Knoten $v \in R$ eine Gewinnstrategie $\rho_v : V_i \to V$ besitzt. Dann existiert eine uniforme Gewinnstrategie ρ mit der Spieler i von Knoten aus R aus gewinnt.*

Beweis: Durch Induktion ergibt sich die Behauptung unmittelbar aus Hilfssatz 2.5. □

Duale Spiele

Zu jedem Paritätsspiel existiert ein ‚duales' Paritätsspiel. Diese Eigenschaft lässt sich in vielen Beweisen ausnutzen.

Sei ein Spielgraph (V, E) mit der Knotenmenge V aufgeteilt auf die Knoten V_0 von Spieler 0 und V_1 von Spieler 1 und eine Färbungsfunktion $c : V \to [|V| + 1]$ gegeben. Das *duale Spiel* wird auf demselben Spielgraphen gespielt. In ihm gehören die Knoten V_0 dem Spieler 1, die Knoten V_1 dem Spieler 0 und die Färbungsfunktion ist $\bar c : V \to [|V| + 2] : v \mapsto c(v) + 1$.

Bemerkung 2.7 *Sei ein Spielgraph (V, E) mit einer Aufteilung der Knoten auf beide Spieler und eine Paritätsgewinnbedingung gegeben. Sei $\pi \in V^\omega$ eine Partie auf diesem Spielgraphen. Dann gewinnt Spieler i diese Partie genau dann, wenn Spieler $1 - i$ sie im dualen Spiel gewinnt (wobei $i \in [2]$).*

Synthese von Gewinnstrategien aus Teilspielen

Die folgenden Hilfssätze über unendliche Spiele erlauben es, wenn bereits Lösungen für bestimmte Teilspiele eines gegebenen Spiels bekannt sind, daraus eine Lösung für das ganze Spiel zu bestimmen. Dieser Rückgriff auf Teilspiele mit bereits bekannten Strategien und Gewinnbereichen wird insbesondere für den in Kapitel 4 beschriebenen Zusammenhang der Paritätsspiele zum Model-Checking benötigt. Für diesen Unterabschnitt legen wir zur Vereinfachung die Variante des Paritätsspiels zugrunde, die auch Knoten ohne ausgehende Kanten zulässt.

Hilfssatz 2.8 *Sei $G = (V, E)$ ein Spielgraph mit Knotenaufteilung $V = V_0 \dot\cup V_1$ und $W_0 \subseteq V$ der Gewinnbereich von Spieler 0. Sei $V' \subseteq V$ mit $V' E \subseteq V'$. Dann ist $G' = (V', E|_{V' \times V'})$ ein Spielgraph mit dem Gewinnbereich $W_0 \cap V'$ für Spieler 0 und $W_1 \cap V'$ für Spieler 1.*

Beweis: Beide Spieler können auch auf den Knoten des Teilspiels die gleichen Nachfolgerknoten wählen wie im ursprünglichen Spiel. Damit kann jede Partie, die auf G von einem Knoten aus V' gespielt wird, auch auf G' gespielt werden. Somit lassen sich Gewinnbereiche und Strategien übertragen. □

Hilfssatz 2.9 *Sei $G = (V, E)$ ein Spielgraph mit Knotenaufteilung $V = V_0 \dot\cup V_1$ und $W_0 \subseteq V$ der Gewinnbereich von Spieler 0. Sei $v \in W_0$ und ein Spielgraph $G' = (V, E|_{(V \setminus \{v\}) \times V})$ mit der Knotenaufteilung $V = V_0' \dot\cup V_1'$ gegeben, wobei $V_0' = V_0 \setminus \{v\}$ und $V_1' = V_1 \cap \{v\}$. Dann ist der Gewinnbereich von Spieler 0 in G' ebenfalls W_0.*

Beweis: Eine Gewinnstrategie auf G kann auch auf G' verwendet werden, außer für v. In G' gewinnt Spieler 0 von Knoten v aus, da v keine Nachfolger hat und $v \in V_1'$. □

Bemerkung 2.10 *In einem Spiel ist die Zugehörigkeit der Knoten mit genau einer ausgehenden Kante zu V_0 oder V_1 unerheblich, da das Verhalten von diesen Knoten aus für beide Spieler gleich ist: Sie können nur den einzigen Nachfolger des Knotens wählen.*

Der folgende Hilfssatz beschreibt die Veränderungen der Gewinnbereiche, die dadurch entstehen, dass Knoten ohne ausgehende Kanten die Zugehörigkeit zu den Spielern ändern.

Hilfssatz 2.11 *Sei $G = (V, E)$ ein Spielgraph mit Knotenaufteilung $V = V_0 \dot\cup V_1$ und einer Paritätsbedingung und sei W_0 der Gewinnbereich von Spieler 0. Sei $S \subseteq V_1$ eine Knotenmenge mit $SE = \emptyset$. Sei $G' = (V, E)$ ein Spielgraph mit derselben Gewinnbedingung und mit der Knotenaufteilung $V = V_0' \dot\cup V_1'$, wobei $V_0' = V_0 \setminus S$ and $V_1' = V_1 \cup S$. Sei W_0' der Gewinnbereich von Spieler 0 in G'. Dann gilt: $W_0 \cup S \subseteq W_0'$.*

Beweis: Sei $G = (V, E)$ ein Spielgraph mit Knotenaufteilung $V = V_0 \dot\cup V_1$ und einer Paritätsbedingung und sei W_0 der Gewinnbereich von Spieler 0. Sei $S \subseteq V_1$ eine Knotenmenge mit $SE = \emptyset$. Sei $G' = (V, E)$ ein Spielgraph mit derselben Gewinnbedingung und mit der Knotenaufteilung $V = V_0' \dot\cup V_1'$, wobei $V_0' = V_0 \setminus S$ and $V_1' = V_1 \cup S$. Sei W_0' der Gewinnbereich von Spieler 0 in G'.

Da $SE = \emptyset$ und $S \subseteq V_1'$ folgt unmittelbar $S \subseteq W_0'$. Auf G und G' können die gleichen Partien gespielt werden. Zu den Partien, die Spieler 0 auf G gewinnen kann, kommen auf G' genau jene hinzu, die in S enden. Somit kann sich der Gewinnbereich von Spieler 0 lediglich vergrößern: $W_0 \subseteq W_0'$. □

Wir definieren die ‚Auffaltung' eines Spielgraphen. Mit Hilfe einer ‚Auffaltung' des Spielgraphen eines Paritätsspiels lässt sich eine Gewinnstrategie für dieses Spiel bestimmen, indem eine Folge von Spielgraphen über einem Teilgraphen konstruiert wird und für diese Gewinnstrategien bestimmt werden. Diese Gewinnstrategien lassen sich zu einer Gewinnstrategie für den ursprünglichen Spielgraphen zusammensetzen.

Sei $G = (V, E)$ ein Spielgraph mit Knotenaufteilung $V = V_0 \dot\cup V_1$ und eine Färbungsfunktion $c : V \to [|V| + 1]$ gegeben. Abkürzend setzen wir $n = |V|$, $c_{\max} = \max_{u \in V} c(u)$ und $V_{\max} = c^{-1}(c_{\max})$. Wir setzen $c_{\max}$ ungerade vorraus (Anderenfalls betrachten wir das duale Spiel). Sei $S = EV_{\max}$ die Knotenmenge der Vorgänger von Knoten mit maximaler Farbe, und sei der Graph so beschaffen, dass alle Knoten in S genau eine ausgehende Kante haben ($|vE| = 1$ für alle $v \in S$). Wir setzen $E_S = E \cap (S \times V_{\max})$ und $E' = E \setminus E_S$. Dann heißt das $n + 1$-Tupel von Spielgraphen

$$(G^{(0)}, \ldots, G^{(n)})$$

eine *Auffaltung von* G, wenn gilt: Für alle $i \in [n]$ ist $G^{(i)} = (V, E')$ mit derselben Gewinnbedingung wie G. Die Gewinnbereiche von $G^{(i)}$ werden mit $W_0^{(i)}$ und $W_1^{(i)}$ bezeichnet und $G^{(i)}$ hat folgende Knotenaufteilungen:

$$\begin{array}{llll} V_0^{(0)} = V_0 \cup S & \text{und} & V_1^{(0)} = V_1 \setminus S & \\ V_0^{(i)} = V_0^{(i-1)} \setminus E_S W_0^{(i-1)} & \text{und} & V_1^{(i)} = V_1^{(i-1)} \cup E_S W_0^{(i-1)} & \text{für } i = 1, \ldots, n. \end{array}$$

Zwischen einem Spielgraph und seiner Auffaltung besteht der folgende Zusammenhang:

Hilfssatz 2.12 *Seien ein Spielgraph $G = (V, E)$ mit Knotenaufteilung $V = V_0 \dot\cup V_1$, eine Färbungsfunktion $c : V \to [|V| + 1]$ und der Gewinnbereich W_0 von Spieler 0 auf G gegeben. Sei $n = |V|$. Sei $\max_{u \in V} c(u)$ ungerade. Sei $(G^{(0)}, \ldots, G^{(n)})$ eine Auffaltung von G und $W_0^{(n)}$ der Gewinnbereich von Spieler 0 auf $G^{(n)}$. Dann gilt:*

$$W_0 = W_0^{(n)}.$$

Ist die größte vorkommende Farbe gerade, so lässt sich der Satz auf das duale Spiel anwenden.

Beweis: Sei ein Spielgraph $G = (V, E)$ mit Knotenaufteilung $V = V_0 \dot\cup V_1$ und eine Färbungsfunktion $c : V \to [|V| + 1]$ gegeben. Sei $n = |V|$. Seien W_0, W_1 die Gewinnbereiche von G. Sei $c_{\max} = \max_{u \in V} c(u)$ und $V_{\max} = c^{-1}(c_{\max})$. Sei $c_{\max}$ ungerade. Sei $S = EV_{\max}$ und $|vE| = 1$ für alle $v \in S$.

Wir setzen $E_S = E \cap (S \times V_{\max})$ und $E' = E \setminus E_S$. Sei $(G^{(0)}, \ldots, G^{(n)})$ eine Auffaltung von G.

Seien $W_0^{(i)}, W_1^{(i)}$ die Gewinnbereiche und $\sigma^{(i)}, \tau^{(i)}$ Gewinnstrategien auf $G^{(i)}$ für $i = 0, \ldots, n$.

Um zu zeigen, dass $W_0 = W_0^{(n)}$ gilt, definieren wir Gewinnstrategien σ und τ auf G und zeigen, dass Spieler 0 mit σ auf $W_0^{(n)}$ und Spieler 1 mit τ auf $W_1^{(n)}$ gewinnt und damit, dass $W_0^{(n)}$ und $W_1^{(n)}$ die Gewinnbereiche von G sind.

Wir setzen

$$\tau : V_1 \cap W_1^{(n)} \to V : \tau(v) = \begin{cases} vE_S, & \text{falls } v \in S \\ \tau^{(n)}(v), & \text{sonst.} \end{cases}$$

Zu zeigen ist, dass Spieler 0 alle in $W_1^{(n)}$ beginnenden nach τ gespielten Partien gewinnt.

Wir betrachten zunächst Partien die nur innerhalb von $W_1^{(n)}$ gespielt werden. Werden irgendwann in der Partie keine Knoten von S mehr besucht, so wird mit τ auch nach $\tau^{(n)}$ gespielt und Spieler 1 gewinnt, da er die Restpartie (die Partie in $W_1^{(n)} \setminus S$) auf $G^{(n)}$ auch gewinnt. Anderenfalls müssen unendlich oft Knoten aus S in der Partie vorkommen, wodurch Spieler 1 ebenfalls gewinnt, da die eindeutigen Nachfolger von Knoten aus S die maximale, ungerade Farbe haben.

Zieht Spieler 1 nach der Stratgie $\tau^{(n)}$ so kann dies nicht aus $W_1^{(n)}$ herausführen, da $\tau^{(n)}$ Gewinnstrategie auf dem Gewinnbereich $W_1^{(n)}$ in $G^{(n)}$ ist. Es bleibt zu zeigen, dass auch die hinzugekommen Kanten E_S nicht aus $W_1^{(n)}$ herausführen.

Wir definieren zusätzlich:

$$V_0^{(n+1)} = V_0^{(n)} \setminus E_S W_0^{(n)}$$

Nach Definition der Knotenaufteilung der Auffaltung gilt:

$$V_0^{(0)} \supseteq \ldots \supseteq V_0^{(n)} \supseteq V_0^{(n+1)}$$

Da jede Knotenmenge nicht mehr als n Elemente enthalten kann, muss für ein $i \in [n+1]$ auch $V_0^{(i)} = V_0^{(i+1)}$ gelten. Daraus folgt auch $W_0^{(i)} = W_0^{(i+1)}$, falls $i \in [n]$. Da die Knotenaufteilung nur von den Gewinnbereichen abhängt folgt, falls $i \in [n]$, auch $V_0^{(i+1)} = V_0^{(i+2)}$. Durch Induktion ergibt sich: $V_0^{(n)} = V_0^{(n+1)}$. Durch Einsetzen der Definition von $V_0^{(n+1)}$ erhält man: $V_0^{(n)} = V_0^{(n)} \setminus E_S W_0^{(n)}$ und somit $V_0^{(n)} \cap E_S W_0^{(n)} = \emptyset$. Da $V = V_0^{(n)} \dot\cup V_1^{(n)}$, folgt $E_S W_0^{(n)} \subseteq V_1^{(n)}$. Nach Definition gilt $E_S W_0^{(n)} \subseteq S$ und daher auch: $E_S W_0^{(n)} \subseteq V_1^{(n)} \cap S$. Da in $G^{(n)}$ die Knoten aus S keine Nachfolger haben, gilt: $V_1^{(n)} \cap S \subseteq W_0^{(n)}$. Aus $E_S W_0^{(n)} \subseteq V_1^{(n)} \cap S$ und $V_1^{(n)} \cap S \subseteq W_0^{(n)}$ erhält man $E_S W_0^{(n)} \subseteq W_0^{(n)}$. Mit $E_S \subseteq V \times V$ und $V = W_0^{(n)} \dot\cup W_1^{(n)}$ folgt: $W_1^{(n)} E_S \subseteq W_1^{(n)}$, was bedeutet, dass in $W_1^{(n)}$ beginnende Partien auch durch die zusätzlichen Kanten E_S nicht aus $W_1^{(n)}$ herausführen. Somit ist τ eine Gewinnstrategie auf $W_1^{(n)}$.

Bevor wir eine Strategie für Spieler 0 angeben, definieren wir folgende Hilfsfunktion:

$$k : W_0^{(n)} \to [n+1] : v \mapsto \min\{i \in [n+1] | v \in W_0^{(i)}\}.$$

Da $V_0^{(0)} \supseteq \ldots \supseteq V_0^{(n)}$, folgt nach Hilfssatz 2.11: $W_0^{(0)} \subseteq \ldots \subseteq W_0^{(n)}$. Somit ist

$$W_0^{(0)}, W_0^{(1)} \setminus W_0^{(0)}, \ldots, W_0^{(n)} \setminus W_0^{(n-1)}$$

eine Partionierung von $W_0^{(n)}$, wodurch k wohldefiniert ist.

Als Strategie für Spieler 0 setzen wir

$$\sigma : V_0 \cap W_0^{(n)} \to V : \sigma(v) = \begin{cases} vE_S, & \text{falls } v \in S \\ \sigma^{k(v)}(v), & \text{sonst.} \end{cases}$$

Aus den folgenden drei Hilfsbehauptungen:

$$uEv \implies v \in W_0^{(n)} \wedge k(u) \geq k(v), \quad \text{für alle } u \in V_1 \cap W_0^{(n)} \setminus S, v \in V \tag{2.1}$$

$$\sigma(u) \in W_0^{(n)} \wedge k(u) \geq k(\sigma(u)), \quad \text{für alle } u \in V_0 \cap W_0^{(n)} \setminus S \tag{2.2}$$

$$uEv \implies v \in W_0^{(n)} \wedge k(u) > k(v), \quad \text{für alle } u \in W_0^{(n)} \cap S, v \in V \tag{2.3}$$

folgt, dass σ eine Gewinnstrategie auf $W_0^{(n)}$ ist:

Sei $\pi \in V^\omega$ eine Partie die in $W_0^{(n)}$ beginnt. Dann kann nach den drei Hilfsbehauptungen weder Spieler 0 mit σ noch Spieler 1 mit einer beliebigen Strategie $W_0^{(n)}$ verlassen. Außerdem fällt der k-Wert der Knoten der Partie monoton. Da es nur endlich viele verschiedene k-Werte gibt, haben bis auf ein endliches Anfangsstück alle Knoten der Partie denselben k-Wert. Sei $\pi' \in V^\omega$ eine ‚Restpartie' von π, deren Knoten alle denselben k-Wert k_1 haben. Die Partie π' verläuft somit innerhalb von $W_0^{(k_1)}$ und, falls $k_1 > 0$ auch außerhalb von $W_0^{(k_1-1)}$. Da Kanten aus E_S den k-Wert vermindern, verläuft π' auch außerhalb von S. Somit ist π' auch eine auf $G^{(k_1)}$ gespielte Partie bei der Spieler 0 nach $\sigma^{(k_1)}$ spielt. Da π' in

$W_0^{(k_1)}$ beginnt, gewinnt Spieler 0 die Partie π' auf $G^{(k_1)}$ und wegen der gleichen Gewinnbedingung auch auf G. Da sich die Partie π' und π nur in einem endlichen Anfangsstück unterscheiden gewinnt Spieler 0 auch π auf G.

Es bleiben die drei Hilfsbehauptungen zu zeigen:

Zu (2.1): Sei $u \in V_1 \cap W_0^{(n)} \setminus S$ und $v \in V$ mit uEv. Da $u \notin S$ hat u in G die gleichen Nachfolger wie in $G^{(k(u))}$. Nach Definition von k gilt $u \in W_0^{(k(u))}$. Da $W_0^{(k(u))}$ der Gewinnbereich von Spieler 0 auf $G^{(k(u))}$ ist, kann Spieler 1 nicht aus $W_0^{(k(u))}$ herausziehen. Somit gilt: $v \in W_0^{(k(u))}$ und auch $v \in W_0^{(n)}$, weil $W_0^{(k(u))} \subseteq W_0^{(n)}$. Nach Definition von k gilt für alle $j \in [n]$ und $x \in W_0^{(j)} : j \geq k(x)$. Damit folgt aus $v \in W_0^{(k(u))}$ auch $k(u) \geq k(v)$.

Zu (2.2): Sei $u \in V_0 \cap W_0^{(n)} \setminus S$. Nach Definition von σ gilt für $u \in V_0 \cap W_0^{(n)} \setminus S : \sigma(u) = \sigma^{(k(u))}(u)$. Da in $G^{(k(u))}$ die Gewinnstrategie $\sigma^{(k(u))}$ nicht aus dem Gewinnbereich $W_0^{(k(u))}$ herausführt gilt

$$u \in W_0^{(k(u))} \Longrightarrow \sigma(u) \in W_0^{(k(u))}.$$

Daraus folgt $\sigma(u) \in W_0^{(n)}$, weil $W_0^{(k(u))} \subseteq W_0^{(n)}$. Und es folgt $k(u) \geq k(\sigma(u))$, weil nach Definition von k für alle $j \in [n]$ und $x \in W_0^{(j)} : j \geq k(x)$ gilt und damit insbesondere aus $\sigma(u) \in W_0^{(k(u))}$ auch $k(u) \geq k(\sigma(u))$ folgt.

Zu (2.3): Sei $u \in W_0^{(n)} \cap S$ und $v \in V$ mit uEv. Nach Definition von $V_0^{(0)}$ gilt $S \subseteq V_0^{(0)}$, woraus $S \subseteq W_1^{(0)}$ folgt. Daher gilt mit $u \in S$ auch $u \notin W_0^{(0)}$ und somit $k(u) > 0$. Nach Definition von k gilt: $u \in W_0^{(k(u))}$. Da in $G^{(k(u))}$ der Knoten u keine Nachfolger hat, folgt aus $u \in W_0^{(k(u))}$ schon $u \in V_1^{(k(u))}$. Nach Definition gilt:

$$V_1^{(k(u))} = V_1^{(k(u)-1)} \cup E_S W_0^{(k(u)-1)}.$$

Nach Definition von k gilt $u \notin W_0^{(k(u)-1)}$. Weil u keine Nachfolger in $G^{(k(u))}$ hat, folgt damit $u \notin V_1^{(k(u)-1)}$. Damit folgt aus $u \in V_1^{(k(u)-1)} \cup E_S W_0^{(k(u)-1)}$ schon $u \in E_S W_0^{(k(u)-1)}$. Aus $u \in E_S W_0^{(k(u)-1)}$ folgt mit $E_S = E \cap (S \times V)$ und uEv auch $v \in W_0^{(k(u)-1)}$. Daraus folgt $v \in W_0^{(n)}$, weil $W_0^{(k(u)-1)} \subseteq W_0^{(n)}$. Außerdem folgt aus $v \in W_0^{(k(u)-1)}$ nach Definiton von k auch $k(u) > k(v)$. □

2.2.3 Strategiekonstruktion nach McNaughton

Die Strategiekonstruktion von McNaughton ([McN93]) erlaubt es, eine Gewinnstrategie durch Rekursion auf kleinere Spielgraphen zu berechnen.

Wir definieren zunächst den für den Algorithmus grundlegenden Begriff des ‚Attraktors'. Anschließend geben wir McNaughtons Algorithmus an und zeigen seine Funktionsweise an einem Beispiel.

Attraktor

Sei ein Spielgraph (V, E) mit einer Aufteilung der Knoten $V = V_0 \dot\cup V_1$ auf beide Spieler gegeben. Wir definieren den *Attraktor* $Attr_i(U)$ einer Menge $U \subseteq V$ für Spieler $i \in [2]$ durch:

$$\begin{aligned} Attr_i^{(0)}(U) &= U \\ Attr_i^{(k+1)}(U) &= Attr_i^{(k)}(U) \\ &\quad \cup (EAttr_i^{(k)}(U) \cap V_i) \\ &\quad \cup \big(V \setminus E(V \setminus Attr_i^{(k)}(U))\big) \text{ für } k \in \mathbb{N} \\ Attr_i(U) &= \bigcup_{k\in\mathbb{N}} Attr_i^{(k)}(U) \setminus U. \end{aligned}$$

Dabei bezeichnet $Attr_i^{(k)}(U)$ die Menge der Knoten, von denen aus Spieler i das Erreichen von U nach höchstens k Schritten garantieren kann. Ein Knoten ist in $Attr_i^{(k+1)}(U)$, wenn er in $Attr_i^{(k)}(U)$ ist, oder wenn Spieler i der Knoten gehört und er $Attr_i^{(k)}(U)$ über eine Kante erreichen kann, oder wenn alle Kanten in $Attr_i^{(k)}(U)$ führen (was hier dadurch formuliert ist, dass keine Kanten außerhalb von $Attr_i^{(k)}(U)$ führen. Wir nutzen aus, dass jeder Knoten mindestens eine ausgehende Kante besitzt).

Der Attraktor $Attr_i(U)$ ist die Menge der Knoten außerhalb von U, von denen aus Spieler i das Erreichen von U nach endlich vielen Schritten garantieren kann. Wir sagen, dass Spieler i die Menge U von den Knoten in $Attr_i(U)$ aus *erzwingen* kann. Als *Attraktorstrategie* auf dem Attraktor $Attr_i(U)$ bezeichnen wir eine Strategie für Spieler i, die U erzwingt.

Algorithmus

Wir betrachten den Spielgraphen eines Paritätsspiels. Im einfachsten Fall enthält der Spielgraph nur Knoten einer Farbe. In diesem Fall sind alle Knoten Gewinnknoten des Spielers, der mit dieser Farbe gewinnt.

Enthält der Spielgraph Knoten mit verschiedenen Farben, so wählt man einen Knoten p mit maximaler Farbe aus und berechnet den Attraktor dieses Knotens für den Spieler, der mit dieser Farbe gewinnt. Für das kleinere Teilspiel, welches dadurch entsteht, dass man alle Knoten des Attraktors einschließlich p entfernt, lassen sich die Gewinnbereiche und -strategien bereits berechnen. Ist für den Spieler, der mit der Farbe von p gewinnt, von p aus das Erreichen seines Gewinnbereichs auf dem Teilspiel oder des Attraktors von p erzwingbar, so gewinnt dieser Spieler auch auf dem Attraktor von p, indem er eine Strategie benutzt, die zu p hinführt. Kann er den Gewinnbereich seines Teilspiels oder den des Attraktors von p jedoch nicht erreichen, so berechnet man den Attraktor des Gegners vom Gewinnbereich des Gegners. Der Gegner gewinnt nicht nur im Teilspiel auf seinem Gewinnbereich, sondern auch im vollständigen Spiel, und dort zusätzlich

noch auf dem für ihn berechneten Attraktor. Somit entfernt man diesen Gewinnbereich und Attraktor aus dem ursprünglichen Spielgraphen, so dass ein Teilspiel entsteht, für welches sich wiederum die Gewinnbereiche und -strategien berechnen lassen.

In der folgenden detaillierteren Beschreibung des Algorithmus wird immer davon ausgegangen, dass der Knoten p eine Farbe hat, mit der Spieler 0 gewinnt. Ist dies nicht der Fall, so wird der Algorithmus jeweils auf das duale Spiel angewendet und am Ende Gewinnstrategien und -bereiche vertauscht.

Die Strategiekonstruktion erfolgt rekursiv, indem zur Berechnung der Strategien auf dem ganzen Graphen auf schon konstruierte Strategien für Teilspiele zurückgegriffen wird. Wir bezeichnen ein Spiel als Teilspiel eines gegebenen Spiels mit Spielgraph (V, E) und einer Färbung $c : V \to [|V| + 1]$, wenn eine Menge $V' \subseteq V$ existiert, so dass $(V', E \cap (V' \times V'))$ der Spielgraph und $c|_{V'}$ die Färbung des Teilspiels ist. Insbesondere gilt auch für das Teilspiel, dass jeder Knoten mindestens eine ausgehende Kante besitzt.

Eine Strategie für einen Spielgraphen $G = (V, E)$ mit Knotenaufteilung $V = V_0 \dot\cup V_1$ lässt sich rekursiv durch die Funktion in Abb. 2.3 bestimmen. Grundlage der Strategieberechnung ist die Berechnung von Attraktoren und Attraktorstrategien. Die Funktion $attract(Q, E, S, T)$ in Abb. 2.2 bestimmt auf dem durch die Kantenmenge E gegebenen Graphen, alle Knoten in S, von welchen eine Zugfolge zu T durch den Spieler erzwungen werden kann, dem die Knoten in Q gehören. Dabei bilden S und T stets eine Partition der Knotenmenge V. Das bedeutet für Knoten aus Q reicht ein Nachfolger, von dem aus das Erreichen von T erzwungen werden kann, während bei allen anderen Knoten von allen Nachfolgern T erzwingbar sein muss.

Die Hauptfunktion $strategy(V_0, V_1, E, c)$ in Abbildung 2.3 liefert ein Tripel (W_0, W_1, σ), in welchem W_0 der Gewinnbereich von Spieler 0 ist, W_1 der Gewinnbereich von Spieler 1 und σ die beiden Strategiefunktionen beider Spieler vereinigt. Die Strategie für Spieler 0 ergibt sich daraus als $\sigma|_{V_0 \cap W_0}$ und entsprechend für Spieler 1 als $\sigma|_{V_1 \cap W_1}$. Die Funktion unterscheidet drei Fälle:

1. Als Rekursionsanfang wird der triviale Fall des Spielgraphen ohne Knoten berücksichtigt. Offensichtlich sind die Gewinnbereiche leer und die Strategiefunktion ist die Funktion mit leerem Definitionsbereich. (Zeilen 1 und 2 in Abb. 2.3)

2. Ist die größte vorkommende Farbe ungerade, wird das Problem für das duale Spiel gelöst. Bei der Rückgabe werden die Gewinnbereich vertauscht, um die Lösung für das ursprüngliche Spiel zu erhalten. (Zeilen 3 bis 5 in Abb. 2.3)

3. Anderenfalls wird ein Knoten, der im Folgenden auch als Pivotelement bezeichnet wird, aus der Menge der maximalen Knoten mit maximaler Farbe

```
attract(Q, E, S, T) :
 1. R := ∅;  ρ := ∅
 2. if T = ∅ then
 3.     return (R, ρ)
 4. else
 5.     for each q ∈ T do
 6.         for each p ∈ Eq ∩ S do
 7.             if p ∈ Q then
 8.                 R := R ∪ {p}
 9.                 ρ := ρ ∪ (p ↦ q)
10.             else
11.                 t := true
12.                 for each q' ∈ pE do
13.                     if q' ∈ S then
14.                         t := false
15.                 if t then
16.                     R := R ∪ {p}
17.     (R', ρ') := attract(Q, E, S \ R, R)
18.     return (R ∪ R', ρ ∪ ρ')
```

Abbildung 2.2: Funktion zum Berechnen eines Attraktors

bestimmt. Von diesem Pivotelement wird der Attraktor R und eine Attraktorstrategie ρ für Spieler 0 bestimmt. Auf das Teilspiel über der Knotenmenge $\bar{V} := V_0 \cup V_1 \setminus R \setminus \{p\}$ wird die Funktion *strategy* rekursiv angewandt. Die Gewinnbereiche dieses Teilspiels sind W_0 für Spieler 0 und W_1 für Spieler 1 und die Strategiefunktion σ. Kann Spieler 0 von dem Pivotelement aus einen Knoten außerhalb von W_1 erzwingen, so gewinnt Spieler 0 auch auf allen Knoten des Attraktors. Damit sind die Gewinnbereiche $W_0 \cup \{p\} \cup R$ für Spieler 0 und W_1 für Spieler 1. Die Strategiefunktion ist $\rho \cup \sigma$, falls $p \in V_1$, und sonst $\rho \cup \{(p \mapsto q)\} \cup \sigma$, wobei q der gewählte Nachfolger des Pivotelements ist.

Tritt der Fall ein, dass Spieler 0 von dem Pivotelement aus keinen Knoten außerhalb von W_1 erzwingen kann, so wird von W_1 der Attraktor R' und eine Attraktorstrategie σ' berechnet. Dann wird rekursiv das Teilspiel über der Knotenmenge $\bar{V} := V_0 \cup V_1 \setminus R' \setminus W_1$ berechnet. Die Gewinnbereiche dieses Teilspiels sind U_0 für Spieler 0 und U_1 für Spieler 1 und die Strategiefunktion σ'. Für den Bereich W_1 ist nach Induktionsvoraussetzung bereits bekannt, dass Spieler 0 dort mit der Strategie $\sigma|_{W_1}$ gewinnt. Somit ergeben sich die Gewinnbereiche U_0 für Spieler 0 und $U_1 \cup R' \cup W_1$ für Spieler 1 und die Strategiefunktion $\sigma|_{W_1} \cup \rho' \cup \sigma'$.

strategy(V_0, V_1, E, c)*:*

1. **if** $V_0 \cup V_1 = \emptyset$ **then**
2. **return** $(\emptyset, \emptyset, \emptyset)$
3. **elseif** odd$(\max_{v \in V_0 \cup V_1} c(v))$ **then**
4. $(W_1, W_0, \sigma) := \mathit{strategy}(V_1, V_0, E, c+1)$
5. **return** (W_0, W_1, σ)
6. **else**
7. **select** $p \in c^{-1}(\max_{v \in V_0 \cup V_1} c(v))$
8. $(R, \rho) := \mathit{attract}(V_0, E, V_0 \cup V_1 \setminus \{p\}, \{p\})$
9. $\bar{V} := V_0 \cup V_1 \setminus R \setminus \{p\}$
10. $(W_0, W_1, \sigma) := \mathit{strategy}(V_0 \cap \bar{V},\ V_1 \cap \bar{V},\ E \cap (\bar{V} \times \bar{V}),\ c)$
11. **if** $(p \in V_0\ \wedge\ pE \setminus W_1 \neq \emptyset)\ \ \vee\ \ (p \in V_1\ \wedge\ pE \cap W_1 = \emptyset)$ **then**
12. **if** $p \in V_1$ **then**
13. **return** $(W_0 \cup \{p\} \cup R,\ W_1,\ \rho \cup \sigma)$
14. **else**
15. **select** $q \in pE \setminus W_1$
16. **return** $(W_0 \cup \{p\} \cup R,\ W_1,\ \rho \cup \{(p \mapsto q)\} \cup \sigma)$
17. **else**
18. $(R', \rho') := \mathit{attract}(V_1, E, V_0 \cup V_1 \setminus W_1, W_1)$
19. $\bar{V} := V_0 \cup V_1 \setminus R' \setminus W_1$
20. $(U_0, U_1, \sigma') := \mathit{strategy}(V_0 \cap \bar{V},\ V_1 \cap \bar{V},\ E \cap (\bar{V} \times \bar{V}),\ c)$
21. **return** $(U_0,\ U_1 \cup R' \cup W_1,\ \sigma|_{W_1} \cup \rho' \cup \sigma')$

Abbildung 2.3: Strategiesynthesefunktion (McNaughtons Algorithmus)

Beispiel

Am Spielgraphen in Abbildung 2.4 zeigen wir den Ablauf des Algorithmus. Der Aufruf von *strategy*() führt zu einem rekursiven Aufruf in Zeile 4 mit dem dualen Spielgraphen. Bei diesem Aufruf fällt der Graph unter keinen der zuerst überprüften Spezialfälle. Somit wird zunächst ein Pivotelement gewählt. Der Knoten mit der größten Farbe ist v_8. Es wird der Attraktor von v_8 für Spieler 1 berechnet (für Spieler 1, weil v_8 mit 3 eine ungerade Farbe, d.h. Gewinnfarbe für Spieler 1, hat). Von v_2 und v_7 aus ist v_8 erreichbar und von v_3 gelangt man immer zu v_2. Diese Attraktormenge $\{v_2, v_3, v_7, v_8\}$ wird vor der Konstruktion des restlichen Teilspiels entfernt, womit der Teilgraph in Abbildung 2.5 verbleibt. Der Aufruf von *strategy*() führt wiederum zu einem rekursiven Aufruf für das duale Teilspiel. In diesem Teilspiel wird der Knoten v_6 als Pivot gewählt und sein Attraktor für Spieler 0 bestimmt. Von v_1 gelangt man immer zu v_6, und von v_5 kann Spieler 0 zu v_1 ziehen. Weiterhin kommt man von v_4 immer zu v_5 oder v_6. Somit umfasst der Attraktor von v_6 den ganzen Teilgraphen. Damit führt der erneute rekursive Aufruf von *strategy*() in den ersten Spezialfall des leeren Graphen. Von dem

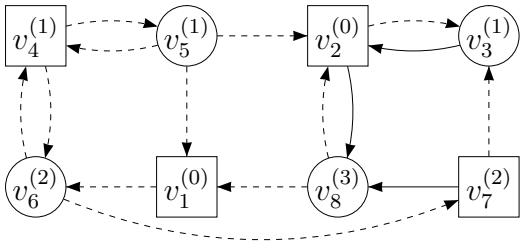


Abbildung 2.4: Spielgraph G_1.

Pivotelement v_6 muss Spieler 0 somit nicht in den Gewinnbereich des Gegners ziehen und gewinnt damit das Teilspiel in Abbildung 2.5, indem er von v_6 zu v_4 zieht.

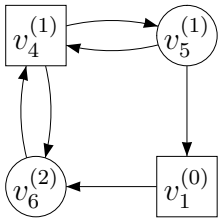


Abbildung 2.5: Spielgraph G_1.

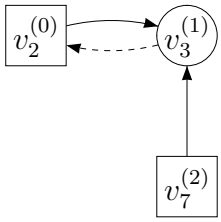


Abbildung 2.6: Spielgraph G_1.

Damit kehren wir zum Aufruf für den dualen ganzen Spielgraphen zurück, nachdem wir für das Teilspiel auf den Knoten $\{v_1, v_4, v_5, v_6\}$ als Ergebnis $W_0 = \{v_1, v_4, v_5, v_6\}$ und $W_1 = \emptyset$ erhalten haben. Vom Pivotelement v_8 aus kann Spieler 1 das Erreichen des Gewinnbereichs W_0 des Teilspiels aus Abbildung 2.5 nicht vermeiden. Somit kommen wir in den else-Fall (Zeile 17). Es wird der Attraktor des Gewinnbereichs W_0 für Spieler 0 berechnet. Dieser umfasst den Knoten v_8. Das verbleibende Teilspiel, dass in Abbildung 2.6 dargestellt ist, besteht somit aus den Knoten v_2, v_3, v_7. Auch der Aufruf von *strategy*() führt zu einem rekursiven Aufruf in Zeile 4 mit dem dualen Teilspiel.

In diesem Teilspiel wird v_7 als Pivotelement gewählt. Sein Attraktor für Spieler 0 enthält keine weiteren Knoten. Somit muss in einer weiteren Rekursion das Teilspiel aus den beiden Knoten v_2 und v_3 berechnet werden. Hier ist erneutes Dualisieren notwendig. Das Pivotelement ist v_3 und v_2 liegt in dem Attraktor für Spieler 1. Das verbleibende Teilspiel enthält somit keine Knoten, so dass Spieler 1 das Teilspiel auf v_2, v_3 gewinnt.

In dem Teilspiel $\{v_2, v_3, v_7\}$ zieht Spieler 1 von v_7 nach v_3 und somit in den Gewinnbereich von Spieler 1. Daher kann Spieler 0 mit dem Pivotelement nicht gewinnen und kommt damit in den else-Fall (Zeile 17). Der Attraktor des Gewinnbereichs $\{v_2, v_3\}$ von Spieler 1 wird bestimmt; er umfasst den ganzen Teilgraph $\{v_2, v_3, v_7\}$. Somit gewinnt Spieler 1 auf diesem Teilspiel. Diese Gewinnmenge wird an den Aufruf für den ganzen Spielgraphen zurückgeliefert.

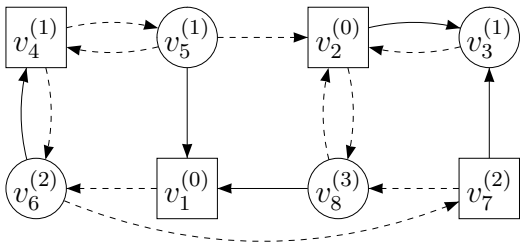


Abbildung 2.7: Spielgraph G_1.

Der Gewinnbereich von Spieler 0 ist somit $\{v_1, v_4, v_5, v_6, v_8\}$ und der von Spieler 1 ist $\{v_2, v_3, v_7\}$. In Abbildung 2.7 sind die Strategien der Spieler auf ihrem jeweiligen Gewinnbereich durch durchgezogene Kanten markiert.

Satz 2.13 [McN93] *Der Algorithmus in Abb. 2.3 berechnet für ein gegebenes Paritätsspiel die Gewinnbereiche und Gewinnstrategien für beide Spieler in Zeit* $O(2^{|V|} \cdot |E|^2)$.

Wir gehen auf die Komplexitätsaussage näher ein:

Um die Komplexität dieser Funktion zu bestimmen, müssen zunächst die rekursiven Aufrufe betrachtet werden. Die Funktion in Abb. 2.3 unterscheidet drei Hauptfälle. Der rekursive Aufruf in Zeile 4 führt zum dualen Spiel, dass stets zu einem der anderen Fälle führt; es kann also nicht vorkommen, dass in einer Folge absteigender rekursiven Aufrufe zweimal hintereinander dualisiert wird. Somit kann dieser rekursive Aufruf nicht häufiger als alle anderen Aufrufe vorkommen und kann daher unberücksichtigt bleiben. Für die übrigen rekursiven Aufrufe gilt, dass sie stets nur auf kleinere Teilspiele angewandt werden. Somit ist die Rekursionstiefe durch die Anzahl der Knoten des Spiels beschränkt. Ein Aufruf von *attract()* lässt sich in Zeit $O(|E|^2)$ berechnen, da er aus einer einfachen Erreichbarkeitssuche besteht, bei der beim Erreichen der Knoten des einen Spielers jeweils alle Vorgänger besucht werden (Zeilen 12-14 in Abb.2.2). Die übrigen Operationen in *strategy()* lassen sich ebenfalls in Zeit $O(|E|^2)$ durchführen. Damit ergibt sich eine Gesamtkomplexität $O(2^{|V|} \cdot |E|^2)$.

Eine Variante dieses Algorithmus mit besserer Zeitkomplexität wird von Zielonka in [Zie98] vorgestellt. Bei seinem Algorithmus wird nicht ein einzelner Kno-

ten (Pivotelement) mit größter Farbe gewählt und der Attraktor berechnet, sondern es wird jeweils der Attraktor aller Knoten mit größter Farbe bestimmt. Dies hat den Nachteil, dass der zweite rekursive Aufruf in der Funktion *strategy()* (Zeile 20) immer erforderlich ist. Andererseits erreicht man jedoch eine Rekursionstiefe, die maximal der Anzahl der Farben entspricht. Für diesen Algorithmus ergibt sich somit eine Gesamtzeitkomplexität von $O(2^{|c(V)|} \cdot |E|^2)$.

2.2.4 Strategiekonstruktion nach Jurdziński

In [Jur00b] beschreibt Jurdziński einen Algorithmus zur Strategiekonstruktion, der auf Fortschrittsmaßen basiert. Der Algorithmus bestimmt zu einem gegebenen Spiel ein Spielfortschrittsmaß, aus welchem sich die Gewinnbereiche beider Spieler und eine Gewinnstrategie für Spieler 0 extrahieren lassen.

Seien ein Spielgraph $G = (V, E)$ mit Knotenaufteilung $V = V_0 \dot\cup V_1$ und eine Färbungsfunktion $c : V \to [k]$ mit $k \in [|V| + 2]$ gegeben. In diesem Abschnitt verwenden wir folgende zur Paritätsbedingung (bei geeigneter Umfärbung) äquivalente Gewinnbedingung:

$$\text{Spieler 0 gewinnt } \pi \iff \min c(\mathit{Inf}(\pi)) \text{ ist gerade}$$

Die kleinste unendlich oft vorkommende Farbe ist somit signifikant für den Gewinn der Partie π. Wir partitionieren die Menge der Knoten anhand ihrer k Farben. Wir setzen für alle $i \in [k] : C_i = c^{-1}(i)$ und $n_i = |C_i|$.

Für jedes $i \in \mathbb{N}$ definieren wir auf der Menge der i-Tupel $\mathbb{N}^i$ die lexikographische Ordnung $\leq_i$; d.h. für alle $\alpha_j, \beta_j \in \mathbb{N}$ mit $j \in [i]$ gilt:

$$\begin{aligned}(\alpha_0, \dots \alpha_{i-1}) \leq_i (\beta_0, \dots \beta_{i-1}) \\ \iff \exists k \in [d+1] : (k = i \vee \alpha_k < \beta_k) \wedge \forall l \in [k] : \alpha_l = \beta_l.\end{aligned}$$

Jede Ordnung $\leq_i$ mit $i \in \mathbb{N}$ lässt sich auf die Menge der d-Tupel $\mathbb{N}^d$ mit $d \in \mathbb{N}$ und $d > i$ übertragen. Man erhält die lexikographische Ordnung der jeweils ersten i Komponenten. Für alle $\alpha_j, \beta_j \in \mathbb{N}$ mit $j \in [d]$ gilt:

$$(\alpha_0, \dots \alpha_{d-1}) \leq_i (\beta_0, \dots \beta_{d-1}) \iff (\alpha_0, \dots \alpha_{i-1}) \leq_i (\beta_0, \dots \beta_{i-1})$$

Fortschrittsmaße

Im Folgenden definieren wir Fortschrittsmaße und Spielfortschrittsmaße für den gegeben Spielgraphen G.

Ein *Fortschrittsmaß* ist eine Funktion $\mu : V \to \mathbb{N}^k$, bei welcher für alle $v, w \in V$ mit $(v, w) \in E$ gilt:

$$\begin{aligned}\mu(v) &\geq_{c(v)} \mu(w) \quad \text{falls } c(v) \text{ gerade,} \\ \mu(v) &>_{c(v)} \mu(w) \quad \text{falls } c(v) \text{ ungerade.}\end{aligned}$$

Existiert ein Fortschrittsmaß für einen Spielgraph, so ist die kleinste Farbe jeder Schleife gerade.

Sei M_G wie folgt definiert:

$$M_G = [1] \times [n_1 + 1] \times [1] \times [n_3 + 1] \times \ldots \times [1] \times [n_{k-1} + 1], \qquad \text{falls } k \text{ gerade},$$
$$M_G = [1] \times [n_1 + 1] \times [1] \times [n_3 + 1] \times \ldots \times [1] \times [n_{k-2} + 1] \times [1], \text{ sonst.}$$

Der folgende Satz besagt, dass für jedes Spiel, in dem Spieler 0 überall gewinnt, ein Fortschrittsmaß existiert, welches mit Werten aus M_G auskommt.

Satz 2.14 [Jur00b] *Sei ein Spielgraph $G = (V, E)$ mit Färbung $c : V \to [k]$ gegeben. Wenn die kleinste Färbung aller Schleifen gerade ist, dann existiert ein Fortschrittsmaß $\mu : V \to M_G$ für G.*

Wir erweitern M_G um ein neues größtes Element $\top$ und definieren $M_G^\top = M_G \dot\cup \{\top\}$. Entsprechend ergibt sich für die Ordnungen $\leq_i$:

$$m \leq_i \top \text{ für alle } m \in M_G^\top \text{ und } i \leq k.$$

Um die Veränderung von Fortschrittsmaßen entlang von Kanten besser beschreiben zu können, führen wir die folgende Funktion Prog: $(V \to M_G^\top) \times V \times V \to M_G^\top$ ein:

$$\text{Prog}(\mu, v, w) = \begin{cases} \mu(w) & \text{falls } c(v) \text{ gerade} \vee \mu(w) = \top, \\ \min\{m \in M_G^\top | m >_{c(v)} \mu(w)\} & \text{sonst.} \end{cases}$$

Zu einem gegebenen Spiel $G = (V, E)$ mit einer Färbung $c : V \to [k]$ heißt eine Funktion $\mu : V \to M_G^\top$ *Spielfortschrittsmaß*, wenn

$$\begin{aligned} &\text{für alle } v \in V_0 : && \mu(v) \geq_{c(v)} \text{Prog}(\mu, v, w) \text{ für ein } w \in vE, \\ &\text{und für alle } v \in V_1 : && \mu(v) \geq_{c(v)} \text{Prog}(\mu, v, w) \text{ für alle } w \in vE. \end{aligned}$$

Wir setzen $\|\mu\| = \{v \in V | \mu(v) \neq \top\}$.

Zu jedem Spielfortschrittsmaß lässt sich eine Strategie $\tilde\mu : V_0 \to V$ für Spieler 0 mit

$$\tilde\mu(v) \in \{w \in vE | \mu(w) = \min_{w' \in vE} \mu(w')\}$$

wählen. Eine so gewählte Strategie hat die folgende Eigenschaft:

Satz 2.15 [Jur00b] *Sei ein Spiel $G = (V, E)$ mit einer Färbung $c : V \to [k]$ gegeben. Sei μ ein Spielfortschrittsmaß. Dann ist $\tilde\mu$ eine Gewinnstrategie auf $\|\mu\|$.*

In Verbindung mit dem folgenden Satz und einem geeigneten Spielfortschrittsmaß für den gegebenen Spielgraph erhält man Gewinnbereiche und eine Gewinnstrategie für Spieler 0 auf seinem Gewinnbereich.

Satz 2.16 [Jur00b] *Sei ein Spiel gegeben. Dann existiert ein Spielfortschrittsmaß $\mu : V \to M_G^\top$, so dass $\|\mu\|$ der Gewinnbereich von Spieler 0 ist.*

Der folgende Unterabschnitt zeigt, wie sich ein solches Spielfortschrittsmaß konstruieren lässt.

Algorithmus

Im Folgenden konstruieren wir ein ‚kleinstes' Spielfortschrittsmaß durch Anwendung eines Liftoperators.

Zunächst definieren wir eine Ordnung auf den Fortschrittsmaßen eines gegebenen Spiels. Seien $\mu, \mu' : V \to M_G^\top$. Dann ist die Ordnung $\sqsubseteq$ definiert durch:

$$\mu \sqsubseteq \mu' \iff \mu(v) \leq \mu'(v) \text{ für alle } v \in V$$

Zur schrittweisen Vergrößerung einer Funktion $V \to M_G^\top$ definieren den folgenden Operator:

$$\mathit{Lift}(\mu, v)(u) = \begin{cases} \mu(u) & \text{falls } u \neq v \\ \max(\mu(v), \min_{w \in vE} \mathrm{Prog}(\mu, v, w)) & \text{falls } u = v \in V_0 \\ \max(\mu(v), \max_{w \in vE} \mathrm{Prog}(\mu, v, w)) & \text{falls } u = v \in V_1 \end{cases}$$

Für jedes $v \in V$ ist der Liftoperator $\mathrm{Lift}(., v)$ in Bezug auf $\sqsubseteq$ monoton.

Mit dem folgenden Algorithmus lässt sich ein Spielfortschrittsmaß berechnen:

ProgressMeasureLifting(G):
1. $\mu = (V \to M_G^\top : v \mapsto (0, \ldots, 0))$
2. **while** $\mu \sqsubset \mathrm{Lift}(\mu, v)$ **for some** $v \in V$ **do**
3. $\quad \mu = Lift(\mu, v)$
4. **return** μ

Satz 2.17 [Jur00b] *Der Algorithmus ProgressMeasureLifting(G) liefert für einen Spielgraphen G ein Spielfortschrittsmaß μ, aus dem sich die Gewinnbereiche $\|\mu\|$ und $V - \|\mu\|$ für beide Spieler, und die Gewinnstrategie $\tilde{\mu}$ für Spieler 0 ergibt. Er hat eine Zeitkomplexität von $O(dm \cdot \left(\frac{n}{\lfloor d/2 \rfloor}\right)^{\lfloor d/2 \rfloor})$, wobei n die Anzahl der Knoten, m die Anzahl der Kanten und d die größte Farbenzahl ist.*

Beispiel

In Abbildung 2.8 stellen wir den Algorithmus an einem Beispiel dar. Die Knoten des Spielgraphen sind mit $v_1, \ldots, v_8$ bezeichnet. In Klammern ist bei jedem Knoten seine Farbe angegeben. Über bzw. unter den Knoten sind die Werte der Fortschrittsmaße angegeben. Die Indizes geben jeweils die Nummer der Iteration des Algorithmus an. Es sind die Tupel der Ausgangssituation mit Index 0 angegeben und für die weiteren Iterationen nur noch die geänderten Bewertungen.

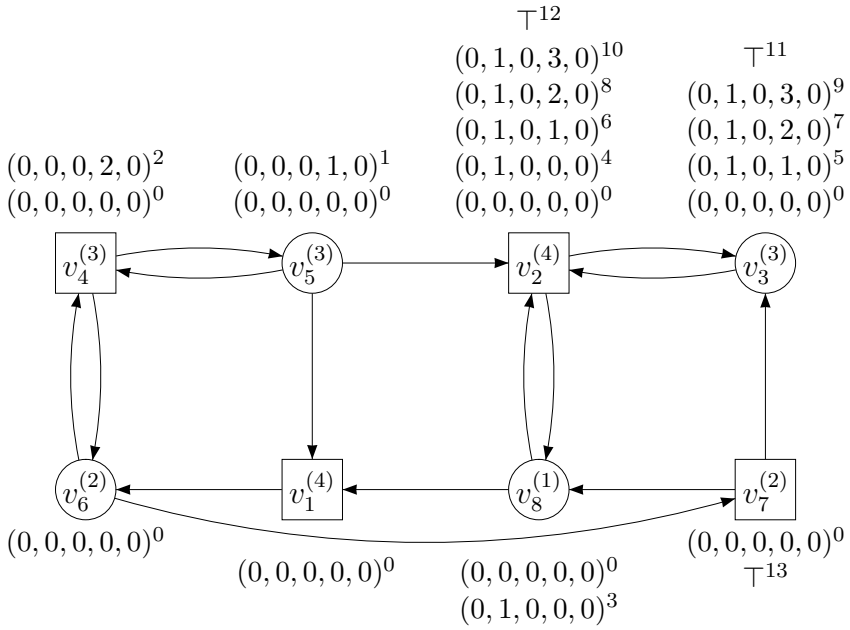


Abbildung 2.8: Spielgraph G_1.

Mit der endgültigen Fortschrittsfunktion ergibt sich als Ergebnis, dass Spieler 0 von den Knoten v_1, v_4, v_5, v_6, v_8 aus gewinnt, und Spieler 1 von v_2, v_3 und v_7 aus. Die Strategie für Spieler 0 erhält man, indem man jeweils den Nachfolger mit dem kleinsten Tupel wählt, womit sich die Züge $v_8 \mapsto v_1$, $v_6 \mapsto v_4$ und $v_5 \mapsto v_1$ ergeben.

Für diesen Algorithmus gibt Jurdziński in [Jur00b] eine Klasse von Beispielen an, bei denen exponentielle Laufzeit des Algorithmus erreicht wird.

2.3 Zusammenhang zu Payoff-Spielen

Payoff-Spiele, wie sie von Ehrenfeucht und Mycielski in [EM79] und Zwick und Paterson in [ZP96] beschrieben werden, sind eine weitere Art von Spielen, die sich auf Spielgraphen spielen lassen. Dabei wird zwischen Discounted-Payoff-Spielen (DPG) und Mean-Payoff-Spielen (MPG) unterschieden. Im Folgenden werden beide Arten von Spielen eingeführt. Es wird dargestellt, dass sich, wie in [Pur95] gezeigt, sowohl die Lösung von Mean-Payoff-Spielen auf die von Discounted-Payoff-Spielen als auch die Lösung von Paritätsspielen auf die von Mean-Payoff-Spielen zurückführen lässt. In Kapitel 3 zeigen wir, wie man ausgehend von diesem Ansatz einen Algorithmus konstruiert, der an Stelle der von Puri verwendeten reellen Bewertungen mit Bewertung aus Tupeln von Knoten und Knotenmengen

auskommt.

2.3.1 Payoff-Spiele

Payoff-Spiele werden ebenfalls auf einem Spielgraph $G = (V, E)$ gespielt, bei dem sich V in die beiden Mengen V_0 und V_1 aufteilt. Zusätzlich ist eine Rewardfunktion $r : V \to \mathbb{R}$ gegeben. Jeder Partie wird eine reelle Zahl zugeordnet: Bei *Discounted-Payoff-Spielen* (DPG) ist das Ergebnis einer Partie $\pi \in \Pi$ für einen gegebenen Discountfaktor β mit $0 < \beta < 1$:

$$R_\beta(\pi) = \sum_{k \in \mathbb{N}} \beta^k r(\pi_k)$$

und bei *Mean-Payoff-Spielen* (MPG):

$$R(\pi) = \lim_{k \to \infty} \frac{1}{k} \sum_{l=0}^{k} r(\pi_l)$$

Ziel des Spiels ist für Spieler 0 in einer Partie ein möglichst großes und für Spieler 1 ein möglichst kleines Ergebnis zu erreichen.

Eine Strategie heißt *optimal* für Spieler 0 (bzw. Spieler 1), wenn sie das maximale (bzw. minimale) Ergebnis liefert, welches unabhängig vom Verhalten des Gegners durch eine Strategie garantiert werden kann. In anderen Worten: Eine Strategie ist für einen Spieler optimal, wenn sie solche Züge bestimmt, die den geringst möglichen Schaden verursachen. Wie wir später zeigen werden, korrespondiert der Begriff der optimalen Strategie für Payoff-Spiele zu dem der Gewinnstrategie bei den unendlichen Spielen.

Wir bezeichnen diejenige Partie, die von einem Knoten $v \in V$ nach einer Strategie $\sigma : V_0 \to V$ für Spieler 0 und einer Strategie $\tau : V_1 \to V$ für Spieler 1 gespielt wird, mit $\pi_{v,\sigma,\tau}$. Wir sprechen davon, dass ein Payoff-Spiel für einen Startknoten v ein Ergebnis hat, wenn für alle Strategien $\sigma : V_0 \to V$ und $\tau : V_1 \to V$ auf dem gegeben Spielgraphen

$$\sup_\sigma \inf_\tau \bar{R}(\pi_{v,\sigma,\tau}) = \inf_\tau \sup_\sigma \bar{R}(\pi_{v,\sigma,\tau})$$

gilt, wobei $\bar{R}$ z.B. R oder R_β sein kann.

Aus [Pur95] seien hier zwei Ergebnisse über Discounted- und Mean-Payoff-Spiele zitiert:

Satz 2.18 [**Pur95**] *Jedes DPG besitzt für alle Knoten ein Ergebnis und es existieren für beide Spieler jeweils optimale positionale Strategien, die zu diesen Ergebnissen führen.*

Satz 2.19 [Pur95] *Gegeben sei ein Spielgraph mit einer Bewertung. Dann existiert ein Discountfaktor β_0, so dass alle Lösungen des Discounted-Payoff-Spiels mit Discountfaktor $\beta > \beta_0$ auch Lösungen des entsprechenden Mean-Payoff-Spiels sind.*

Es sei darauf hingewiesen, dass ein Mean-Payoff-Spiel auch andere Lösungen als ein entsprechendes Discounted-Payoff-Spiel haben kann, da endlich oft besuchte Knoten durch die Grenzwertbildung beim der MPG-Bewertung im Ergebnis nicht berücksichtigt werden und daher nicht wie bei DPGs optimal gewählt sein müssen.

2.3.2 Paritätsspiele als spezielle Mean-Payoff-Spiele

Wie Puri in [Pur95] zeigt, lassen sich Paritätsspiele als Mean-Payoff-Spiele in dem Sinne kodieren, dass jede Lösung des Mean-Payoff-Spiels eine Lösung für das Paritätsspiel liefert.

Satz 2.20 [Pur95] *Sei $G = (V, E)$ ein Spielgraph, $V = V_0 \dot\cup V_1$ und $c : V \to [|V|]$ eine Färbung der Knoten. Sei $r : V \to \mathbb{R}$ eine Rewardfunktion mit:*

$$r(v) = (-|V|)^{c(v)}$$

für alle $v \in V$. Seien f und g optimale positionale Strategien für Spieler 0 resp. 1 bzgl. des Mean-Payoff-Spiels mit der Rewardfunktion r. Dann gilt für das Paritätsspiel mit der Färbung c:

$$Win_0 = \{v \in V | R(\pi_{v,f,g}) \geq 0\}$$

$$Win_1 = \{v \in V | R(\pi_{v,f,g}) < 0\}$$

und $f|_{Win_0}$ ist eine Gewinnstrategie für Spieler 0 von Win_0 aus, und $g|_{Win_1}$ ist eine Gewinnstrategie für Spieler 1 von Win_1 aus.

Beweis: Sei $G = (V, E)$ ein Spielgraph, $V = V_0 \dot\cup V_1$ und $c : V \to [|V|]$ eine Färbung der Knoten. Sei $r : V \to \mathbb{R}$ eine Rewardfunktion mit:

$$r(v) = (-|V|)^{c(v)}$$

für alle $v \in V$. Seien f und g optimale positionale Strategien für Spieler 0 resp. 1 bzgl. des Mean-Payoff-Spiels mit der Rewardfunktion r. Zu zeigen ist für jeden Startknoten $v \in V$ und jede positionale Strategie f für Spieler 0 und jede positionale Strategie g für Spieler 1:

$$\max c(Inf(\pi_{v,f,g})) \in 2\mathbb{N} \iff R(\pi_{v,f,g}) \geq 0.$$

Sei $L = \mathit{Inf}(\pi_{v,f,g})$. Dann gilt:

$$R(\pi_{v,f,g}) \;=\; \lim_{k\to\infty} \frac{1}{k} \sum_{l=0}^{k} r(\pi_l) \;=\; \frac{1}{|L|} \sum_{v\in L} r(v)$$

Sei $M = \{v \in L | c(v) = \max c(L)\}$.

$$\left| \sum_{v\in L\setminus M} r(v) \right| \leq (|L| - |M|) \cdot |V|^{\max c(L)-1} < |V|^{\max c(L)} \leq |M| \cdot |V|^{\max c(L)}$$

Also

$$R(\pi_{v,f,g}) \geq 0 \iff |M| \cdot (-|V|)^{\max c(L)} \geq 0 \iff \max c(L) \in 2\mathbb{N}.$$

Somit gewinnt Spieler 0 genau dann eine Partie nach der Mean-Payoff-Bedingung, wenn er sie nach der Paritätsbedingung gewinnt, sofern Rewardfunktion und Färbung in der vorausgesetzten Beziehung stehen. □

Dass sich bei der gewählten Färbung für eine Partie der Gewinner des Paritätsspiels an dem Ergebnis des Mean-Payoff-Spiels ablesen lässt, gilt nur bei positionalen Strategien. Betrachten wir in dem Spielgraph in Abbildung 2.9 Partien, die von Knoten v_2 ausgehen. Zieht Spieler 0 nach der positionalen Strategie v_2 zu v_1, so ist das Ergebnis des MPG $\frac{1+1}{2} = 1$ und Spieler 0 gewinnt nach Paritätsbedingung. Zieht Spieler 0 nach der positionalen Strategie v_2 zu v_3, so ist das Ergebnis des MPG $\frac{1-3}{2} = -1$ und Spieler 1 gewinnt nach Paritätsbedingung. Zieht Spieler 0 jedoch je zweimal von v_2 zu v_1 und dann einmal zu v_3, so ergibt sich die Partie $(v_2v_1v_2v_1v_2v_3)^\omega$, die das MPG-Ergebnis $\frac{1+1+1+1+1+1-3}{6} = \frac{1}{3}$ hat, in der aber Spieler 1 nach Paritätsbedingung gewinnt, da der Knoten v_3 mit der ungeraden Farbe 1 unendlich oft erreicht wird.

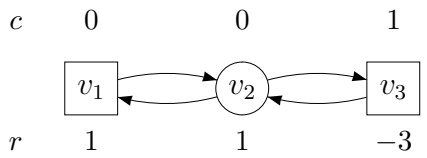


Abbildung 2.9: Unterschiede bei nicht-positionalen Strategien.

2.4 Puris Synthesealgorithmus

Der in diesem Abschnitt beschriebene Algorithmus zur Synthese optimaler Strategien für Discounted-Payoff-Spiele wurde von Puri [Pur95] entwickelt. Er basiert

auf dem allgemeineren Ansatz der sukzessiven Strategieverbesserung, wie er vorher bereits für Stochastische Spiele [Sha53] angewandt wurde.

Gegeben ist ein Spielgraph (V, E), wobei sich die Knotenmenge V in die Knoten V_0 von Spieler 0 und die Knoten V_1 von Spieler 1 aufteilt. Weiterhin ist eine Gewichtsfunktion $r : V \to \mathbb{R}$ und ein Discountfaktor $\beta \in (0, 1)$ gegeben.

Wir betrachten zunächst den Fall, dass positionale Strategien σ und τ für beide Spieler gegeben sind. Aus der Definition des Ergebnisses der Partie eines DPG ergibt sich für jeden Knoten v und seinen durch die Strategien festgelegten Nachfolger:

$$R_\beta(\pi_{v,\sigma,\tau}) = r(v) + \beta R_\beta(\pi_{w,\sigma,\tau}) \text{ für } w = \sigma(v), \text{ falls } v \in V_0$$

oder

$$R_\beta(\pi_{v,\sigma,\tau}) = r(v) + \beta R_\beta(\pi_{w,\sigma,\tau}) \text{ für } w = \tau(v), \text{ falls } v \in V_0.$$

Ersetzt man $R_\beta(\pi_{\,\cdot\,,\sigma,\tau})$ durch die Variable $\varphi \in \mathbb{R}^V$, so erhält man folgendes lineare Gleichungssystem:

$$\varphi(v) = r(v) + \beta\varphi(\sigma(v)) \text{ für } v \in V_0$$

$$\varphi(v) = r(v) + \beta\varphi(\tau(v)) \text{ für } v \in V_1$$

Dieses Gleichungsystem besitzt genau eine Lösung, da φ der Fixpunkt einer Kontraktion $(0 < \beta < 1)$ im $\mathbb{R}^V$ ist. D.h., wenn φ Lösung dieses Gleichungssystems ist, so gilt $R_\beta(\pi_{v,\sigma,\tau}) = \varphi(v)$ für alle $v \in V$. Im Folgenden bezeichnen wir φ auch als Bewertungsfunktion.

2.4.1 Das Optimierungsproblem

Dieses Vorgehen, die Ergebnisse von Partien als Lösungen eines Gleichungssystems aufzufassen, werden wir in diesem Unterabschnitt auf den Fall ausdehnen, in welchem die Strategien noch zu wählen sind.

Betrachtet man die Nachfolger eines Knotens und kennt für sie das Ergebnis einer von ihnen aus nach optimalen Strategien gespielten Partie, so kann für diesen Knoten allein anhand dieser Ergebnisse eine optimale Strategie gewählt werden. Spieler 0 (1) wählt offensichtlich den Nachfolger mit dem größten (kleinsten) Ergebnis.

Auf diese Weise erhält man eine lokale Eigenschaft für die Wahl einer optimalen Strategie. Nutzt man die Existenz positionaler optimaler Strategien aus, so lässt sich die Lösung eines Discounted-Payoff-Spiels, eine Bewertung $\varphi : V \to \mathbb{R}$, durch folgendes Gleichungssystem formulieren:

$$\varphi(v) = r(v) + \beta \max_{v' \in vE} \varphi(v') \text{ für } v \in V_0$$

$$\varphi(v) = r(v) + \beta \min_{v' \in vE} \varphi(v') \text{ für } v \in V_1$$

Die positionalen Strategien ergeben sich dabei offensichtlich durch die für jeden Knoten gewählten Nachfolger.

Problem bei der Lösung dieses Gleichungssystems ist, dass gleichzeitig Minima und Maxima bestimmt werden müssen; d.h. die Optima für die verschiedenen Komponenten von φ können nicht unabhängig voneinander bestimmt werden.

Beschränkt man sich zunächst auf das Problem zu einer gegebenen Strategie des Spielers 0 eine optimale Antwortstrategie für Spieler 1 zu bestimmen, so erhält man das folgende einfachere Gleichungssystem:

$$\begin{aligned} \varphi(v) &= r(v) + \beta\varphi(\sigma(v)) && \text{für } v \in V_0 \\ \varphi(v) &= r(v) + \beta \min_{v' \in vE} \varphi(v') && \text{für } v \in V_1 \end{aligned} \tag{2.4}$$

In diesem Gleichungssystem lassen sich die Komponenten von φ unabhängig optimieren.[2] Dieses Optimierungsproblem lässt sich als Maximierungsproblem für das folgende Ungleichungssystem auffassen:

$$\varphi(v) = r(v) + \beta\varphi(\sigma(v)) \text{ für } v \in V_0$$

$$\varphi(v) \leq r(v) + \beta\varphi(v') \text{ für } v \in V_1, v' \in vE$$

Dies ist jedoch ein sehr spezielles lineares Ungleichungssystem, da in jeder Ungleichung nur zwei Variablen vorkommen. Megiddo zeigt in [Meg83], dass sich für solche Ungleichungssysteme in Zeit $O(mn^3 \log(m))$ eine maximale Lösung bestimmen lässt, wobei n die Anzahl der Variablen (hier: die Anzahl der Knoten von Spieler 1) und m die Anzahl der Ungleichungen (hier: die Anzahl der Kanten, die von Knoten von Spieler 1 ausgehen) ist.

Die Lösung des Gleichungssystems 2.4 liefert somit zu einer gegebenen Strategie von Spieler 0 eine optimale Antwortstrategie und gleichzeitig die Ergebnisse aller nach diesen beiden Strategien gespielten Partien.

2.4.2 Der Strategieverbesserungs-Algorithmus

Der von Puri entwickelte Strategieverbesserungs-Algorithmus verbessert die Strategie von Spieler 0 ausgehend von einer beliebigen Anfangsstrategie schrittweise. In jedem Schritt wird dabei eine optimale Antwortstrategie für Spieler 1 berechnet.

Der Algorithmus (siehe Abbildung 2.10) besteht im wesentlichen aus drei Teilen:

1. Wahl einer initialen Strategie für Spieler 0.

[2] Puri bemerkt in seiner Arbeit, dass für dieses Gleichungssystem in Polynomzeit eine Lösung bestimmt werden kann, führt hierzu jedoch kein Verfahren an.

1. Setze $i = 0$; Spieler 0 wählt eine positionale Strategie σ_0.
2. **repeat**
3. Spieler 1 wählt eine optimale Antwortstrategie τ_i.
4. Bewertung φ von (σ_i, τ_i) berechnen.
5. Für $u \in V_0$ setze $\sigma_{i+1}(u) = \sigma_i(u)$, falls $\sigma_i(u) \in \arg\max_{v \in V, uEv} \varphi(v)$, sonst wähle $\sigma_{i+1}(u) \in \arg\max_{v \in V, uEv} \varphi(v)$.
6. Setze $i = i + 1$.
7. **until** $\sigma_i = \sigma_{i-1}$
8. **return** σ_{i-1}, τ_{i-1}

Abbildung 2.10: Puris Strategiesynthese-Algorithmus für Discounted-Payoff-Spiele

2. Eine Schleife, in der in jeder Iteration eine optimale Antwortstrategie für Spieler 1 im Bezug auf die aktuelle Strategie von Spieler 0 berechnet wird und eine Bewertung der Partien, die sich durch die für beide Spieler festgelegten Strategien ergeben. Aufgrund dieser Bewertung wird jeweils eine neue Strategie für Spieler 0 ermittelt. Stimmt diese Strategie für Spieler 0 mit der zuvor gewählten überein, so terminiert die Schleife.
3. Rückgabe der für beide Spieler zuletzt berechneten Strategien.

Die Wahl einer optimalen Anwortstrategie und das Berechnen einer Bewertung kann wie im vorangegangen Unterabschnitt beschrieben durchgeführt werden.

Satz 2.21 [Pur95] *Der Algorithmus in Abbildung 2.10 terminiert und liefert zu einem gegebenen Spielgraphen mit Paritätsgewinnbedingung die Gewinnbereiche beider Spieler und Gewinnstrategien auf diesen Gewinnbereichen.*

Beweisidee: In jedem Schritt verbessert sich die Bewertung, d.h. es gibt keinen Knoten, für den das Partieergebnis kleiner wird.

Ein Verbesserungschritt ist eine Kontraktion im Bezug auf die Bewertung. Somit gibt es nur eine Bewertung, die die Terminationsbedingung erfüllen kann. Die Lösung des Spiels ist eine Bewertung, die nicht mehr verbessert werden kann. Somit liefert der Algorithmus eine korrekte Lösung, wenn er terminiert.

Da in jedem Schritt die Bewertung echt größer wird, und die Bewertung nur von der Strategie von Spieler 0 abhängt, der nur endlich viele Strategien wählen kann, terminiert der Algorithmus nach endlich vielen Verbesserungsschritten.

Diskussion Mit Hilfe der beiden Transformationen aus diesem Abschnitt lässt sich jedes Paritätsspiel auf ein Mean-Payoff-Spiel und dies auf ein Discounted-Payoff-Spiel zurückführen. Verwendet man zur Strategiekonstruktion für Discounted-Payoff-Spiele Puris Algorithmus, so hat dies den Nachteil, dass in jedem Iterationsschritt ein Ungleichungssystem mit reellen Konstanten mit hoher Genauigkeit gelöst werden muss.

Im folgenden Kapitel wird ein Algorithmus beschrieben, der eine diskrete Bewertung verwendet. In Abschnitt 3.7 wird der Bezug zur hier dargestellten Bewertung von Puri aufgezeigt. Aus diesem Zusammenhang ergibt sich, dass die von Puri verwendeten reellen Bewertung große Redundanzen enthalten, was zu einem unnötig hohen Aufwand beim Lösen der Ungleichungssysteme führt.

Kapitel 3

Der diskrete Strategie-verbesserungs-Algorithmus

In diesem Kapitel beschreiben wir einen Algorithmus zur Strategiesynthese, der zu der Klasse der Strategieverbesserungs-Algorithmen (SVA) gehört. Die Grundlage dieses Algorithmus bildet der in Abschnitt 2.4 beschriebener SVA von Puri [Pur95], mit dem optimale Strategien für Discounted-Payoff-Spiele (DPGs) konstruiert werden können. Im Gegensatz zum Algorithmus von Puri werden jedoch diskrete Bewertungen verwendet, so dass wir den hier beschriebenen Algorithmus kurz als *diskreten SVA (dSVA)* bezeichnen.

Wir beginnen mit der Einführung von Partieprofilen. Ein Partieprofil beschreibt die wesentlichen Eigenschaften einer gegebenen Partie. Darauf aufbauend werden Bewertungen definiert. Sie ordnen jedem Knoten eines Graphen das Profil einer von ihm ausgehenden Partie zu. Dabei sind die Partien, die von den verschiedenen Knoten ausgehen, insofern miteinander verträglich, als sie alle durch uniforme Strategien induziert werden. Eine Bewertung enthält hinreichend viele Informationen, um ihr Strategien für beide Spieler zu entnehmen, die Partien induzieren, deren Partieprofile diese Bewertung ergeben.

Im darauf folgenden Abschnitt identifizieren wir die Klasse der optimalen Bewertungen. Es zeigt sich, dass diese Bewertungen gerade die jener Partien sind, die durch Gewinnstrategien induziert werden.

Danach geben wir den Algorithmus zur Strategiesynthese an. Der Algorithmus wählt zunächst eine Bewertung (aufbauend auf einer beliebigen Strategie für Spieler 0), die optimal für Spieler1 ist. Dann wird diese Bewertung schrittweise für Spieler 0 verbessert, bis eine für beide Spieler optimale Bewertung entsteht, aus der sich Gewinnbereiche und -strategie leicht extrahieren lassen. Es folgt der Korrektheitsbeweis, eine Komplexitätsabschätzung und eine Diskussion von Verbesserungsmöglichkeiten.

Im letzten Abschnitt des Kapitels zeigen wir die Verbindung zum Strategiesynthese-Algorithmus von Puri auf.

Für dieses Kapitel setzen wir voraus, dass die gegebenen Spielgraphen bipartit sind. Dies stellt keine echte Einschränkung dar, da aus einem Spielgraphen, der diese Bedingung nicht erfüllt, leicht einer konstruiert werden kann, der sie erfüllt: Jede Kante, deren Quell- und Zielknoten dem gleichen Spieler gehören, wird entfernt und durch eine Kante vom Quellknoten zu einem neuen Knoten des anderen Spielers und eine Kante von diesem neuen Knoten zum Zielknoten ersetzt. Die hinzugefügten Knoten erhalten alle die geringstwertige Farbe.

Weiterhin sind alle betrachteten Strategien positionale Strategien und können daher in der Form:

$$\sigma : U_0 \to V_1 \text{ mit } U_0 \subseteq V_0 \qquad \text{für Spieler 0}$$

$$\tau : U_1 \to V_0 \text{ mit } U_1 \subseteq V_1 \qquad \text{für Spieler 1}$$

dargestellt werden. Auch bei der Betrachtung von Partien werden wir uns auf solche beschränken, die nach positionalen Strategien gespielt sind. Zu jeder Partie $\pi \in V^\omega$, die nach positionalen Strategien gespielt ist, existieren eindeutige $\alpha, \beta \in V^*$, wobei in $\alpha\beta$ kein Knoten mehrfach vorkommt, so, dass $\pi = \alpha\beta^\omega$ gilt. Somit besteht jede solche Partie π aus dem Weg zur Schleife α und der Schleife β. Die Knoten in β bezeichnen wir als *Schleifenknoten* der Partie π.

3.1 Partieprofile und Bewertungen

Aufbauend auf der Färbung $c : V \to [|V| + 1]$ der Paritätsgewinnbedingung und die dadurch gegebene partielle Ordnung auf den Knoten eines Spielgraphen definieren wir eine verfeinerte totale Ordnung, die wir als Relevanzordnung bezeichnen. Unter ihrer Verwendung lässt sich das Profil einer Partie definieren. Partieprofile beschreiben die für eine Bewertung entscheidenden Eigenschaften einer Partie. Eine Bewertung ordnet jedem Knoten eines Spielgraphen ein Partieprofil zu, wobei die Partien der Profile durch uniforme Strategien induziert sein müssen. Die exakte Formulierung dieser Verträglichkeitsbedingung erfolgt mit Hilfe der Nachfolgerprofilrelation. Ein nachfolgender Unterabschnitt zeigt, wie sich Bewertungen vergleichen lassen.

Sei $G = (V, E)$ ein Spielgraph, bei dem sich die Knotenmenge V auf die Knotenmenge V_0 für Spieler 0 und die Knotenmenge V_1 für Spieler 1 aufteilt. Durch eine Funktion $c : V \to [|V| + 1]$ sei eine Färbung der Knoten gegeben.

Im Folgenden benötigen wir eine totale Ordnung der Knoten, welche die auf der Färbung gegebene Ordnung $<$ verfeinert. Wir wählen eine *Relevanzordnung* $<_c$ auf den Knoten, die folgende Bedingung erfüllt:

$$c(u) < c(v) \implies u <_c v \text{ für alle } u, v \in V. \tag{3.1}$$

Offensichtlich kann es verschiedene solcher Ordnungen zu einer Färbung geben. Für die folgenden Überlegungen legen wir eine solche beliebige aber feste Ordnung zugrunde.

Weitet man den zugelassenen Wertebereich für Färbungen aus ($c : V \to [2|V|]$), so kann man zu jeder Färbungsfunktion eine injektive Färbungsfunktion bestimmen, für die alle Partien den gleichen Gewinner wie zuvor haben. Ist die Färbungsfunktion injektiv, legt 3.1 eine Relevanzordnung $<_c$ eindeutig fest. Zur Vereinfachung werden wir daher im Folgenden stets eine äquivalente injektive Färbungsfunktion bestimmen, falls die gegebene nicht injektiv ist. Außerdem werden wir verkürzend $<$ anstelle von $<_c$ schreiben.

Wir teilen die Menge der Knoten in die beiden Mengen

$$V_+ = \{v \in V \mid c(v) \textit{ ist gerade}\} \quad \text{und} \quad V_- = \{v \in V \mid c(v) \textit{ ist ungerade}\}$$

auf. Wir bezeichnen die Knoten in V_- als negative Knoten und die in V_+ als positive Knoten.

Beispiel

In Abbildung 3.1 ist der Spielgraph $G_1 = (\{v_1, \ldots, v_8\}, E)$ dargestellt. Durch die Färbungsfunktion $c : V \to [4]$ sind den Knoten folgende Farben zugeordnet:

v	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8
$c(v)$	0	0	1	1	1	2	2	3

Eine Färbungsfunktion $c' : V \to [12]$, die die gleiche Gewinnbedingung beschreibt, aber zu einer eindeutig festgelegten Relevanzordnung führt, erhalten wir durch ‚Einfügen' zusätzlicher Farben:

v	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8
$c'(v)$	0	2	3	5	7	8	10	11

In der Abbildung sind Knoten aus V_- mit $-$ und solche aus V_+ mit $+$ markiert. Als Relevanzordnung erhalten wir:

$$v_1 < v_2 < v_3 < v_4 < v_5 < v_6 < v_7 < v_8$$

Auch bei weiteren Beispielen in diesem Kapitel werden wir die Konvention beibehalten, die Knoten in aufsteigender Relevanzordnung zu nummerieren.

Wie im vorigen Kapitel bezeichnen wir mit $\mathit{Inf}(\pi)$ die Menge der unendlich oft in der Partie π vorkommenden Knoten. Mit diesen Definitionen können wir die Gewinnbedingung für Spieler 0 in einer Partie π wie folgt formulieren:

$$\max_<(\mathit{Inf}(\pi)) \in V_+.$$

Wesentliche Daten einer Partie π sind die folgenden drei Werte:

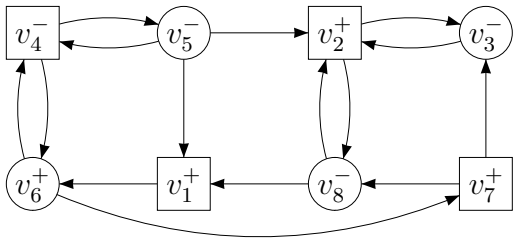


Abbildung 3.1: Spielgraph G_1.

1. Der relevanteste Schleifenknoten u_π:

$$u_\pi = \max_<(\mathit{Inf}(\pi)).$$

2. Die Menge der Knoten P_π in der Partie π, die relevanter als u_π sind:

$$P_\pi = \{v \in V \mid u_\pi < v \wedge \pi \in V^* v V^\omega\}.$$

3. Die Anzahl von Knoten, die vor dem ersten Besuch von u_π besucht werden:

$$e_\pi = \min\{i \in \mathbb{N} \mid \pi \in V^i u_\pi V^\omega\}.$$

Wir bezeichnen das Tripel (u_π, P_π, e_π) als *Partieprofil* von π. Partieprofile beschreiben Äquivalenzklassen von Partien auf dem gegebenen Spielgraph. Per Definition gehört jedes Partieprofil zu folgender Menge:

$$\mathcal{D} \;=\; \{(u, P, e) \in V \times 2^V \times [|V|] \mid \forall v \in P : u < v\}.$$

Zu einer Partie $v\pi \in V^\omega$ mit $v \in V$ und $\pi \in V^\omega$ bezeichnen wir die Partie $s(v\pi) = \pi$ als *Nachfolgerpartie*. Sie ist die Partie, die im Nachfolger des Anfangsknotens der gegebenen Partie beginnt und bei der sich beide Spieler genau wie bei der ursprünglichen Partie verhalten.

Um die Beziehung zwischen den Partieprofilen aufeinanderfolgender Knoten einer Partie zu beschreiben, definieren wir die *Nachfolgerprofilrelation* $N \subseteq V \times \mathcal{D} \times \mathcal{D}$. Sind $x \in V$ und zwei Profile (u, P, e) und (v, Q, f) gegeben, dann gilt $N(x, (u, P, e), (v, Q, f))$, wenn:

$$\begin{aligned} u = v \;\wedge\; (\quad & (x = u \;\wedge\; P = Q = \emptyset \quad \wedge\; e = 0) \\ \vee\; & (x < u \;\wedge\; P = Q \qquad\;\; \wedge\; e = f + 1) \\ \vee\; & (x > u \;\wedge\; P = Q \,\dot\cup\, \{x\} \;\wedge\; e = f + 1)) \end{aligned}$$

Man beachte, dass im Fall $x > u$ die Relation N nur zutreffen kann, wenn $x \notin Q$.

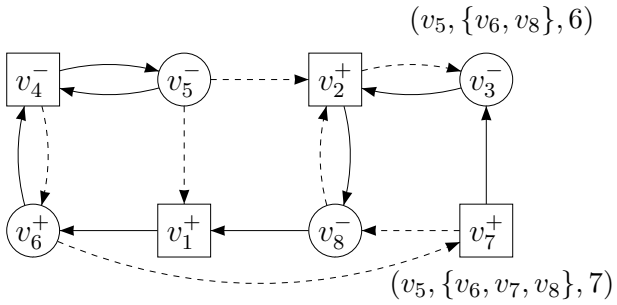


Abbildung 3.2: Spielgraph mit eingezeichneten Strategien.

Beispiel

In Abbildung 3.2 ist eine Partie angegeben, die von v_7 über v_3, v_2, v_8, v_1, v_6 in die Schleife zwischen v_4 und v_5 führt. Am Knoten v_7 ist das Profil $(v_5, \{v_6, v_7, v_8\}, 7)$ dieser Partie vermerkt. Am Knoten v_3 steht das Profil $(v_5, \{v_6, v_8\}, 6)$, der in diesem Knoten beginnenden Nachfolgerpartie. Der Knoten v_7 und die beiden gegeben Profile stehen in der Nachfolgerprofilrelation:

$$\big(v_7,\ (v_5, \{v_6, v_7, v_8\}, 7),\ (v_5, \{v_6, v_7, v_8\}, 7)\big) \in N$$

Die Nachfolgerprofilrelation beschreibt den zwischen den Partieprofilen aufeinanderfolgender Knoten einer Partie geltenden Zusammenhang:

Bemerkung 3.1 *Sei (V, E) ein Spielgraph mit einer Aufteilung der Knotenmenge $V = V_0 \dot\cup V_1$ auf beide Spieler und einer Färbung $c : V \to [|V| + 1]$, und sei $\pi \in V^\omega$ eine Partie auf diesem Spielgraph. Sei $R \in \mathcal{D}$ das Partieprofil von π und $S \in \mathcal{D}$ das Partieprofil von $s(\pi)$. Dann gilt $N(\pi_0, R, S)$, wobei π_0 der erste Knoten der Partie π ist.*

Das folgende Lemma zeigt: Wenn den Knoten einer Partie beliebige Partieprofile zugeordnet sind und die Nachfolgerprofilrelation entlang der Knoten dieser Partie und ihrer Profile gilt, dann beschreiben die Partieprofile jeweils genau diese von ihrem jeweiligen Knoten ausgehende Partie.

Lemma 3.2 *Sei (V, E) ein Spielgraph mit einer Aufteilung der Knotenmenge $V = V_0 \dot\cup V_1$ auf beide Spieler und einer Färbung $c : V \to [|V| + 1]$, und sei $\pi \in V^\omega$ eine Partie auf diesem Spielgraph. Sei $U \subseteq V$ die Menge der Knoten, die in π vorkommen und $\varphi : U \to \mathcal{D}$ eine Funktion, die den Knoten der Partie Profile zuordnet. Stehen aufeinanderfolgende Knoten der Partie in der Nachfolgerprofilrelation, d. h. gilt*

$$\forall u, v \in V, \alpha, \beta \in V^* : \pi = \alpha u v \beta \implies N(u, \varphi(u), \varphi(v)),$$

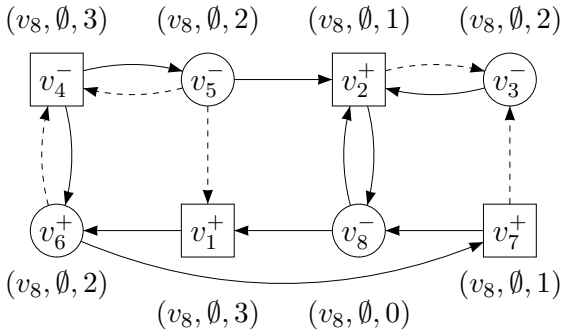


Abbildung 3.3: Spielgraph G_1 mit Bewertung.

dann ist $\varphi(\pi_0)$ das Profil der Partie π, wobei π_0 der Anfangsknoten der Partie π ist.

Dieses Lemma liefert eine lokale Bedingung zur Konstruktion von Bewertungen. Eine *Bewertung* ist eine Funktion $\varphi : V \to \mathcal{D}$, die jedem Knoten ein Partieprofil zuordnet und für die gilt:

$$\forall x \in V\, \exists y \in V\colon xEy \wedge N(x, \varphi(x), \varphi(y)).$$

Dass jeder Knoten einen Nachfolger in Nachfolgerprofilrelation besitzt, garantiert nach Lemma 3.2, dass von jedem Knoten aus eine Partie existiert, die seinem Profil entspricht.

Für $N(x, \varphi(x), \varphi(y))$ führen wir die Abkürzung $x \lhd_\varphi y$ ein. Wir bezeichnen $\lhd_\varphi$ als φ-progress-Relation.

Um uns in einer Bewertung φ auf die Komponenten eines Profils $\varphi(v) = (u, P, e)$ zu beziehen, schreiben wir φ_0, φ_1 und φ_2, wobei $\varphi_0(v) = u$, $\varphi_1(v) = P$ und $\varphi_2(v) = e$.

Beispiel

In Abbildung 3.3 ist für den bereits verwendeten Beispielgraphen G_1 eine Bewertung angegeben. Stehen die Knoten einer Kante von x nach y in Progress-Relation $(x \lhd_\varphi y)$, so ist diese Kante in der Abbildung durchgezogen gezeichnet, anderenfalls gestrichelt. An diesem Beispiel ist zu sehen, dass ein Knoten durchaus mehrere Nachfolger, die mit ihm in Progress-Relation stehen, haben kann; d.h. ein Profil beschreibt, wie bereits erwähnt, eine Klasse von Partien.

Eine einfache Möglichkeit, eine Bewertung für einen Graphen zu bestimmen, besteht darin, für jeden Knoten eine ausgehende Kante zu wählen. Durch diese Wahl wird jedem Knoten eindeutig eine Partie und damit auch ein Partieprofil zugeordnet. Nach Konstruktion erfüllt diese Zuordnung von Profilen zu Knoten die Eigenschaften einer Bewertung.

Eine Strategie $\rho : V_i \to V_{1-i}$ ($i \in \{0,1\}$) heißt *i-verträglich* (oder kurz: *verträglich*) mit einer Bewertung φ, wenn $\forall v \in V_i : v \lhd_\varphi \rho(v)$.

3.1.1 Vergleich von Bewertungen

Eine Bewertung repräsentiert für jeden Knoten eine Klasse von Partien und damit für beide Spieler jeweils die Klasse der uniformen Strategien, nach denen diese Partien gespielt werden. Ziel ist es, eine partielle Ordnung auf den Bewertungen zu definieren, welche die Qualität der Partien bzw. Strategien aus Sicht von Spieler 0 vergleicht; d.h. Bewertungen sind um so größer, je besser die repräsentierten Strategien für Spieler 0 und je schlechter die repräsentieren Strategien für Spieler 1 sind.

Zunächst definieren wir eine weitere totale Ordnung $\prec$ auf den Knoten des Spielgraphen, die wir als Reward-Ordnung bezeichnen. Ein Knoten ist in Bezug auf diese Ordnung größer, wenn er als unendlich oft erreichter Knoten in einer Partie für Spieler 0 besser ist. Diese Ordnung lässt sich auf alle Komponenten eines Partieprofils und damit auch auf das Profil ausdehnen. Aus dieser totalen Ordnung auf den Partieprofilen ergibt sich eine partielle Ordnung auf den möglichen Bewertungen eines Spielgraphen.

Die Reward-Ordnung $\prec$ misst den Wert von Knoten aus der Sicht von Spieler 0 unter Rückgriff auf die eingeführte Relevanzordnung. Der kleinste Reward ist der des Knotens mit höchster Relevanz in V_-, und der größte Reward der des Knotens mit höchster Relevanz in V_+. Somit sind insbesondere alle negativen Knoten $\prec$-kleiner als die positiven. Genauer:

$$u \prec v \iff (u < v \wedge v \in V_+) \vee (v < u \wedge u \in V_-).$$

Beispiel

Im Beispiel aus Abbildung 3.3 ergibt sich für die $\prec$-Ordnung:[1]

$$v_8^- \prec v_5^- \prec v_4^- \prec v_3^- \prec v_1^+ \prec v_2^+ \prec v_6^+ \prec v_7^+.$$

Wir erweitern die Reward-Ordnung $\prec$ auf Mengen von Knoten. Wenn $P, Q \in 2^V$ verschieden sind, entscheidet der Knoten v mit höchster Relevanz in der symmetrischen Mengendifferenz $P \triangle Q$, ob $P \prec Q$ oder $Q \prec P$ gilt: Wenn $v \in V_+$, dann ist die Menge, die v enthält, „größer“. Anderenfalls, wenn $v \in V_-$, ist die Menge, die v nicht enthält, „größer“. Formal:

$$P \prec Q \iff P \neq Q \wedge \max_<(P \triangle Q) \in Q \triangle V_-$$

[1] Die Markierungen + und − kennzeichnen die Zuordnung der Knoten zu V_+ bzw. V_-. Sie erleichtern es zwei Knoten bezüglich der Ordnung $\prec$ zu vergleichen. Die Ordnung $\prec$ entspricht der Ordnung auf den entsprechend vorzeichenbehafteten Indizes der Knoten.

Diese Ordnung ist eine totale Ordnung auf 2^V. Wir benutzen dasselbe Symbol $\prec$ für Knoten und Mengen von Knoten, da insbesondere auch $\{u\} \prec \{v\} \iff u \prec v$ für alle $u, v \in V$ gilt.

Wir verwenden zusätzlich eine gröbere Version $\prec_w$ von $\prec$, die sich auf einen Referenzknoten w bezieht und nur Knoten aus P und Q berücksichtigt, deren Relevanz größer als oder gleich w ist:

$$P \prec_w Q \iff P \cap \{x \in V | x \geq w\} \prec Q \cap \{x \in V | x \geq w\}.$$

Die zu $\prec_w$ gehörende Äquivalenzrelation $\sim_w$ ist definiert durch:

$$P \sim_w Q \iff P \cap \{x \in V | x \geq w\} = Q \cap \{x \in V | x \geq w\}.$$

Wir definieren eine lineare Ordnung $\prec$ auf $\mathcal{D}$. Eine Partie ist größer im Bezug auf diese Ordnung, wenn die durch das Profil beschriebenen Partien „besser“ für Spieler 0 sind, und kleiner, wenn sie besser für Spieler 1 sind. Sei (u, P, e), $(v, Q, f) \in \mathcal{D}$:

$$(u, P, e) \prec (v, Q, f) \iff \begin{cases} & u \prec v \\ \vee & (u = v \wedge P \prec Q) \\ \vee & (u = v \wedge P = Q \wedge v \in V_- \wedge e < f) \\ \vee & (u = v \wedge P = Q \wedge v \in V_+ \wedge e > f). \end{cases}$$

Die Idee dieser Definition ist, dass es für Spieler 0 immer besser ist, eine Schleife zu wählen, deren relevantester Knoten ein höheren Reward hat ($u \prec v$). Sind die relevantesten Schleifenknoten gleich, so wird nach den Knoten auf dem Weg zur jeweiligen Schleife unterschieden, die relevanter als der relevanteste Schleifenknoten sind. Unter diesen Knoten ist wiederum der relevanteste, der nicht in beiden Profilen auftritt, entscheidend ($P \prec Q$).

Es verbleibt der Fall, in dem es auch auf dem Weg zu den Schleifen keinen Unterschied im Bezug auf die relevanteren Knoten gibt (d.h. $u = v$ und $P = Q$). Hier ist es anschaulich besser für Spieler 0, wenn er den längsten Weg nimmt, um einen negativen relevantesten Schleifenknoten ($v \in V_-$) zu erreichen, und den kürzesten Pfad, um einen positiven relevantesten Schleifenknoten ($v \in V_+$) zu erreichen.[2]

Für die folgenden Lemmata benötigen wir eine gröbere Ordnung $\prec_w$ der Partieprofile:

$$(u, P, e) \prec_w (v, Q, f) \iff u \prec v \vee (u = v \wedge P \prec_w Q).$$

Für diese Ordnung ist weder die Distanz zum relevantesten Knoten der Schleife erheblich noch die Knoten auf dem Weg zu w, die weniger relevant als w sind. Die zu $\prec_w$ gehörende Äquivalenzrelation $\sim_w$ ist entsprechend definiert durch:

$$(u, P, e) \sim_w (v, Q, f) \iff u = v \wedge P \sim_w Q.$$

[2] Anschaulich: Spieler 0 versucht den Verlust des Spiels (negativer relevantester Schleifenknoten) durch möglichst lange Umwege hinauszuzögern. Kann er dagegen das Spiel gewinnen (positiver relevantester Schleifenknoten), so wählt den kürzesten Weg zur Schleife.

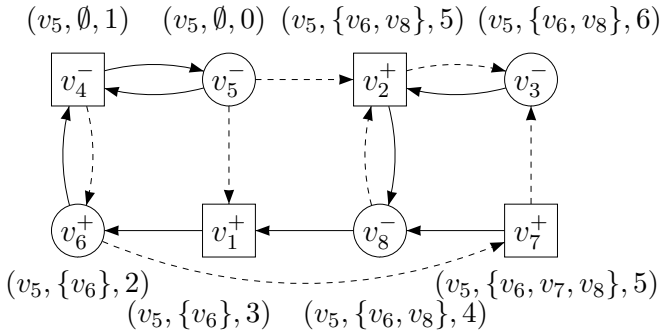


Abbildung 3.4: Spielgraph G_1 mit durch Strategien für beide Spieler induzierter Bewertung.

Beispiel

Für den Spielgraphen in Abbildung 3.4, in dem für beide Spieler Strategien festgelegt sind (durchgezogene Kanten), ergibt sich folgende Anordnung der Partieprofile:

$$\begin{array}{ccccccc} & (v_5, \{v_6, v_8\}, 4) & \prec & (v_5, \{v_6, v_8\}, 5) & \prec & (v_5, \{v_6, v_8\}, 6) \\ \prec & (v_5, \{v_6, v_7, v_8\}, 5) & \prec & (v_5, \emptyset, 0) & \prec & (v_5, \emptyset, 1) \\ \prec & (v_5, \{v_6\}, 2) & \prec & (v_5, \{v_6\}, 3). & & \end{array}$$

3.2 Optimale Bewertungen

Wir definieren die spezielle Klasse der optimalen Bewertungen und zeigen, dass sie nur Profile von solchen Partien enthalten, die nach Gewinnstrategien gespielt worden sind. In einem Unterabschnitt zeigen wir, wie sich eine gegebene Bewertung im Bezug auf einen Spieler verbessern lässt.

Eine Bewertung φ wird optimal für Spieler 0 heißen, wenn die φ-progress-Relation $x \lhd_\varphi y$ nur für Kanten xEy gilt, deren $\varphi(y)$- Wert $\preceq$-maximal unter den E-Nachfolgern von x ist (d.h. unter den $\varphi(z)$ mit xEz). Dabei wird eine geringere Anforderung gestellt, wenn $x = \varphi_0(x)$ ist. In diesem Fall wird die Entfernungskomponente φ_2 nicht berücksichtigt. (Es gilt $(u, P, e) \preceq_x (v, Q, f)$, wenn $u \prec v$ oder wenn $u = v$ und $P \preceq_x Q$; und e und f bleiben unberücksichtigt.)

Formal wird φ *optimal für Spieler* 0 genannt, wenn für alle $x \in V_0$ und alle $y \in V$ mit einer Kante xEy:

$$x \lhd_\varphi y \iff \forall z \in V\colon xEz \Rightarrow \varphi(z) \preceq \varphi(y) \ \vee \ (\varphi_0(x) = x \ \wedge \ \varphi(z) \preceq_x \varphi(y)).$$

Entsprechend heißt φ *optimal für Spieler* 1, wenn für alle $x \in V_1$ und alle $y \in V$

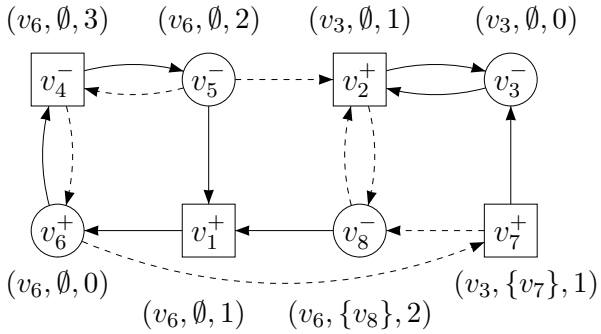


Abbildung 3.5: Spielgraph mit einer optimalen Bewertung.

mit einer Kante xEy:

$$x \lhd_\varphi y \iff \forall z \in V\colon xEz \Rightarrow \varphi(y) \preceq \varphi(z) \ \vee \ (\varphi_0(x) = x \ \wedge \ \varphi(y) \preceq_x \varphi(z)).$$

Eine Bewertung heißt *optimal*, wenn sie optimal für beide Spieler ist.

Man beachte folgendes: Wenn φ optimal für Spieler 0 und $x \in V_0$ ist, dann gilt (da $\preceq_x$ eine gröbere Relation als $\preceq$ ist) $x \lhd_\varphi y$ impliziert $\varphi(z) \preceq_x \varphi(y)$ für xEz. Entsprechend gilt, wenn φ optimal für Spieler 1 und $x \in V_1$ ist, dass $x \lhd_\varphi y$ auch $\varphi(y) \preceq_x \varphi(z)$ für xEz impliziert.

Beispiel

In Abbildung 3.4 ist eine Bewertung von G_1 angegeben, die optimal für Spieler 1 ist. Sie ist nicht optimal für Spieler 0, weil $v_5 \lhd_\varphi v_4$ gilt, aber nicht $\varphi(v_1) \preceq_x \varphi(v_4)$.

Abbildung 3.5 gibt eine optimale Bewertung des Spielgraphen G_1 an. Von den Knoten v_1, v_4, v_5, v_6, v_8 aus gewinnt Spieler 0 und von den Knoten v_2, v_3, v_7 aus gewinnt Spieler 1. Die Strategien beider Spieler sind auf den jeweiligen Gewinnbereichen Gewinnstrategien, d.h. der jeweilige Spieler kann mit seiner Strategie unabhängig von der Strategiewahl des Gegners gewinnen. Dass diese Eigenschaft allgemein für optimale Bewertungen gilt, soll nachfolgend gezeigt werden.

Optimale Bewertungen haben eine direkte Verbindung mit der gewünschten Lösung des Spiels, einem Paar von Gewinnstrategien. Betrachtet man eine Bewertung φ, die optimal für Spieler 0 ist, und sei W_1 die Menge der Knoten u, für welche $\varphi_0(u) \in V_-$. Dann gilt für jede Partie, die in W_1 beginnt und in der Spieler 1 nur $\lhd_\varphi$-Züge macht, dass er diese Partie gewinnt.

Lemma 3.3 *Sei (V, E) ein Spielgraph mit einer Aufteilung der Knotenmenge $V = V_0 \dot\cup V_1$ auf beide Spieler und einer Färbung $c : V \to [|V| + 1]$. Sei φ eine Bewertung von G, die optimal für Spieler 0 ist. Sei $W_1 = \{v \in V \mid \varphi_0(v) \in V_-\}$. Dann gewinnt Spieler 1 jede Partie, die in W_1 beginnt und 1-verträglich mit φ ist.*

Anschaulich bedeutet dieses Lemma, dass, wenn Spieler 0 von einem Knoten aus optimal gegen eine Strategie von Spieler 1 spielt und trotzdem verliert, der Gewinn von diesem Knoten aus für Spieler 1 für die von ihm gewählte Strategie sicher ist.[3]

Durch Anwendung des Lemmas auf das duale Spiel ergibt sich entsprechend für Spieler 1:

Folgerung 3.4 *Sei (V, E) ein Spielgraph mit einer Aufteilung der Knotenmenge $V = V_0 \dot\cup V_1$ auf beide Spieler und einer Färbung $c : V \to [|V| + 1]$. Sei φ eine Bewertung von G, die optimal für Spieler 1 ist. Sei $W_0 = \{v \in V \mid \varphi_0(v) \in V_+\}$. Dann gewinnt Spieler 0 jede Partie, die in W_0 beginnt und 0-verträglich mit φ ist.*

Beweis von Lemma 3.3: Sei $G = (V, E)$ ein Spielgraph mit einer Aufteilung der Knotenmenge $V = V_0 \dot\cup V_1$ auf beide Spieler und einer Färbung $c : V \to [|V| + 1]$, und sei φ eine Bewertung, die optimal für Spieler 0 ist. Sei $W_1 = \{v \in V \mid \varphi_0(v) \in V_-\}$. Sei π eine Partie, die 1-verträglich mit φ ist, wobei v der Startknoten von π ist.

Zu zeigen: Spieler 1 gewinnt die Partie π.

Sei $L = \mathit{Inf}(\pi)$, die Menge der Schleifenknoten von π, und ν die Nachfolgerfunktion auf π. Sei $w = max_< L$ und $x = \varphi_0(w)$. Beachte, dass $\varphi_0(w)$ und w verschieden sein können, da die Partie nicht 0-verträglich sein muss (d.h. die Schleife L kann verschieden von allen in φ kodierten Schleifen sein).

Die Bewertung φ ist optimal für Spieler 0, und π ist 1-verträglich mit φ. Wir beginnen mit zwei Hilfsbehauptungen:

$$\varphi_0(\nu(u)) \preceq \varphi_0(u) \qquad \text{für alle } u \in V \tag{3.2}$$

woraus sich durch Transitivität Folgendes ergibt:

$$\varphi_0(u) = x \qquad \text{für alle } u \in L;$$

und weiterhin

$$\varphi_1(\nu(u)) \preceq \varphi_1(u) \setminus \{u\} \qquad \text{für alle } u \in L \tag{3.3}$$

Behauptung (3.2) gilt im Fall $u \in V_0$, da ein Knoten $y \in V$ mit uEy und $u \lhd_\varphi y$ existiert (da φ eine Bewertung ist), so dass sich aus der Optimalität von φ für Spieler 0 ergibt, dass $\varphi(\nu(u)) \preceq_u \varphi(y)$ (beachte, dass $\preceq$ auch $\preceq_u$ impliziert). Weiterhin gilt nach Definition von $\preceq_u$, dass $\varphi_0(\nu(u)) \preceq \varphi_0(y)$.
Aus $u \lhd_\varphi y$ ergibt sich auch $\varphi_0(u) = \varphi_0(y)$, woraus sich Behauptung (3.2) im Fall

[3] Anders formuliert: Gewinnt Spieler 0 eine Partie, indem er lokal gute Züge macht (lokal optimal spielt), so sagt dies nichts über die Qualität seiner verwendeten Strategie im Allgemeinen. Verliert Spieler 0 eine Partie, in der er lokal gute Züge macht, so bedeutet dies, dass Spieler 1 nach einer universell guten Strategie (einer Gewinnstrategie) gespielt hat.

$u \in V_0$ ergibt. Wenn $u \in V_1$ gilt, lässt sich anwenden, dass Spieler 1 verträglich mit φ spielt; es gilt $u \lhd_\varphi \nu(u)$ und daraus folgend $\varphi_0(u) = \varphi_0(\nu(u))$ und somit (3.2).

Behauptung (3.3) ist offensichtlich im Fall $u \in V_1$, da Spieler 1 verträglich mit φ spielt (und, da $\varphi_0(\nu(u)) = \varphi_0(u)$, erhalten wir $\varphi_1(\nu(u)) = \varphi_1(u) \setminus \{u\}$). Für $u \in V_0$ sei $y \in V$ mit uEy und $u \lhd_\varphi y$ wie im Beweis von (3.2). Dann gilt $\varphi_0(y) = \varphi_0(u)$ und $\varphi_1(y) = \varphi_1(u) \setminus \{u\}$. Weil φ optimal für Spieler 0 ist, gilt $\varphi(\nu(u)) \preceq_x \varphi(y)$. Es folgt $\varphi_1(\nu(u)) \preceq \varphi_1(u) \setminus \{u\}$, weil sich aus Behauptung (3.2) ableiten lässt, dass auf L der φ_0-Wert x ist und somit $\varphi_0(\nu(u)) = \varphi_0(u)$.

Wir zeigen die Behauptung des Lemmas, indem wir $w \in V_-$ zeigen. Dazu unterscheiden wir zwei Fälle:

Fall w > x: Aus $w = \max_< L$ und (3.3) folgt für alle $u \in L \setminus \{w\}$: $\varphi(\nu(u)) \preceq_w \varphi(u)$ und damit $\varphi(w) \preceq_w \varphi(\nu(w))$. Sei $y \in V$ mit wEy und $w \lhd_\varphi y$. Es gilt die Behauptung:

$$\varphi(\nu(w)) \preceq_w \varphi(y) \tag{3.4}$$

aus der sich durch Transitivität $\varphi(w) \preceq_w \varphi(y)$ ergibt. Um (3.4) im Fall $w \in V_0$ zu beweisen, benutzen wir wEy und $w \lhd_\varphi y$ für Behauptung (3.2), so dass sich aus der Optimalität von φ für Spieler 0 ergibt, dass $\varphi(\nu(w)) \preceq_w \varphi(y)$. Wenn $w \in V_1$, nutzen wir aus, dass Spieler 1 verträglich mit φ spielt, woraus $w \lhd_\varphi \nu(w)$ folgt, dass zusammen mit $w \lhd_\varphi y$ auch $\varphi(\nu(w)) \sim_w \varphi(y)$ ergibt.

Wie erwähnt wissen wir aus Behauptung (3.4), dass $\varphi(w) \preceq_w \varphi(y)$, und im aktuellen Fall $w > x$ und $x = \varphi_0(w)$, auch $\varphi(w) \prec_w \varphi(y)$ gilt. Aus wEy und $w \lhd_\varphi y$ erhalten wir $\varphi_0(w) = \varphi_0(y)$ und somit gilt auch $\varphi_1(w) \prec_w \varphi_1(y)$. Ebenfalls wegen wEy und $w \lhd_\varphi y$, können wir $\varphi_1(w)$ als $\varphi_1(y) \dot\cup \{w\}$ formulieren, so dass wir auch $\varphi_1(w) \prec \varphi_1(y)$ erhalten, wobei die symmetrische Differenz beider Mengen $\{w\}$ ist. Aus der Definition von $P \prec Q$ erhalten wir somit $w \in V_-$.

Fall w ≤ x: Wir beginnen mit einer Vorbemerkung. Weil $w = max_< L$ folgt, dass für alle $u \in L$ auch $u \leq x$ gilt. Somit gilt für alle $u \in L : u \notin \varphi_1(u)$ und mit (3.3) auch für alle $u \in L$: $\varphi(\nu(u)) \preceq_x \varphi(u)$. Durch Transitivität und weil L eine Schleife ist, gilt

$$\varphi(u) \sim_x \varphi(u') \text{ für alle } u, u' \in L; \tag{3.5}$$

d.h. die φ_0 und φ_1-Komponenten von u, u' stimmen überein.

Unser Ziel ist es zu zeigen, dass x (der φ_0-Wert des relevantesten Knotens w der Schleife L) zu L gehört, so dass sich daraus $x = w$ ergibt. Wir erreichen dies, indem wir einen Knoten $y \in L$ wählen und dann $y = x$ zeigen: Sei $y \in L$ ein Knoten mit $\varphi_2(\nu(y)) = max_{u \in L} \varphi_2(u)$. Für dieses y wähle z mit yEz und $y \lhd_\varphi z$.

Wir zeigen $x = y$ zunächst für den Fall $\varphi_2(\nu(y)) > \varphi_2(z)$. Aus $y \leq w$ (w ist der relevanteste Knoten in L) und, weil wir im Fall $w \leq x$ sind, gilt $y \leq x$. Wir erhalten $\varphi(\nu(y)) \sim_x \varphi(z)$. (Dies ergibt sich daraus, dass $\varphi_0(z) = \varphi_0(y) = \varphi_0(\nu(y))$, weil $y \lhd_\varphi z$ und aus (3.2.) Somit gilt $\varphi_1(z) = \varphi_1(y) \sim_x \varphi_1(\nu(y))$: Die Gleichheit folgt aus den ersten beiden Termen der Definition von $y \lhd_\varphi z$ unter

Ausnutzung von $\varphi_0(y) = \varphi_0(z)$ und $y \leq x$; die $\sim_x$-Äquivalenz wurde bereits in (3.5) für die Knoten der Schleife L gezeigt.) Es bleibt zu zeigen:

$$\varphi_2(\nu(y)) > \varphi_2(z) \Longrightarrow x = y.$$

Im Fall $y \in V_1$ gilt $y \lhd_\varphi \nu(y)$, weil Spieler 1 verträglich mit φ spielt. Zusammen mit $y \lhd_\varphi z$ und unserer Voraussetzung $\varphi_2(\nu(y)) > \varphi_2(z)$ erhalten wir nach Definition von $\lhd_\varphi$, dass $\varphi_0(y) = y$. Somit erhalten wir $x = \varphi_0(y) = y$.

Nun nehmen wir $y \in V_0$ an und erinnern uns daran, dass die Partie π in $v \in W_1$ beginnt, d.h. $\varphi_0(v) \in V_-$. Aus Behauptung (3.2) und Transitivität erhalten wir $\varphi_0(w) \preceq \varphi_0(v)$. Es ergibt sich $x = \varphi_0(w) \in V_-$. Mit $x \in V_-$ und der Voraussetzung $\varphi_2(z) < \varphi_2(\nu(y))$ erhalten wir $\varphi(z) \prec \varphi(\nu(y))$, weil die φ_0- und φ_1-Werte, wie gezeigt, übereinstimmen. Nutzt man aus, dass φ optimal für Spieler 0 ist, so liefert die Definition von $\lhd_\varphi$ zu $y = \varphi_0(y) = x$.

Im anderen Fall $\varphi_2(\nu(y)) \leq \varphi_2(z)$ erhalten wir mit der Maximalität von $\varphi_2(\nu(y))$ auf L auch $\varphi_2(y) \leq \varphi_2(\nu(y)) \leq \varphi_2(z)$. Somit ist die Distanz von y zu x in Partien verträglich mit φ höchstens die Distanz von z zu x. Da $y \lhd_\varphi z$ muss $y = x$ gelten.

Also ist $y = x$ und somit $x \in L$. Mit $w \leq x$ und $w = \max_< L$ erhalten wir $w = x \in V_-$. □

Wendet man dieses Lemma symmetrisch für beide Spieler an, so erhält man den folgenden Satz:

Satz 3.5 *Sei $G = (V, E)$ ein Spielgraph mit einer Aufteilung der Knotenmenge $V = V_0 \dot\cup V_1$ auf beide Spieler und einer Färbung $c : V \to [|V| + 1]$. Sei φ eine optimale Bewertung von G. Dann sind alle Strategien, die verträglich mit φ sind, Gewinnstrategien.*

Beweis: Sei $G = (V, E)$ ein Spielgraph mit einer Aufteilung der Knotenmenge $V = V_0 \dot\cup V_1$ auf beide Spieler und einer Färbung $c : V \to [|V| + 1]$. Sei $W_0 = \{v \in V \mid \varphi_0(v) \in V_+\}$ und $W_1 = \{v \in V \mid \varphi_0(v) \in V_-\}$. Sei $\sigma : V_0 \to V_1$ eine Strategie für Spieler 0, die mit φ verträglich ist, und $\tau : V_1 \to V_0$ eine Strategie von Spieler 1, die mit φ verträglich ist.

Wendet man Lemma 3.3 und Folgerung 3.4 an, so folgt, dass σ eine Gewinnstrategie für Spieler 0 auf W_0 und τ eine Gewinnstrategie für Spieler 1 auf W_1 ist. Die Mengen W_0 und W_1 sind die Gewinnbereiche des Spiels G, weil sie V partitionieren und Spieler 0 auf W_0 gewinnt und Spieler 1 auf W_1 gewinnt. Somit sind σ und τ Gewinnstrategien. □

3.2.1 Verbessern von Bewertungen

Unser Vorgehen zum Bestimmen optimaler Bewertungen ist, von einer Anfangsbewertung ausgehend, diese schrittweise bis zum Optimum zu verbessern.

Es seien zwei Bewertungen φ und φ' gegeben. Wir bezeichnen φ' als *verbessert für Spieler* 0 *im Bezug auf* φ, wenn

$$\forall x \in V_0 \; \exists y \in V : xEy \wedge x \lhd_{\varphi'} y \wedge \forall z \in V : xEz \Rightarrow \varphi(z) \preceq \varphi(y).$$

Eine Bewertung φ', die für Spieler 0 im Bezug auf φ verbessert ist, kann dadurch aus der gegeben Bewertung φ konstruiert werden, dass man eine Strategie $\sigma : V_0 \to V_1$ für Spieler 0 definiert, die im Bezug auf φ jeweils maximale E-Nachfolger wählt, und daraus eine Bewertung φ' konstruiert, so dass σ verträglich mit dieser Bewertung φ' ist.

Beispiel

Die Bewertung des Graphen G_1 in Abbildung 3.4 ist für Spieler 0 verbessert im Bezug auf die Bewertung in Abbildung 3.3. Ebenso ist auch die Bewertung in Abbildung 3.5 für Spieler 0 verbessert im Bezug auf die Bewertung in Abbildung 3.4. Dabei sind jeweils die Strategien für Spieler 0, mit denen die verbesserte Bewertung konstruiert wurde, mit durchgezogenen Kanten angegeben.

Man beachte, dass eine Bewertung φ, die verbessert für Spieler 0 im Bezug auf sich selbst ist, eine Bewertung ist, die optimal für Spieler 0 ist.

Lemma 3.6 *Sei $G = (V, E)$ ein Spielgraph mit einer Aufteilung der Knotenmenge $V = V_0 \dot\cup V_1$ auf beide Spieler und einer Färbung $c : V \to [|V|+1]$, und sei φ eine Bewertung, die optimal für Spieler 1 ist und φ' eine Bewertung, die für Spieler 0 im Bezug auf φ verbessert ist. Dann gilt für alle $v \in V$: $\varphi(v) \preceq \varphi'(v)$.*

Beweis: Sei $G = (V, E)$ ein Spielgraph mit einer Aufteilung der Knotenmenge $V = V_0 \dot\cup V_1$ auf beide Spieler und einer Färbung $c : V \to [|V| + 1]$. Sei φ eine Bewertung, die optimal für Spieler 1 ist und φ' eine Bewertung, die für Spieler 0 im Bezug auf φ verbessert ist.

Wir gehen von einem speziellen Strategiepaar (σ, τ) aus, das φ' induziert. Dies ist keine Einschränkung, da jede Bewertung von mindestens einem Strategiepaar induziert wird. Dann zeigen wir die Behauptung durch Induktion über die φ'_2-Werte (d.h. die Distanzkomponente der Partieprofile) der Knoten, indem wir uns auf die Partie beschränken, die nach dem Strategiepaar (σ, τ) gespielt wird.

Da φ' eine für Spieler 0 im Bezug auf φ verbesserte Bewertung ist, können wir eine Strategie $\sigma : V_0 \to V_1$ wählen mit

$$vE\sigma(v) \wedge v \lhd_{\varphi'} \sigma(v) \wedge \forall x \in V_1 : vEx \Rightarrow \varphi(x) \preceq \varphi(\sigma(v)).$$

Man beachte, dass diese Definition von σ verträglich mit φ' ist. Es existiert eine Strategie τ für Spieler 1, die verträglich mit φ' ist, weil φ' eine Bewertung ist. Im Folgenden kombinieren wir das Strategiepaar (σ, τ) zu einer Nachfolgerfunktion $\nu = \sigma \cup \tau$.

Zunächst beweisen wir eine wichtige Eigenschaft von ν. Für alle $v \in V$ und alle $x \in V$ gilt:

$$vEx \wedge v \lhd_\varphi x \implies \varphi(x) \preceq \varphi(\nu(v)) \ \vee \ (\varphi_0(v) = v \ \wedge \ \varphi(x) \preceq_v \varphi(\nu(v))). \quad (3.6)$$

Wenn $v \in V_0$, gilt $vEx \implies \varphi(x) \preceq \varphi(\nu(v))$ nach Definition von σ, was die Behauptung impliziert. Fall $v \in V_1$: Wir wissen, dass φ optimal für Spieler 1 ist, was bedeutet, dass für alle $v \in V_1$ und $x \in V_0$ mit einer Kante vEx gilt:

$$v \lhd_\varphi x \iff \forall y \in V_0 : vEy \Rightarrow \varphi(x) \preceq \varphi(y) \ \vee \ (\varphi_0(v) = v \ \wedge \ \varphi(x) \preceq_v \varphi(y)).$$

Wählt man $\nu(v)$ für y, so erhält man die Behauptung (3.6). Im Folgenden benötigen wir auch eine schwächere Version der Behauptung (3.6):

$$vEx \wedge v \lhd_\varphi x \implies \varphi(x) \preceq_v \varphi(\nu(v)) \qquad \text{für alle } v, x \in V \quad (3.7)$$

Sie folgt aus (3.6), weil $\preceq$ auch $\preceq_v$ impliziert.

Induktionsanfang $\boldsymbol{\varphi_2'(\mathbf{v}) = 0}$:

Sei π die Partie, die nach ν beginnend in v gespielt ist. Sei $L = \mathit{Inf}(\pi)$ die Schleife dieser Partie. Aus $\varphi_2'(v) = 0$ folgt, da φ' eine Bewertung ist, dass $\varphi'(v) = (v, \emptyset, 0)$. Weil π eine Partie ist, die nach Strategien gespielt wird, die verträglich mit φ' sind, wissen wir, dass v der relevanteste Knoten in der Schleife dieser Partie ist:

$$v = \max_< L \quad (3.8)$$

Wir unterscheiden drei Fälle für den Wert von v:

Fall $\boldsymbol{\varphi_0(\mathbf{v}) = \mathbf{v}}$: Es gilt $\varphi(v) = (v, \emptyset, 0) = \varphi'(v)$, was $\varphi(v) \preceq \varphi'(v)$ impliziert.

Fall $\boldsymbol{\varphi_0(\mathbf{v}) < \mathbf{v}}$: Wir beweisen zunächst für alle $u \in L \setminus \{v\}$: $\varphi(u) \sim_v \varphi(\nu(u))$. Damit werden wir zeigen, dass $\varphi(\nu(v)) \preceq_v \varphi(v)$. Daraus ergibt sich $v \in V_+$, was zur Behauptung des Lemmas führt.

Wir zeigen nun für alle $u \in L \setminus \{v\}$:

$$\varphi(u) \sim_v \varphi(\nu(u)).$$

Sei $u \in L \setminus \{v\}$. Weil φ eine Bewertung ist, existiert ein Knoten $x \in V$ mit uEx und $u \lhd_\varphi x$. Nach (3.8) gilt $u < v$ und damit folgt aus der Definition von $u \lhd_\varphi x$, dass $\varphi(u) \sim_v \varphi(x)$. Aus Behauptung (3.7) erhalten wir $\varphi(x) \preceq_u \varphi(\nu(u))$. Also erhalten wir $\varphi(u) \sim_v \varphi(x) \preceq_u \varphi(\nu(u))$. Mit $u < v$ ergibt sich, dass $\preceq_u$ auch $\preceq_v$ impliziert, und somit $\varphi(u) \preceq_v \varphi(\nu(u))$ gilt.

Daraus ergibt sich durch Transitivität: $\varphi(\nu(v)) \preceq_v \varphi(\nu^{|L|}(v))$, weil die Partie, die nach ν gespielt ist, eine Schleife ergibt mit der Menge L an Schleifenknoten, was bedeutet, dass $\nu^i(v) \neq v$ für alle i mit $1 \leq i < |L|$ gilt. Außerdem wissen wir, dass $\nu^{|L|}(v) = v$ gilt. Somit erhalten wir:

$$\varphi(\nu(v)) \preceq_v \varphi(v). \quad (3.9)$$

Weil φ eine Bewertung ist, existiert ein Knoten $y \in V$ mit vEy und $v \lhd_\varphi y$. Mit der Voraussetzung $\varphi_0(v) < v$ des aktuellen Falls erhalten wir aus der Definition von $v \lhd_\varphi y$, dass $\varphi_0(v) = \varphi_0(y)$ und $\varphi_1(v) = \varphi_1(y) \dot\cup \{v\}$. Aus vEy und $v \lhd_\varphi y$ erhalten wir mit (3.7), dass $\varphi(y) \preceq_v \varphi(\nu(v))$ gilt, und mit (3.9) (und Transitivität), auch $\varphi(y) \preceq_v \varphi(v)$. Aus $\varphi_0(v) = \varphi_0(y)$ erhalten wir $\varphi_1(y) \preceq_v \varphi_1(v)$. Zusammen mit $\varphi_1(v) = \varphi_1(y) \dot\cup \{v\}$ ergibt sich $\varphi_1(y) \preceq_v \varphi_1(y) \dot\cup \{v\}$. Daraus folgt nach Definition von $\preceq_v$, dass $v \in V_+$ gilt.

Aus $v \in V_+$ und der Voraussetzung $\varphi_0(v) < v$ und der Definition von $\prec$ ergibt sich $\varphi_0(v) \prec v$. Mit der Annahme des Induktionsanfangs, dass $\varphi'_2(v) = 0$ gilt, die auch $\varphi'_0(v) = v$ (nach Definition einer Bewertung φ') impliziert, ergibt sich $\varphi_0(v) \prec v = \varphi'_0(v)$, was die Behauptung des Lemmas $\varphi(v) \preceq \varphi'(v)$ liefert.

Fall $\boldsymbol{\varphi_0(\mathbf{v}) > \mathbf{v}}$: Zunächst beweisen wir, dass $\varphi_0(u) = \varphi_0(\nu(u))$ für alle $u \in L$ gilt. Wir benutzen es, um für alle $u \in L$ auch $u < \varphi_0(u)$ zu zeigen. Damit und einem speziell gewählten Knoten u aus der Schleife L mit einem $\prec$-kleineren Nachfolger, erhalten wir $\varphi_0(v) \in V_-$, das uns direkt die Behauptung des Lemmas liefert.

Zunächst zeigen wir die Hilfsbehauptung:

$$\varphi_0(u) = \varphi_0(\nu(u)) \qquad \text{für alle } u \in L.$$

Sei $u \in L$. Weil φ eine Bewertung ist, existiert ein Knoten $x \in V$ mit uEx und $u \lhd_\varphi x$. Aus (3.7) ergibt sich $\varphi(x) \preceq_u \varphi(\nu(u))$ und somit auch $\varphi_0(x) \preceq \varphi_0(\nu(u))$. Nach Definition von $u \lhd_\varphi x$ gilt $\varphi_0(u) = \varphi_0(x)$. Zusammen mit $\varphi_0(x) \preceq \varphi_0(\nu(u))$ erhalten wir die Behauptung $\varphi_0(u) \preceq \varphi_0(\nu(u))$.

Weil ν eine Schleife mit der Knotenmenge L erzeugt, gilt $\nu^{|L|}(u) = u$ für alle $u \in L$. Mit Transitivität erhalten wir:

$$\forall u \in L : \varphi_0(u) = \varphi_0(\nu(u)).$$

und, weil $v \in L$ ist, mit der vorangegangenen Hilfsaussage:

$$\forall u \in L : \varphi_0(u) = \varphi_0(v). \tag{3.10}$$

Wir zeigen nun:

$$u < \varphi_0(u) \qquad \text{für alle } u \in L \tag{3.11}$$

Sei $u \in L$. Aus (3.10) folgt $\varphi_0(u) = \varphi_0(v)$. Im aktuellen Fall gilt $v < \varphi_0(v)$ und damit auch $v < \varphi_0(u)$. Mit (3.8) ergibt sich $u \leq v$ und damit schließlich $u < \varphi_0(u)$.

Weil ν eine Schleife mit der Knotenmenge L erzeugt, existiert ein $u \in L$ mit $\varphi(\nu(u)) \preceq \varphi(u)$ (das Gegenteil führt durch Transitivität von $\succ$ unmittelbar zu einem Widerspruch). Weil φ eine Bewertung ist, lässt sich ein $x \in V$ mit uEx und $u \lhd_\varphi x$ wählen. Mit (3.6) ergibt sich

$$\varphi(x) \preceq \varphi(\nu(u)) \ \vee \ (\varphi_0(u) = u \ \wedge \ \varphi(x) \preceq_u \varphi(\nu(u))).$$

und aus (3.11) erhalten wir somit

$$\varphi(x) \preceq \varphi(\nu(u)).$$

Nach Voraussetzung gilt $\varphi(\nu(u)) \preceq \varphi(u)$ und somit auch

$$\varphi(x) \preceq \varphi(u).$$

Weil $u \lhd_\varphi x$, gilt auch $\varphi_0(x) = \varphi_0(u)$, und, weil $u < \varphi_0(u)$, gilt $\varphi_1(x) = \varphi_1(u)$. Somit ergibt sich:

$$\varphi_2(x) \preceq \varphi_2(u).$$

Aus $u \lhd_\varphi x$ erhalten wir $\varphi_2(u) = \varphi_2(x)+1$ und auch $\varphi_2(x) < \varphi_2(u)$. Zusammen mit $\varphi_2(x) \preceq \varphi_2(u)$ impliziert dies $\varphi_0(u) \in V_-$ und mit (3.10) auch $\varphi_0(v) \in V_-$. Aus $\varphi_0(v) \in V_-$ und $\varphi_0(v) > v$ erhalten wir $\varphi_0(v) \prec v$. Also gilt $\varphi_0(v) \prec v = \varphi'_0(v)$, und somit erhalten wir die Behauptung des Lemmas $\varphi(v) \preceq \varphi'(v)$.

Induktionsschritt:

Induktionsannahme: $\varphi(v) \preceq \varphi'(v)$ gilt für alle $v \in V$ mit $\varphi'_2(v) = k$.
Um die Induktionsbehauptung zu zeigen nehmen wir an, dass $\varphi'_2(v) = k+1$: Wir wählen einen $\lhd_\varphi$-Nachfolger x und einen $\lhd_{\varphi'}$-Nachfolger y von v. Weil φ eine Bewertung ist, können wir einen Knoten $x \in V$ mit vEx und $v \lhd_\varphi x$ wählen. Sei $y = \nu(v)$. Dann gilt vEy und $v \lhd_{\varphi'} y$, weil beide in ν kombinierten Strategien verträglich mit φ' sind. Im Folgenden werden die Profile $\varphi(x)$ und $\varphi'(y)$ unter Ausnutzung von (3.6) und der Induktionsvoraussetzung verglichen. Mit $v \lhd_\varphi x$ und $v \lhd_{\varphi'} y$ kann die Relation zwischen $\varphi(x)$ und $\varphi'(y)$ auf die zwischen $\varphi(v)$ und $\varphi'(v)$ übertragen werden.

Aus vEx und $v \lhd_\varphi x$ und $y = \nu(v)$ erhalten wir mit (3.7), dass $\varphi_0(x) \preceq \varphi_0(y)$. Nach Induktionsvoraussetzung gilt $\varphi_0(y) \preceq \varphi'_0(y)$. Also folgt durch Transitivität:

$$\varphi_0(x) \preceq \varphi'_0(y).$$

Zunächst betrachten wir den einfacheren Fall $\varphi_0(x) \prec \varphi'_0(y)$: Mit $v \lhd_\varphi x$ gilt auch $\varphi_0(v) = \varphi_0(x)$ und mit $v \lhd_{\varphi'} y$ auch $\varphi'_0(v) = \varphi'_0(y)$. Somit erhalten wir $\varphi_0(v) = \varphi_0(x) \prec \varphi'_0(y) = \varphi'_0(v)$, und dies impliziert die Behauptung des Lemmas $\varphi(v) \preceq \varphi'(v)$.

Es verbleibt der Fall $\varphi_0(x) = \varphi'_0(y)$. Es gilt $\varphi_0(v) = \varphi_0(x) = \varphi'_0(y) = \varphi'_0(v)$. Mit $\varphi'_2(v) = k+1$ erhalten wir $\varphi_0(v) \neq v$. Somit ergibt sich mit (3.6), dass $\varphi(x) \preceq \varphi(y)$. Nach Induktionsvoraussetzung gilt $\varphi(y) \preceq \varphi'(y)$. Also ergibt sich mit Transitivität:

$$\varphi(x) \preceq \varphi'(y). \tag{3.12}$$

Im Folgenden übertragen wir diese Relation auf $\varphi(v)$ und $\varphi'(v)$. Unser erster Schritt ist zu zeigen:

$$\varphi_1(v) \preceq \varphi'_1(v).$$

Aus der Voraussetzung $\varphi'_2(v) = k+1$ des Induktionsschrittes folgt $\varphi_0(v) \neq v$. Somit verbleiben die beiden Fälle $\varphi_0(v) > v$ und $\varphi_0(v) < v$:
Wenn $\varphi_0(v) > v$ ergibt sich aus der Definition von $v \lhd_\varphi x$, dass $\varphi_1(v) = \varphi_1(x)$, und mit $v \lhd_{\varphi'} y$ auch $\varphi'_1(v) = \varphi'_1(y)$. Somit gilt $v \notin \varphi_1(x)$ und $v \notin \varphi'_1(y)$. Daraus ergibt sich mit (3.12) auch $\varphi_1(x) \dot\cup \{v\} \preceq \varphi'_1(y) \dot\cup \{v\}$. Zusammengenommen erhalten wir:

$$\varphi_1(v) = \varphi_1(x) \dot\cup \{v\} \preceq \varphi'_1(y) \dot\cup \{v\} = \varphi'_1(v).$$

Nachdem wir $\varphi_1(v) \preceq \varphi'_1(v)$ gezeigt haben (im aktuellen Fall $\varphi_0(x) = \varphi'_0(y)$) unterscheiden wir zwei Unterfälle. Wenn $\varphi_1(v) \prec \varphi'_1(v)$ folgt die Behauptung des Lemmas $\varphi(v) \preceq \varphi'(v)$, weil wir im Fall $\varphi_0(x) = \varphi'_0(y)$ sind, was bedeutet, dass $\varphi_0(v) = \varphi'_0(v)$ gilt.

Es verbleibt der Unterfall $\varphi_1(v) = \varphi'_1(v)$. Nach Definition von $v \lhd_\varphi x$ gilt $\varphi_1(x) = \varphi_1(v) \setminus \{v\}$ und mit $v \lhd_{\varphi'} y$ gilt $\varphi'_1(v) \setminus \{v\} = \varphi'_1(y)$. Somit gilt $\varphi_1(x) = \varphi'_1(y)$. Berücksichtigen wir, dass wir in Fall $\varphi_0(x) = \varphi'_0(y)$ sind, ergibt sich aus der Definition von (3.12):
Wenn $\varphi_0(v) \in V_-$, dann gilt $\varphi_2(x) \leq \varphi'_2(y)$, und wir erhalten

$$\varphi_2(v) = \varphi_2(x) + 1 \leq \varphi'_2(y) + 1 = \varphi'_2(v).$$

Wenn $\varphi_0(v) \in V_+$, dann gilt $\varphi_2(x) \geq \varphi'_2(y)$, und wir erhalten die umgekehrte Relation

$$\varphi_2(v) = \varphi_2(x) + 1 \geq \varphi'_2(y) + 1 = \varphi'_2(v).$$

Da wir uns im Fall $\varphi_0(v) = \varphi'_0(v)$ und dem Unterfall $\varphi_1(v) = \varphi'_1(v)$ befinden, ergibt sich nach Definition von $\preceq$ (wenn man die beiden Möglichkeiten $\varphi_0(v) \in V_-$ und $\varphi_0(v) \in V_+$ unterscheidet) die Behauptung des Lemmas: $\varphi(v) \preceq \varphi'(v)$. □

3.3 Der Algorithmus (dSVA)

In diesem Abschnitt geben wir den diskreten SVA an. Er liefert eine optimale Bewertung für einen gegebenen Spielgraphen mit Paritätsgewinnbedingung. Daraus ergeben sich nach Satz 3.5 unmittelbar auch Gewinnstrategien für beide Spieler dieses Spiels. Der Algorithmus ist in drei Funktionen aufgeteilt: *main()*, *valuation()*, und *subvaluation()*.

In der Funktion *main()* wird eine Folge von Strategien von Spieler 0 erzeugt und für jede dieser Strategien wird durch Aufruf von *valuation()* eine Bewertung berechnet. Diese Bewertung ist so konstruiert, dass die Strategie von Spieler 0 verträglich mit ihr ist und dass sie optimal für Spieler 1 ist. Die erste Strategie von Spieler 0 wird zufällig gewählt. Die folgenden Strategien von Spieler 0 werden so gewählt, dass sie optimal im Bezug auf die zuletzt berechnete Bewertung sind. Die Hauptschleife terminiert, wenn die für Spieler 0 gewählte Strategie mit in der vorangehenden Iteration gewählten übereinstimmt. Schließlich wird eine Gewinnstrategie für Spieler 1 aus der zuletzt berechneten Bewertung extrahiert.

main(G, V_0, V_1, c)*:*

1. **for each** $v \in V_0$ {Select initial strategy for player 0.}
2. **select** $\sigma(v) \in V_1$ **with** $v\, E_G\, \sigma(v)$
3. **repeat**
4. $G_\sigma = (V_G, E')$, where f.a. $u, v \in V_G$:
   $uE'v \iff (\sigma(u) = v \wedge v \in V_0) \vee (uE_Gv \wedge v \in V_1)$
5. $\varphi = \mathit{valuation}(G_\sigma)$
6. $\sigma' = \sigma$ {Store σ under name σ'}
7. **for each** $v \in V_0$ {Optimize σ locally according to φ}
8. **if** $\varphi(\sigma(v)) < \max\{d \in \mathcal{D} | \exists v' \in V_G : v\, E_G\, v' \wedge d = \varphi(v')\}$ **then**
9. **select** $\sigma(v) \in V_1$
   with $\varphi(\sigma(v)) = \max\{d \in \mathcal{D} | \exists v' \in V_G : v\, E_G\, v' \wedge d = \varphi(v')\}$
10. **until** $\sigma = \sigma'$
11. **for each** $v \in V_1$
12. **select** $\tau(v) \in V_0$
    with $\varphi(\tau(v)) = \min\{d \in \mathcal{D} | \exists v' \in V_G : v\, E_G\, v' \wedge d = \varphi(v')\}$
13. $W_0 = \{v \in V_G \mid \varphi_0(v) \in V_+\}$
14. $W_1 = \{v \in V_G \mid \varphi_0(v) \in V_-\}$
15. **return** W_0, W_1, σ, τ

Abbildung 3.6: Diese Funktion berechnet die Gewinnmengen und Gewinnstrategien für beide Spieler aus einem gegebenen Spielgraph.

In den Funktionen *valuation()* und *subvaluation()*, benutzen wir die Funktionen *reach*(G, u), *minimal_distances*(G, u) und *maximal_distances*(G, u). Diese Funktionen arbeiten auf dem Graphen G. Der Knoten u ist jeweils das ‚Ziel' der entsprechenden Operation.

- Die Funktion *reach*(G, u) berechnet die Menge aller Knoten, von denen aus der Knoten u in G erreicht werden kann. Dies kann durch eine einfache rückwärtsgerichtete Tiefensuche erreicht werden, die von u ausgeht.

- Die Funktion *minimal_distances*(G, u) berechnet einen Vektor $\delta : V_G \to [|V_G|]$, der jedem Knoten v die Länge $\delta(v)$ des kürzesten Pfades von v nach u zuordnet. Dies kann durch eine rückwärtsgerichtete Breitensuche erreicht werden, die von u ausgeht.

- Die Funktion *maximal_distances*(G, u) berechnet einen Vektor $\delta : V_G \to [|V_G|]$, der jedem Knoten v die Länge $\delta(v)$ des längsten Pfades von v nach u zuordnet. Dies kann durch eine rückwärtsgerichtete Suche erreicht werden, die von u ausgeht, wobei ein neuer Knoten jeweils nur dann besucht wird, wenn bereits alle seine Nachfolger besucht wurden. Dieser Algorithmus funktioniert nur, wenn jeder Zykel im Graph den Knoten u enthält.

valuation(H):

1. **for each** $v \in V_H$ **do**
2. $\quad \varphi(v) = \bot$.
3. **for each** $w \in V_H$ (ascending order with respect to $\prec$) **do**
4. $\quad$ **if** $\varphi(w) = \bot$ **then**
5. $\quad\quad L = \ reach(H|_{\{v \in V_H | v \leq w\}}, w)$
6. $\quad\quad$ **if** $E_H \cap \{w\} \times L \neq \emptyset$ **then**
7. $\quad\quad\quad R = \ reach(H, w)$
8. $\quad\quad\quad \varphi|_R = subvaluation(H|_R, w)$
9. $\quad\quad\quad E_H = E_H \ \setminus \ (R \times (V_H \setminus R))$
10. **return** φ

Abbildung 3.7: Diese Funktion berechnet für einen gegebenen Graph eine Bewertung, die optimal für Spieler 1 ist.

Die Funktion *valuation(H)* berechnet eine Bewertung für den Spielgraphen H, die optimal für Spieler 1 ist. Der Graph wird dazu in mehrere Teilgraphen zerlegt, für die jeweils mit Hilfe der Funktion *subvaluation()* eine Bewertung bestimmt wird.

Die Funktion sucht die Schleife mit dem besten relevantesten Knoten für Spieler 1 und dann die Menge aller Knoten R, von welchen aus diese Schleife erreichbar ist. Die Bewertung für den Teilgraph, der durch R induziert wird, wird durch Aufruf von *subvaluation()* berechnet. Der restliche Graph, der durch $V_H \setminus R$ induziert wird, wird in derselben Weise bearbeitet, bis der ursprüngliche Graph in Teilgraphen zerlegt ist, die durch *subvaluation()* bewertet sind.

Die Funktion *valuation(H)* aus Abbildung 3.7 funktioniert wie folgt:

1. φ wird für alle Knoten auf den Wert ‚undefiniert' gesetzt.

2. In absteigender Relevanzordnung werden die Knoten w gefunden, die zu einer Schleife L gehören, die nur aus $\leq$-kleineren Knoten besteht. Zu jedem solchen Knoten w wird die Menge R_w aller Knoten bestimmt, von denen aus dieser Knoten v und damit die zugehörige Schleife L erreichbar ist. Dabei bleiben jeweils die Knoten, die in Mengen $R_{w'}$ mit einem früher berechneten w' liegen, unberücksichtigt. Die Mengen R_w (wobei w relevantester Schleifenknoten ist) bilden somit eine Partition von V. Durch den Aufruf von *subvaluation()* für jedes w werden die Bewertungen aller Knoten in R_w berechnet.

   Die nächste Bewertung wird wie folgt berechnet: Ist eine Menge R_w berechnet, so werden alle ausgehenden Kanten entfernt. Dadurch wird verhindert, dass man von Knoten außerhalb von R_w durch Rückwärtssuchen zu Knoten innerhalb von R_w gelangen kann.

subvaluation(K, w)*:*

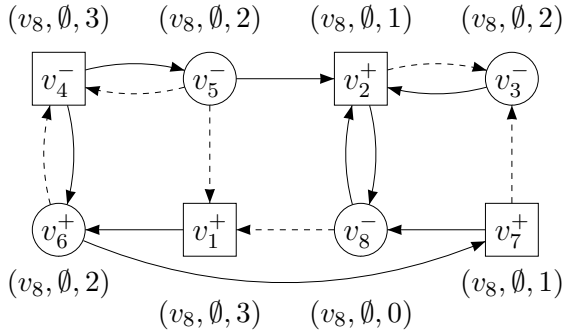
1. **for each** $v \in V_K$ **do**
2. $\quad \varphi_0(v) = w$
3. $\quad \varphi_1(v) = \emptyset$
4. **for each** $u \in \{v \in V_K \mid v > w\}$ (descending order with respect to $<$) **do**
5. $\quad$ **if** $u \in V_+$ **then**
6. $\quad\quad \bar{U} = \ reach(K|_{V_K \setminus \{u\}}, w)$
7. $\quad\quad$ **for each** $v \in V_K \setminus \bar{U}$ **do**
8. $\quad\quad\quad \varphi_1(v) = \varphi_1(v) \cup \{u\}$
9. $\quad\quad E_K = E_K \setminus ((\bar{U} \cup \{u\}) \times (V_K \setminus \bar{U}))$
10. $\quad$ **else**
11. $\quad\quad U = \ reach(K|_{V_K \setminus \{w\}}, u)$
12. $\quad\quad$ **for each** $v \in U$ **do**
13. $\quad\quad\quad \varphi_1(v) = \varphi_1(v) \cup \{u\}$
14. $\quad\quad E_K = E_K \setminus ((U \setminus \{u\}) \times (V_K \setminus U))$
15. **if** $w \in V_+$ **then**
16. $\quad \varphi_2 = \ maximal\_distances(K, w)$
17. **else**
18. $\quad \varphi_2 = \ minimal\_distances(K, w)$
19. **return** φ

Abbildung 3.8: Diese Funktion berechnet eine Bewertung auf einem Teilgraph.

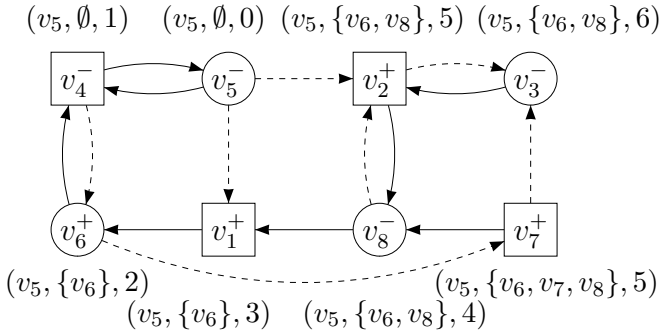
In *subvaluation(H)*wird die Rolle von Knoten v in R_w in Bezug auf $\prec_w$ analysiert. Zunächst wird für alle Knoten $v \in R_w$ auch $\varphi_0(v) = w$ gesetzt. Dann werden alle Knoten $u \in R_w$, die relevanter als w sind, in absteigender Relevanzordnung bearbeitet:

- Ist $u \in V_+$, so wird die Menge $\bar{U}$ aller Knoten berechnet, von denen aus w erreichbar ist, ohne dabei u zu besuchen. Für die übrigen Knoten v, von denen aus sich u nicht vermeiden lässt, wird u zu $\varphi_1(v)$ hinzugefügt und alle Kanten, die zu Knoten in $\bar{U}$ führen, werden entfernt.
- Ist $u \in V_-$, so wird für jeden Knoten v, von dem aus u erreichbar ist, u zu $\varphi_1(v)$ hinzugefügt und alle ausgehenden Kanten von v, die zu Knoten führen, von denen aus u nicht erreichbar ist (ohne w zu besuchen), werden entfernt.

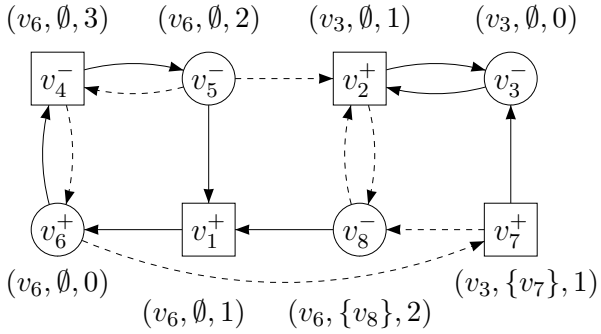
Die Funktion *subvaluation(H)* wird in Abbildung 3.8 dargestellt. Sie berechnet die Bewertungen für alle Knoten des gegebenen Teilgraphen. Das bedeutet, dass sie für alle Knoten jeweils einen Weg zu w mit $\prec$-minimalen Kosten berechnet.



a) Nach der Bewertung der initialen Strategie (erste Iteration).



b) Bewertung in der zweiten Iteration.



c) Bewertung in der dritten und auch letzten Iteration.

Abbildung 3.9: Iterationsschritte des Strategiesynthese-Algorithmus für den Spielgraph G_1.

Beispiel

In Abbildung 3.9 wird der Ablauf des Algorithmus für den Spielgraph G_1 dargestellt. Es wird jeweils die Situation nach der Berechnung der Bewertungsfunktion dargestellt, d.h. nach Ausführung der Zeile 5 in *main()*. Für Spieler 0 markieren die durchgezogenen Kanten die für ihn vor der Bewertung gewählte Strategie σ und für Spieler 1 die möglichen Wahlen einer Gegenstrategie.

In a) kann Spieler 0 entsprechend der Bewertung so ziehen, dass er immer in eine Schleife mit den Knoten v_2 und v_8 kommt. In b) hat Spieler 0 seine Strategie in Knoten v_8 verbessert, indem er von Knoten v_8 zu v_1 zieht und auf diese Weise den Weg zum relevantesten Schleifenknoten v_8 verlängert.[4] Ebenfalls kann er den Weg zu v_8 von v_6 aus verlängern, indem er zu v_4 zieht. Aus demselben Grund zieht er auch von v_5 zu v_4 (hier wäre es auch möglich von v_5 nach v_1 zu ziehen, was unmittelbar zu c) führt. Die beiden Züge sind gleichwertig, da beide die Distanz zu v_8 auf 3 vergrößern.)

Bei der Verbesserung der Strategie für Spieler 0 ist zu beachten, dass sie für alle Knoten zugleich durchgeführt wird ohne die Wechselwirkungen zu beachten. So ist in a) aus globaler Sicht durch die neue Wahl der Züge von v_5 und v_6 zu v_4 hin klar, dass der ursprüngliche Schleifenknoten v_8 nicht mehr erreichbar ist. Das lokale Verlängern des Weges zu einer Schleife, wie es hier geschieht, welches global gesehen zum 'Abschneiden' des Weges führt, ist einer der beiden Mechanismen, mit dem für Spieler 0 bessere Schleifen gefunden werden.

Auf die neue Strategie von Spieler 0 reagiert Spieler 1, indem er von v_4 statt wie vorher nach v_6 — was zu einer positiven Schleife führen würde — zu v_5 zieht.

In c) verbessert Spieler 0 seine Strategie in v_5, indem er nach v_1 zieht. Diese Wahl ist entsprechend der Bewertung aus b) besser, weil der Knoten v_6, der relevanter als v_5 ist, auf dem Weg zu v_6 besucht werden kann. Dies zeigt den anderen Mechanismus, durch den eine neue Schleife (hier mit dem relevantesten Knoten v_6) entstehen kann. Spieler 1 kann das Erreichen dieser positiven Schleife nur für die Knoten v_2, v_3 und v_7 verhindern, indem er von v_2 und v_7 zu v_3 zieht. Offensichtlich ist die neue Bewertung in c) optimal für Spieler 0, so dass sich seine Strategiewahl nicht mehr verändert und der Algorithmus terminiert. Somit gewinnt Spieler 0 mit dieser Strategie von den Knoten v_1, v_4, v_5, v_6, v_8 aus und Spieler 1 mit seiner Strategie von den Knoten v_2, v_3, v_7 aus.

3.4 Korrektheit

In diesem Abschnitt wird gezeigt, dass der angegebene Algorithmus terminiert und für einen gegebenen Spielgraphen die Gewinnmengen und Gewinnstrategien für beide Spieler liefert.

[4] Die Wahl der neuen Strategie für Spieler 0 in Situation b) erfolgt ausschließlich auf der Basis der Bewertung der vorangehenden Situation a).

Lemma 3.7 *Sei $G = (V, E)$ ein Spielgraph mit einer Aufteilung der Knotenmenge $V = V_0 \dot\cup V_1$ auf beide Spieler und einer Färbung $c : V \to [|V| + 1]$, und sei $w \in V$ ein Knoten mit folgenden Eigenschaften:*

1. *Der Knoten w ist von allen anderen Knoten aus erreichbar.*
2. *Es existiert eine Schleife $L \subseteq V$ mit $w = \max_< L$.*
3. *Es existiert keine Schleife $L' \subseteq V$ mit $\max_< L' \prec w$.*

Dann terminiert der Funktionsaufruf subvaluation(G, w) (siehe Abbildung 3.8) und liefert eine Bewertung φ, die optimal für Spieler 1 ist.

Beweis: In der Funktion *subvaluation()* haben alle Schleifen eine feste endliche Anzahl von Iterationen. Daher reicht es zu zeigen, dass alle Funktionsaufrufe terminieren. Für die Aufrufe von *reach()* und *minimal_distances()* ist die Termination offensichtlich. Für den Funktionaufruf *maximal_distances(K, w)* muss man zeigen, dass alle Schleifen in K den Knoten w enthalten, wenn $w \in V_+$. Der Graph K ist ein Teilgraph von G. Wenn $w \in V_+$, dann gilt mit den Voraussetzungen 2 und 3 dieses Lemmas, dass für jede Schleife L' in K auch $w \leq \max_< L' \in V_+$. Aus Zeile 9 in *subvaluation()* folgt, dass keine Schleife durch $\max_< L'$ im Teilgraph K existieren kann. Somit gilt für jede Schleife L' in K: $w \leq \max_< L'$, womit w auch in L' vorkommt. Also terminiert der Funktionsaufruf *subvaluation(G, w)*.

Im Folgenden wird gezeigt, dass die durch φ den Knoten zugeordneten Partieprofile eine Bewertung bilden, d.h. es ist zu zeigen:

$$\forall x \in V_G : \exists y \in V_G : x \lhd_\varphi y.$$

In der Funktion werden einige Kanten aus dem übergebenen Graphen G entfernt, so dass schließlich der Teilgraph K entsteht. Es reicht zu zeigen, dass die Bewertung φ eine Bewertung von K und damit auch G ist, da diese Eigenschaft durch Hinzufügen von Kanten nicht verloren geht.

Mit $\forall v \in V_G : \varphi_0(v) = w$ sind die φ_0-Bedingungen für $\lhd_\varphi$ erfüllt. Die Schleife in Zeile 4 beginnt mit $\forall v \in V_G : \varphi_1(v) = \emptyset$. Für diese Schleife gilt nach jeder Iteration die folgende Invariante für u:

$$\forall x, y \in V : xEy \Longrightarrow (x < u \wedge \varphi_1(x) = \varphi_1(y)) \vee (x \geq u \wedge \varphi_1(x) = \varphi_1(y) \dot\cup \{x\}).$$

Wenn die Schleife terminiert, gilt somit:

$$\forall x, y \in V : xEy \Longrightarrow (x < w \wedge \varphi_1(x) = \varphi_1(y)) \vee (x \geq w \wedge \varphi_1(x) = \varphi_1(y) \dot\cup \{x\}).$$

Weil die Werte von $\varphi_1(w)$ nicht verändert werden, erhalten wir $\varphi_1(w) = \emptyset$. Somit ist auch die φ_1-Bedingung für $\lhd_\varphi$ erfüllt.

Schließlich wird für jedes $v \in V$ der Wert von $\varphi_2(v)$ auf die minimale bzw. maximale Distanz zu w gesetzt. Es folgt:

$$\forall x \in V : (x = w \wedge \varphi_2(w) = 0) \ \vee \ \exists y \in V : xEy \wedge \varphi_2(x) = \varphi(y) + 1,$$

was die φ_2-Bedingung für $\lhd_\varphi$ erfüllt. Somit ist φ eine Bewertung von K.

Es bleibt zu zeigen, dass φ eine optimale Bewertung für Spieler 1 auf G ist. Zuerst zeigen wir, dass es zu jeder Kante, die durch die Funktion entfernt wird, eine vom selben Knoten ausgehende Kante existiert, die zu einem Knoten mit einem $\prec$-kleineren Wert führt. Somit reicht es zu zeigen, dass φ eine optimale Bewertung auf dem schließlich entstehenden Teilgraph K für Spieler 1 ist.

In Zeile 6 werden die Knoten in $\bar{U}$ gesammelt, für die ein Pfad zu w existiert, der u vermeidet. Somit hat jeder Knoten in $\bar{U} \setminus \{w\}$ einen E-Nachfolger in $\bar{U}$. In Zeile 9 werden alle Kanten von $\bar{U} \cup \{u\}$ zu $V \setminus \bar{U}$ entfernt. Für eine solche Kante von x nach y sei z ein E-Nachfolger von x mit $z \in \bar{U}$. Aus obiger Invariante folgt:

$$\varphi_1(y) \setminus \{u\} = \varphi_1(x) \setminus \{u, x\} = \varphi_1(z) \setminus \{u\}$$

Weil $u \in \varphi_1(y)$, $u \notin \varphi_1(z)$ und $u \in V_+$ folgt $\varphi_1(z) \prec_u \varphi_1(y)$. Später werden in dieser Funktion den φ_1-Werten nur noch $<$-kleinere Knoten hinzugefügt. Somit gilt $\varphi_1(z) \prec_u \varphi_1(y)$ und somit $\varphi(z) \prec \varphi(y)$. Somit wird die Optimalität der Bewertung für Spieler 1 durch das Entfernen dieser Kanten nicht beeinflusst.

In Zeile 11 werden die Knoten in U gesammelt, zu denen ein Pfad zu u existiert. Das bedeutet, dass jeder Knoten in $U \setminus \{u\}$ einen E-Nachfolger in U besitzt. In Zeile 14 werden alle Kanten von $U \setminus \{u\}$ nach $V_K \setminus U$ gelöscht. Für eine solche Kante von x nach y sei z ein E-Nachfolger von x mit $z \in U$. Aus der Invariante folgt:

$$\varphi_1(y) \setminus \{u\} = \varphi_1(x) \setminus \{u, x\} = \varphi_1(z) \setminus \{u\}$$

Weil $u \notin \varphi_1(y)$, $u \in \varphi_1(z)$ und $u \in V_-$ folgt $\varphi_1(z) \prec_u \varphi_1(y)$. Später werden in dieser Funktion nur $<$-kleinere Knoten zu den φ_1-werten hinzugefügt. Somit erhält man schließlich $\varphi_1(z) \prec_u \varphi_1(y)$ und damit $\varphi(z) \prec \varphi(y)$. Somit wird die Optimalität der Bewertung für Spieler 1 durch das Entfernen dieser Kanten nicht beeinflusst.

Bei Ausführung von Zeile 19 gilt mit der Invariante, dass für jeden Knoten alle E-Nachfolger den gleichen φ_1-Wert haben. Das heißt, dass die φ-Werte sich lediglich in φ_2 unterscheiden. Somit ist die Optimalitätsbedingung für Spieler 1 für w bereits erfüllt. Wenn $w \in V_+$ ist der φ_2-Wert die maximale Distanz zu w. Sei $x \in V \setminus \{w\}$. Dann gilt für jeden E-Nachfolger $y \in V$ von x mit minimalem φ_2-Wert: $\varphi_2(x) = \varphi_2(y) + 1$, was bedeutet $x \lhd_\varphi y$, womit die Optimalitätsbedingung für Spieler 1 erfüllt ist. Der Fall $w \in V_-$ ist analog. □

Lemma 3.8 *Sei $G = (V, E)$ ein Spielgraph mit einer Aufteilung der Knotenmenge $V = V_0 \dot\cup V_1$ auf beide Spieler und einer Färbung $c : V \to [|V| + 1]$. Sei $\sigma : V_0 \to V_1$ eine Strategie für Spieler 0. Sei G_σ der Spielgraph G eingeschränkt auf die Strategie σ, d.h. $G_\sigma = (V_0, V_1, E', c)$, wobei für alle $u, v \in V$:*

$$uE'v \iff (\sigma(u) = v \wedge v \in V_0) \vee (uEv \wedge v \in V_1)$$

Dann terminiert der Funktionsaufruf valuation(G_σ) (siehe Abbildung 3.7), liefert eine Bewertung zurück, die Strategie σ ist verträglich mit dieser Bewertung und diese Bewertung ist optimal für Spieler 1.

Beweis: Um zu beweisen, dass die Funktion *valuation(G_σ)* eine Bewertung liefert, so dass σ verträglich mit ihr ist, reicht es zu zeigen, dass die Funktion eine Bewertung liefert, weil es zu jeder Bewertung mindestens eine Strategie für jeden Spieler gibt, so dass diese Strategie mit der Bewertung verträglich ist. In G_σ ist σ die einzige Strategie für Spieler 0 und ist daher verträglich mit jeder Bewertung von G_σ.

In der Funktion *valuation()* werden zuerst die φ-Werte aller Knoten auf den Wert $\perp$ gesetzt. In der Hauptschleife (Zeile 3) wird unter den Knoten mit undefiniertem Profil nach einem $\prec$-kleinsten Knoten w gesucht, durch den eine Schleife (Zeile 5) verläuft, deren relevantester Knoten w ist (Zeile 6). Für einen solchen Knoten w wird die Menge R aller Knoten bestimmt, von denen aus w erreichbar ist (Zeile 7). Durch den Aufruf von *subvaluation()* wird für den durch R induzierten Teilgraph eine Bewertung berechnet und an φ zugewiesen. Schließlich werden Kanten entfernt, die R verlassen. Durch diese Kanten kann die Optimalität der berechneten Bewertung nicht verletzt werden, da für alle $v \in R$: $\varphi_0(v) = w$ gilt und Kanten, die R verlassen, zu Knoten v' mit $\varphi(v') = \perp$führen, was bedeutet, dass $\varphi_0(v')$ später auf ein Wert gesetzt wird, der $\prec$-größer als w ist. Das bedeutet, dass alle Kanten zwischen den berechneten Teilgraphen die Optimalität für Spieler 1 nicht verletzen, und aus Lemma 3.8 folgt, dass die Bewertung für die Teilgraphen optimal für Spieler 1 ist, womit die Bewertung für den ganzen Graphen optimal für Spieler 1 ist. □

Lemma 3.9 *Sei $G = (V, E)$ ein Spielgraph mit einer Aufteilung der Knotenmenge $V = V_0 \dot\cup V_1$ auf beide Spieler und einer Färbung $c : V \to [|V| + 1]$. Dann terminiert der Funktionsaufruf main(G, V_0, V_1, c) (siehe Abbildung 3.6).*

Beweis: Der einzige nicht-triviale Funktionsaufruf in *main()* ist *valuation(G_σ)* und nach Lemma 3.8 terminiert dieser. Somit bleibt zu zeigen, dass die Anzahl der Iterationen der **repeat-until**-Schleife beschränkt ist, da dieses die einzige Schleife in *main()* ist, deren Durchlaufanzahl nicht vor Ausführung der Schleife feststeht.

Aus Lemma 3.8 ist bekannt, dass die in jedem Schritt erzeugten Bewertungen optimal für Spieler 1 sind und dass die vor ihrer Berechnung festgelegte Strategie σ mit ihr verträglich ist. Sei φ eine in einer Iteration berechnete Bewertung und φ' die für Spieler 0 in Bezug auf φ verbesserte Bewertung, die in der folgenden Iteration berechnet wird. Mit Lemma 3.6 erhält man:

$$\forall v \in V : \varphi(v) \preceq \varphi'(v). \tag{3.13}$$

Es gibt nur endlich viele Strategien für Spieler 0. Daher wird nach endlich vielen Iterationen eine Strategie σ für Spieler 0 noch einmal gewählt. Für gleiche Stra-

tegien von Spieler 0 werden stets die gleichen Bewertungen berechnet. Aus (3.13) ergibt sich sofort, dass alle Bewertungen von dem ersten Auftreten der Strategie σ an gleich sind. Ist eine Strategie für Spieler 0 bereits optimal, so wird sie nicht mehr verändert. Somit verändert sich auch die gewählte Strategie nach dem ersten Auftreten von σ nicht mehr, was unmittelbar zum terminieren der Schleife und damit auch zum terminieren von *main()* führt. □

Satz 3.10 *Sei $G = (V, E)$ ein Spielgraph mit einer Aufteilung der Knotenmenge $V = V_0 \dot\cup V_1$ auf beide Spieler und einer Färbung $c : V \to [|V| + 1]$, und seien W_0, W_1 und die Strategien σ für Spieler 0 und τ für Spieler 1 die Rückgabewerte des Funktionsaufrufs main(G, V_0, V_1, c) (siehe Abbildung 3.6).*

Dann bilden W_0 und W_1 eine Partition von V und σ ist eine Gewinnstrategie auf W_0 für Spieler 0 und τ eine Gewinnstrategie auf W_1 für Spieler 1.

Beweis: Nach Definition von W_0 und W_1 in den Zeilen 13, 14 von Abbildung 3.6 bilden die Mengen W_0, W_1 eine Partition von V.

Aus Lemma 3.8 folgt, dass die zuletzt berechnete Bewertung optimal für Spieler 1 ist. Weil Spieler 0 seine Strategie im Bezug auf die vorangehende Bewertung nicht verbessern kann (siehe Zeilen 8–10 in Abbildung 3.6), folgt, dass die Bewertung optimal für Spieler 0 ist. Somit ist die zuletzt berechnete Bewertung eine optimale Bewertung. Aus Lemma 3.8 folgt, dass σ verträglich mit dieser Bewertung ist. In Zeile 12 von Abbildung 3.6 wird die Strategie τ so gewählt, dass sie auch mit dieser Bewertung verträglich ist. Nach Satz 3.5 ist die Strategie σ eine Gewinnstrategie für Spieler 0 (auf W_0) und τ eine Gewinnstrategie für Spieler 1 (auf W_1). □

3.5 Komplexität

In diesem Abschnitt gehen wir zunächst auf den Speicherplatzbedarf des diskreten SVA ein und schätzen dann seine Laufzeit ab.

3.5.1 Platzkomplexität

Legt man das Einheitskostenmaß zu Grunde, so erhält man die folgende Abschätzung des Platzbedarfs:

Satz 3.11 *Sei $G = (V, E)$ ein Spielgraph mit einer Aufteilung der Knotenmenge $V = V_0 \dot\cup V_1$ auf beide Spieler und einer Färbung $c : V \to [|V| + 1]$. Dann lässt sich der Aufruf main(G, V_0, V_1, c) in Platz $O(|V|^2 + |E|)$ berechnen.*

Beweis: Sei $G = (V, E)$ ein Spielgraph mit einer Aufteilung der Knotenmenge $V = V_0 \dot\cup V_1$ auf beide Spieler und einer Färbung $c : V \to [|V| + 1]$.

In *main*() benötigen der übergebene Graph Platz $O(|V|+|E|)$ und die Parameter V_0, V_1, c jeweils Platz $O(|V|)$. Alle Variablen für Knotenmengen V_+, V_-, W_0, W_1

und auch Strategien σ, σ', τ brauchen Platz $O(|V|)$. Die Variable E' für Kanten benötigt Platz $O(|E|)$, da $E' \subseteq E$ gilt. Die Variable φ für die Bewertung enthält für jeden Knoten ein Tripel aus $V \times 2^V \times [|V|]$. Somit ist der Platzaufwand pro Knoten $O(1+|V|+1)$ und damit für φ insgesamt $O(|V|^2)$. Die übrigen Variablen für Knoten haben lediglich konstanten Platzbedarf.

In *valuation*() benötigt der übergebene Graph Platz $O(|V| + |E|)$. Die Variablen für Knotenmengen L und R benötigen Platz $O(|V|)$. Die Variable φ für die Bewertung benötigt, wie schon in *main*(), Platz $O(|V|^2)$.

In *subvaluation*() benötigt der übergebene Graph Platz $O(|V| + |E|)$. Die Variablen für Knotenmengen U und $\bar{U}$ benötigen Platz $O(|V|)$. Die Variable φ für die Bewertung benötigt, wie schon in *main*(), Platz $O(|V|^2)$.

Da diese Funktionen nicht rekursiv aufgerufen werden, wird für alle Parameter und lokalen Variablen einmalig Platz benötigt. Somit ist ihr Platzaufwand $O(|V|^2 + |E|)$.

Die im Algorithmus verwendeten Hilfsfunktionen *reach*(), *maximal_distances*() und *minimal_distances*() suchen den übergebenen Graphen jeweils nach unterschiedlichen Durchlaufregeln (Modifikationen von Tiefen- und Breitensuche) ab. Dazu reicht es, eine konstante Menge von Informationen pro Knoten zu speichern. Weiterhin gibt *reach*() eine Teilmenge der Knoten des übergebenen Graphen zurück und *maximal_distances*() und *minimal_distances*() einen Distanzwert zu jedem Knoten des gegebenen Graphen. Somit ist der Platzbedarf der Parameter jeweils $O(|V| + |E|)$ und für lokale Variablen und Rückgabewert $O(V)$.

Damit ergibt sich zur Ausführung des Algorithmus ein Gesamtplatzbedarf von $O(|V|^2 + |E|)$. □

Bei der Abschätzung der Platzkomplexität fällt auf, dass lediglich für die zweite Komponente der in der Bewertungsvariablen φ gespeicherten Knotenmenge Platz $O(|V|^2)$ erforderlich ist. In Unterabschnitt 3.6.1 zeigen wir, dass sich dieser Platzaufwand bei einer Variante des Algorithmus ohne Nachteile[5] auf $O(|V|)$ reduzieren lässt.

Weiterhin kann man das Kopieren des Graphen und damit der Kanten vermeiden. Damit ist der Platzaufwand, abgesehen von dem übergebenen Graphen, $O(|V|^2)$, wenn die Kanten in geeigneten Datenstrukturen vorliegen und ihre Reihenfolge durch den Algorithmus verändert werden darf.

Schätzt man den Platzaufwand nach logarithmischem Kostenmaß ab, so ergibt sich $O(\log |V| \cdot (|V|^2 + |E|)$, da die Größe der Wertebereiche aller Variablen und Parameter, für die ein Platz nach Einheitskostenmaß benötigt wird, nicht größer als die Anzahl der Knoten bzw. Kanten sind (Es gilt: $O(\log |E|) \subseteq O(\log |V|)$, da $E \subseteq V^2$).

[5] Ausgenommen die Übersichtlichkeit der Darstellung.

3.5.2 Zeitkomplexität

Die Laufzeitabschätzung des diskreten SVA lässt sich auf die Abschätzung der Laufzeit einer Iteration und die Anzahl der möglichen Iterationen aufteilen.

Lemma 3.12 *Sei $G = (V, E)$ ein Spielgraph. Dann wird für eine Iteration des diskreten SVA (main(), Abbildung 3.6) auf G Zeit $O(|V| \cdot |E|)$ benötigt.*

Beweisidee: Sei $G = (V, E)$ ein Spielgraph mit einer Aufteilung der Knotenmenge $V = V_0 \dot\cup V_1$ auf beide Spieler und einer Paritätsgewinnbedingung gegeben. In einer Iteration wird eine Bewertung bestimmt und eine neue Strategie für Spieler 0 gewählt. Um die neue Strategie zu wählen muss jede Kante, die von einem Knoten von Spieler 0 ausgeht, betrachtet werden und jeweils ein Vergleich von Partieprofilen durchgeführt werden, da jeweils der Nachfolger mit dem für Spieler 0 besten Profil bestimmt werden soll. Der Aufwand für einen Vergleich von Partieprofilen ist $O(|V|)$, da die zweiten Komponenten der verglichenen Profile Knotenmengen (repräsentiert durch absteigend sortierte Listen von Knoten) sind. Somit ist der Aufwand für die Strategiewahl $O(|E| \cdot |V|)$. Beim Berechnen der Bewertung wird für jeden Knoten, der relevanter als der relevanteste Schleifenknoten der Schleife ist, zu der man von diesem Knoten aus gelangt, eine Tiefensuche durchgeführt. Eine Tiefensuche kann in Zeit $O(|E|)$ ausgeführt werden, da jeder Knoten mindestens eine ausgehende Kante besitzt. Somit ergibt sich hier ebenfalls ein Zeitaufwand von $O(|E| \cdot |V|)$.

Der folgende Hilfssatz dient zur Abschätzung der Anzahl von Iterationen, die in *main()* (siehe Abbildung 3.6) ausgeführt werden.

Hilfssatz 3.13 *Sei $G = (V, E)$ ein Spielgraph mit einer Aufteilung der Knotenmenge $V = V_0 \dot\cup V_1$ auf beide Spieler und einer Paritätsgewinnbedingung. Dann gilt, wenn bei der Ausführung des diskreten SVA in zwei aufeinanderfolgenden Iterationen die gleichen Bewertungen auftreten, so sind auch die jeweils daraus für Spieler 0 berechneten Strategien gleich.*

Beweisidee: Sei $G = (V, E)$ ein Spielgraph mit einer Aufteilung der Knotenmenge $V = V_0 \dot\cup V_1$ auf beide Spieler und einer Paritätsgewinnbedingung. Angenommen in der i-ten und $i+1$-ten Iteration des Algorithmus tritt dieselbe Bewertung auf. Die Wahl der Strategie für Spieler 0 in einer Iteration erfolgt durch Wahl der bzgl. der Bewertung besten Nachfolger. Ein spezieller Fall ist, dass die Bewertung mehrerer Nachfolger optimal ist, und unter diesen Nachfolgern derjenige ist, der in der Strategie der vorangehenden Iteration gewählt wurde. In diesem Fall wird derselbe Nachfolger auch für die neue Strategie gewählt (siehe Abbildung 3.6 Zeile 8). Da die Bewertungen der i-ten und $i+1$-ten Iteration nach Voraussetzung gleich sind, tritt dieser spezielle Fall hier für alle Knoten von Spieler 0 ein. Somit ist die in der $i+1$-ten Iteration gewählte Strategie dieselbe wie in der i-ten Iteration.

Lemma 3.14 *Sei $G = (V, E)$ ein Spielgraph mit einer Aufteilung der Knotenmenge $V = V_0 \dot\cup V_1$ auf beide Spieler und einer Paritätsgewinnbedingung. Dann ist die Anzahl der Iterationen des diskreten SVA (Aufruf von main() durch $O(|V|^{|V_0|})$ beschränkt.*

Beweisidee: Sei ein Spielgraph (V, E) mit einer Paritätsgewinnbedingung gegeben. Seien V_0 die Knoten von Spieler 0.

Die Anzahl möglicher Strategien von Spieler 0 ist beschränkt durch $O(|V|^{|V_0|})$. Es reicht also im Folgenden zu zeigen, dass der Algorithmus terminiert, sobald eine bereits vorher von Spieler 0 gewählte Strategie nocheinmal gewählt wird.

Angenommen in der k-ten Iteration tritt erstmalig eine Strategie für Spieler 0 auf, die bereits vorher vorgekommen ist. Sei i mit $i < k$ die Nummer der Iteration in der diese Strategie bereits vorher vorkam. Dann sind die Bewertungen, die sich aus beiden Strategien (in den jeweils nächsten Iterationen) ergeben gleich, da die Bewertungen ausschließlich von der Wahl der Strategie von Spieler 0 abhängen. Nach Lemma 3.6 ist in jeder Iteration die berechnete Bewertung größer oder gleich der zuvor berechneten Bewertung. Daraus folgt, dass die Bewertungen in der $i + 1$-ten Iterationen bis zur k-ten Iteration gleich sind. Nach Hilfssatz 3.13 sind somit auch die in der $i + 1$-ten Iterationen bis zur k-ten Iteration gewählten Strategien gleich. Somit ist $k = i+1$, da die k-te Iteration nach Definition das erste wiederholte Vorkommen einer Strategie aufweist. Der Algorithmus terminiert, wenn in zwei aufeinanderfolgenden Iterationen dieselbe Strategie für Spieler 0 gewählt wird. Somit terminiert er nach der k-ten Iteration.

Satz 3.15 *Sei $G = (V, E)$ ein Spielgraph mit einer Aufteilung der Knotenmenge $V = V_0 \dot\cup V_1$ auf beide Spieler und einer Paritätsgewinnbedingung. Dann wird ein Aufruf des diskreten SVA für G in Zeit $O(|E| \cdot |V|^{|V_0|+1})$ ausgeführt.*

Der Beweis dieses Satzes ergibt sich unmittelbar aus den Lemmata 3.12 und 3.14.

Abschätzung der Anzahl benötigter Iterationen

Die zentrale Frage über diesen Algorithmus ist:

> „Ist die Laufzeit des diskreten SVA polynomiell?“

was, da der Zeitaufwand für jeden Iterationsschritt polynomiell ist, identisch mit der Frage ist, ob die Anzahl der Iterationsschritte polynomiell ist.

Eine exponentielle obere Schranke liefert die Abschätzung der Anzahl der Iterationen in Lemma 3.14. Eine lineare untere Schranke für die Anzahl der Iterationen ist durch die skalierte Version des Spielgraphen aus Abbildung 3.10 gegeben. Die genauere Formulierung dieser unteren Schranke ist: Für einen Spielgraph mit n Knoten können $\frac{n}{2}$ Iteration erforderlich sein.

Es ist unklar, ob es Spielgraphen gibt, für die eine quadratische Anzahl von Iterationen notwendig ist.

Ein Ansatz zur Analyse der Anzahl der Iterationen besteht darin, die Iterationen nach den Veränderungen der Bewertung zu klassifizieren. Die ‚großen Schritte' unter den Iterationen sind die, in welchen beim Übergang von der alten zur neuen Bewertung sich die erste Komponente des Partieprofils eines Knotens ändert; die anderen Iterationen bezeichnen wir als ‚kleine Schritte'. Da die erste Komponente eines Profils einen Knoten beinhaltet, kann sie maximal $|V|-1$ mal (bzgl. der Rewardordnung) größer werden. Somit sind nicht mehr als $|V|^2$ solcher Iterationen möglich. Somit bedeutet eine superpolynomielle Anzahl von Iterationen bereits, dass eine superpolynomiell lange Folge kleiner Schritte auftreten muss.

Beispiel für linear viele Iterationen

Ein einfaches Beispiel für einen Spielgraphen ist in Abbildung 3.10 dargestellt. In

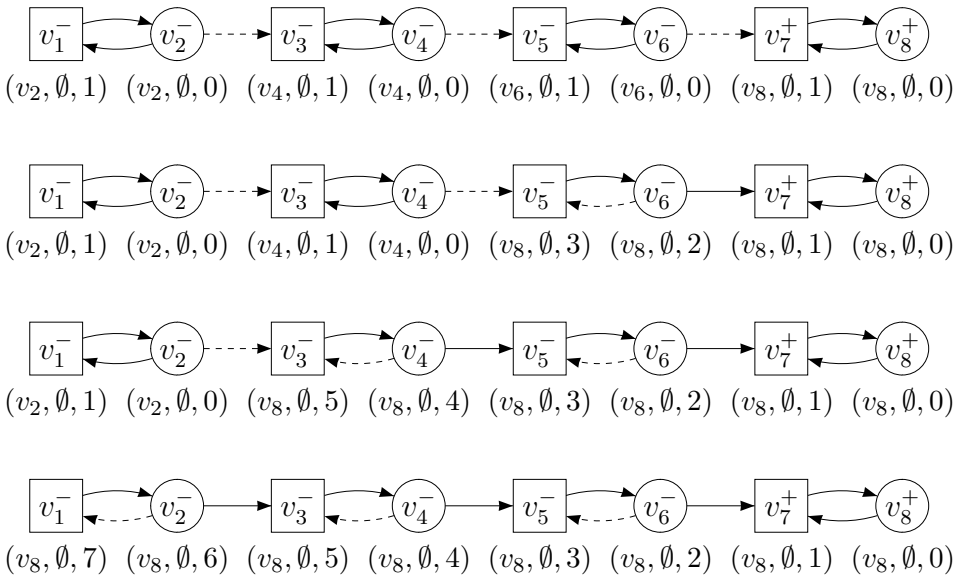


Abbildung 3.10: Ein Spielgraph, für den in vier Iterationsschritten Gewinnstrategien berechnet werden. Es ist jeweils die Situation nach der Berechnung der Bewertung noch vor der Wahl der neuen Strategie für Spieler 0 dargestellt. Aus der Terminationsbedingung ergibt sich, dass die für Spieler 0 nach der letzten Bewertung gewählte Strategie gleich der vor der letzten Bewertung gewählten ist. Sie ist daher nicht noch zusätzlich dargestellt.

diesem Beispiel benötigt der Algorithmus vier Iterationen, um die Gewinnstrategien zu bestimmen. Skaliert man dieses Beispiel, so benötigt der Algorithmus für $2n$ Knoten jeweils genau n Iterationen.

Dieses Beispiel weist eine Besonderheit auf: Für Spieler 1 gibt es nur eine Strategie, die damit auch die Gewinnstrategie ist. Der folgende Satz zeigt, dass

alle Spiele mit dieser Eigenschaft in linear vielen Iterationen berechnet werden können.

Satz 3.16 *Sei $G = (V, E)$ ein Spielgraph mit einer Aufteilung der Knotenmenge $V = V_0 \dot\cup V_1$ auf beide Spieler und einer Paritätsgewinnbedingung gegeben. Dann benötigt der diskrete SVA für G nicht mehr als $|V|$ Iterationen, wenn Spieler 1 nur eine Strategie besitzt (d.h. seine Knoten alle vom Grad 1 sind).*

Beweisidee: Sei $G = (V, E)$ ein Spielgraph mit einer Aufteilung der Knotenmenge $V = V_0 \dot\cup V_1$ auf beide Spieler und einer Paritätsgewinnbedingung gegeben. Sei eine initiale Strategie und ein von ihr ausgehender Lauf des Algorithmus gegeben. Zu zeigen ist, dass der Lauf nicht mehr als $|V|$ Iterationen hat. Es reicht somit zu zeigen, dass in jedem Iterationsschritt mindestens für einen weiteren Knoten eine endgültige Kantenwahl erfolgt.

Dies lässt sich durch Induktion über die Menge der Knoten beweisen, die durch die gewählte Strategie bereits einen optimalen Pfad zu ihrem optimalen relevantesten Schleifenknoten haben. Es liegt dieselbe Idee zu Grunde, die dual im Beweis der Korrektheit von *subvaluation*() verwendet wird.

Abschätzung der Anzahl benötigter Kantentraversierungen

Zur genaueren Angabe des Zeitaufwandes für die Ausführung des Algorithmus sind die zuvor beschriebenen Abschätzungen ungeeignet, da der Aufwand für die Berechnung einzelner Iterationen auch für einen festen Spielgraphen sehr unterschiedlich ausfallen kann. Eine gute Annäherung an den tatsächlichen Zeitaufwand liefert die Anzahl der Kantentraversierungen, da für alle aufwendigen Operationen des Algorithmus die Anzahl der Ausführungen jeder elementaren Operation linear durch die Anzahl der Kantentraversierungen beschränkt ist. Daher ist die Anzahl der Kantentraversierungen ein geeignetes Maß, um den Aufwand für verschiedene Läufe des Algorithmus systemunabhängig zu vergleichen.

3.6 Alternativen der Strategiekonstruktion

In diesem Abschnitt gehen wir auf mögliche Verbesserungen und Varianten des diskreten SVA ein. Anschließend vergleichen wir ihn mit den bereits in Kapitel 2 vorgestellten Algorithmen.

3.6.1 Varianten des Algorithmus

Zu dem vorgestellten Algorithmus existieren eine Reihe möglicher Varianten, die zum einen eine effizientere Ausführung oder eine größere Übersichtlichkeit der Iterationschritte erreichen. Wir stellen vier dieser teilweise kombinierbaren Varianten vor.

Die erste Variante ‚Einfache Verbesserungsschritte' führt zu einem übersichtlicheren Ablauf des Algorithmus, was ist insbesondere für seine Analyse von Bedeutung ist. Die zweite Variante ‚Keine unnötigen Verbesserungen' führt zu einer geringeren Laufzeit. Die dritte Variante ‚Vereinfachte Bewertungen' zeigt, dass man die Mengenkomponente der Bewertungen verkleinern kann (einige Elemente können weggelassen werden). Die letzte Variante ‚Linearer Speicherplatzbedarf' zeigt, dass basierend auf der dritten Variante die Platzkomplexität des Algorithmus reduziert werden kann.

Die erst beiden Varianten beruhen auf der Ausnutzung der folgenden Nicht-Determinismen im vorgestellten Algorithmus:

Nicht-Determinismen Der diskrete SVA weist an zwei Stellen Nicht-Determinismen auf: Die Wahl der initialen Strategie für Spieler 0 erfolgt nicht-deterministisch, und bei der Wahl einer Strategie für Spieler 0 in jeder Iteration können mehrere Nachfolger eines Knotens ein maximales Profil aufweisen, ohne dass einer von ihnen durch die Strategie der vorangehenden Iteration gewählt wurde. In diesem Fall wird ebenfalls nicht-deterministisch einer dieser Nachfolger für die neue Strategie gewählt.

Der zweite aufgeführte Nicht-Determinismus entfällt, wenn alle Knoten des Spielgraphen höchstens vom Grad zwei sind, da neben dem für die Strategie gewählten Nachfolger nicht mehr als ein anderer Nachfolger mit besserem Profil existieren kann. Jeder Spielgraph lässt sich durch Einfügen zusätzlicher Knoten so transformieren, dass alle Knoten höchstens zwei ausgehende Kanten haben. (Hierzu werden höchstens so viele neue Knoten und Kanten benötigt, wie es im ursprünglichen Graphen Kanten gibt.)

Die Anzahl der Iterationen, die zur Berechnung eines gegebenen Paritätsspiels nötig sind, hängt somit im wesentlichen von der gewählten initialen Strategie ab. (Im günstigsten Falle ist dies bereits die Gewinnstrategie und der Algorithmus terminiert nach einer Iteration.)

Einfache Verbesserungsschritte

In der beschriebenen Version des Algorithmus wird in jeder Iteration entsprechend der Bewertung für jeden Knoten von Spieler 0 eine lokal optimale Strategie der bestbewerteten Nachfolger gewählt. Für die Korrektheit des Algorithmus ist dies jedoch nicht erforderlich. Es reicht, wenn in jeder Iteration, nach welcher der Algorithmus nicht terminiert, eine lokale Verbesserung der Strategie für *einen* Knoten von Spieler 0 durchgeführt wird, und bei diesem Knoten muss nicht einer der am besten bewerteten Nachfolger, sondern nur einer, der besser als der zuvor verwendete bewertet ist, gewählt werden.

Bei diesem Vorgehen ist interessant, dass stets eine Verbesserungsfolge existiert, die nur n Iterationsschritte benötigt. Hierbei tritt jedoch das Problem auf

diese Verbesserungsfolge zu finden, da dadurch, dass der zu verbessernde Knoten mit dem besseren Nachfolger gewählt werden muss, ein zusätzlicher Nicht-Determinismus entsteht.

Keine unnötigen Verbesserungen

Es ist nicht erforderlich die Strategie von Spieler 0 für solche Knoten zu verbessern, von denen er mit der bereits gefundenen Strategie gewinnt. D.h. alle positiven Bewertungen können zu einem einzigen Wert zusammengefasst werden, so dass im Bereich der positiven Bewertungen keine Verbesserungen mehr stattfinden können.

Vereinfachte Bewertungen

Die einzige Weise, in welcher die berechneten Bewertungen verwendet werden, ist der Vergleich im Bezug auf ihre Größe. Veränderungen an den Bewertungen, die invariant im Bezug auf den Größenvergleich der Bewertungen einer beliebigen Iteration sind, haben somit keinen Einfluss auf das Verhalten des Algorithmus.

Diese Eigenschaft lässt sich insbesondere für die zweite Komponente („Menge der größeren Knoten auf dem Weg zur Schleife“) der Bewertungen ausnutzen. Beim Vergleich der Bewertungen zweier Knoten ist die zweite Komponente nur dann von Bedeutung, wenn die ersten Komponenten gleich sind; d.h. die Partien zur selben Schleife führen. Weiterhin ist für den Vergleich der zweiten Komponenten nur ihre symmetrische Mengendifferenz von Bedeutung und in dieser nur der relevanteste Knoten. Ein Knoten q in der zweiten Komponente der Bewertung eines Knoten p kann somit weggelassen werden, wenn zwischen p und q ein Knoten q' mit größerer Relevanz als q liegt.

Linearer Speicherplatzbedarf

In der zuerst beschriebenen Version des Algorithmus können die zweiten Komponenten der Bewertungen einzeln als Listen gespeichert werden. Für den Speicherplatzbedarf des Algorithmus ist dabei die Anordnung der Knoten in diesen Listen von entscheidender Bedeutung.

Ordnet man die Knoten nach Relevanz absteigend an, so ermöglicht dies, sie in linearer Zeit zu vergleichen. Der Aufbau in dieser Anordnung ist trivial, weil die Einfügeoperationen des Algorithmus die Knoten mit absteigender Relevanz einfügen; d.h. ‚Einfügen‘ bedeutet hier ‚Anhängen‘. Jedoch müssen die zweiten Komponenten für jeden Knoten einzeln abgespeichert werden, was zu einem Speicherplatzbedarf von $O(|V|^2)$ führt.

Die zweite mögliche Anordnung richtet sich nach der Reihenfolge, in der die Knoten in einer der Bewertung entsprechenden Partie vorkommen. Durch diese Anordnung sind die zweiten Bewertungskomponenten der einem Knoten v folgenden Knoten alle Teillisten der zweiten Bewertungskomponente des Knoten v. Da

dies für jeden Knoten gilt, lassen sich die Teillisten mehrfach benutzen; d.h. jeder Knoten enthält in der zweiten Bewertungskomponente nur einen Knoten und einen Verweis auf die nächst kleinere Teilliste; man erhält eine Baumstruktur (genauer: einen Baum pro Schleife, so dass die Struktur der zweiten Komponenten insgesamt ein Wald ist). Der Aufwand zum Speichern dieser Listen ist somit linear. Nachteil dieser Anordnung ist jedoch, dass der Vergleich aufwendiger wird. Das Bestimmen des größten Knotens, der in genau einer der beiden Listen enthalten ist, erfordert mehrfaches Suchen oder die Sortierung mindestens einer der beiden Listen, womit ein Zeitaufwand von $O(|V| \cdot \log |V|)$. Dies wirkt sich auf die Zeitkomplexität einer Iteration aus, sie steigt auf $O(\log(|V|) \cdot |V| \cdot |E|)$.

Somit bringt die eine Anordnung einen Vorteil in der Zeitkomplexität, die andere in der Platzkomplexität. Mit der im vorangehenden Absatz vorgeschlagenen vereinfachten Bewertung lassen sich beide Vorteile vereinigen, da beide Ordnungen jeweils für die in einer Liste auftretenden Knoten identisch sind. Die Listen werden als Wald abgespeichert, in dem jeder Knoten eine Liste repräsentiert. Der Vergleich erfolgt in Linearzeit, indem in den beiden gegebenen Listen vom Ende zum Anfang (vom Blatt zur Wurzel) hin der erste gemeinsame Knoten gesucht wird. Entscheidend für den Vergleich ist der größte zuvor gesehene Knoten.

3.6.2 Vergleich zu anderen Algorithmen

Der vorgestellte dSV-Algorithmus unterscheidet sich in wesentlichen Punkten von den anderen bekannten Algorithmen zur Strategiesynthese für Paritätsspiele. Wir zeigen hier die Unterschiede zu diesen Algorithmen auf. Dem Zusammenhang des Algorithmus mit dem von Puri vorgestellten SVA mit reellen Bewertungen ist der folgende Abschnitt 3.7 gewidmet.

Der Algorithmus arbeitet durchgehend auf dem ganzen Graphen. Im Gegensatz dazu werden beim Algorithmus von McNaughton rekursiv Strategien für Teilspiele berechnet.

McNaughtons Algorithmus

Der in Abschnitt 2.2.3 beschriebene Algorithmus von McNaughton [McN93] bearbeitet rekursiv kleinere Teilspiele. Er ruft sich selbst zweimal zur Konstruktion von Strategien für Teilspiele des Spielgraphen auf. Dabei ist der zweite Aufruf jedoch vom ersten abhängig, so dass es hier keinen trivialen Ansatz für eine Parallelisierung des Algorithmus gibt. Der Algorithmus baut die Gewinnstrategien durch Zusammensetzen von Attraktorstrategien — Strategien, die zur Zielmenge des Attraktors führen — auf. Die Auswahl zwischen mehreren solchen Strategien ist nicht-deterministisch.

Dagegen arbeitet der diskrete SVA durchgehend auf dem ganzen Graphen. Das Bestimmen von Attraktoren findet sich auch hier beim Ermitteln der Bewertung wieder. Jedoch wird unter den möglichen Attraktorstrategien noch anhand der

Bewertungen unterschieden, so dass im Falle von Spielgraphen vom Grad 2 die Strategiewahl sogar eindeutig ist.

Fortschrittsmaßalgorithmus

Der vorgestellte diskrete Algorithmus und der in Abschnitt 2.2.4 dargestellte Fortschrittsmaßalgorithmus aus [Jur00b] führen beide schrittweise Verbesserungen von Bewertungen durch. Obwohl beide Algorithmen asymmetrisch im Bezug auf die Spieler vorgehen, liefert der diskrete Algorithmus Gewinnstrategien für beide Spieler, der Fortschrittsmaßalgorithmus hingegen nur für Spieler 0. Er muss dual angewandt werden, um eine Gewinnstrategie für Spieler 1 zu bestimmen.

3.7 Zusammenhang zu Puris Synthesealgorithmus

In diesem Abschnitt zeigen wir, dass es einen engen Zusammenhang zwischen dem vorgestellten diskreten SVA und dem von Puri in [Pur95] vorgeschlagenen SVA gibt. Der von Puri verwendete Algorithmus verbessert eine anfänglich beliebig gewählte Strategie schrittweise, was auch als policy iteration bezeichnet wird. Dieses bereits in [How60] beschriebene Vorgehen lässt sich auf Stochastische Spiele übertragen (siehe [HK66, Ros83, KV86]) und von diesen auf die einfacheren Payoff-Spiele.

Für die Folge von Strategien, die in den Iterationen eines SVA erzeugt wird, ist die verwendete Bewertungsfunktion maßgebend. In jeder Iteration ergibt sich eine neue Strategie ausschließlich aus der alten und durch den Vergleich der Bewertungen der einzelnen Knoten. Daraus folgt unmittelbar, dass Bewertungsfunktionen, die durch ihre Bewertungen auf allen Graphen jeweils dieselbe Ordnung auf den Knoten des Graphen induzieren, auch gleiche Strategiefolgen erzeugen, wenn sie in einem SVA eingesetzt werden.

Zunächst geben wir eine Bewertungsfunktion an, die für einen hinreichend großen Discountfaktor die von Puri angegebene Bewertungsfunktion so gut approximiert, dass sie obige Eigenschaft erfüllt und damit zum gleichen Ablauf des SVA führt.

Danach zeigen wir, dass bei geeigneter Wahl der Gewichte der Größenvergleich der im diskreten SVA verwendeten Bewertungsfunktion eine Vergröberung des Größenvergleichs auf der approximierten Bewertung ist. Das bedeutet, dass ein Verbesserungsschritt, der gemäß der Bewertungsfunktion des diskreten Algorithmus durchgeführt wird, auch eine Verbesserung bezogen auf die approximierte Bewertung darstellt.

3.7.1 Die approximative Bewertungsfunktion

Im Folgenden wird gezeigt, dass es bei einem hinreichend großen Discountfaktor β für den Ablauf des Algorithmus nicht erforderlich ist, die von Puri definierte Bewertung exakt zu berechnen. Der Ablauf des Algorithmus hängt vom Vergleich verschiedener Bewertungen ab. Wir zeigen, dass die hier definierte approximative Bewertung zu denselben Vergleichsergebnissen führt. Dies ist insbesondere auch für die Strategieberechnung in Mean-Payoff-Spielen (MPGs) von Bedeutung, da die Gewinnstrategien in einem DPG bei hinreichend großem Discountfaktor auch Gewinnstrategien des MPGs über demselben Spielgraphen sind.

Zuerst zeigen wir, dass die im Folgenden definierte Bewertung diejenige von Puri approximiert, und danach zeigen wir, dass für einen hinreichend großen Discountfaktor, die Approximation gut genug ist, um die Ordnung zu erhalten.

Sei ein Spielgraph (V, E) gegeben. Sei $\pi \in V^\omega$ eine Partie, die wie alle Partien dieses Kapitels nach positionalen Strategien gespielt ist. Dann existieren $s, t \in \mathbb{N}$ mit $t \geq 0$, $s \geq 1$ und $v_i \in V$ für $i \in [t+s]$ mit $\pi = v_0 \dots v_{t-1}(v_t \dots v_{t+s-1})^\omega$, wobei alle v_i paarweise verschieden sind. (In dem Spezialfall $t = 0$ besteht die Partie nur aus einer Schleife: $\pi = (v_0 \dots v_{s-1})^\omega$.) Sei eine Rewardfunktion $r : V \to \mathbb{R}$ gegeben.

Beim DPG ist das Ergebnis der Partie π wie bereits in Abschnitt 2.3.1 angegeben:

$$R_\beta(\pi) = \sum_{k \in \mathbb{N}} \beta^k r(\pi_k)$$

Nutzt man aus dass die Partie π nach positionalen Strategien gespielt ist, so ergibt sich durch Berechnen der geometrischen Reihe:

$$R_\beta(\pi) = \sum_{i=0}^{t-1} \beta^i r(v_i) + \sum_{i=t}^{t+s-1} \frac{\beta^i}{1-\beta^s} r(v_i) \tag{3.14}$$

Wir definieren als approximatives Ergebnis der Partie π:

$$\tilde{R}_\beta(\pi) = \sum_{i=0}^{t-1} r(v_i) + \sum_{i=t}^{t+s-1} \frac{1}{s} \left(\frac{1}{1-\beta} + \frac{s-1}{2} - i \right) r(v_i) \tag{3.15}$$

Das folgende Lemma zeigt, dass das approximative Ergebnis einer Partie für Discountfaktoren β nahe bei 1 einen gute Annäherung an des Ergebnis im DPG ist.

Lemma 3.17 *Sei ein Spielgraph (V, E) und $\pi = v_0 \dots v_{t-1}(v_t \dots v_{t+s-1})^\omega$ eine Partie, wobei $t \geq 0$, $s \geq 1$, $v_i \in V$ für alle $i \in [t+s]$ und alle v_i paarweise verschieden sind, und sei $r : V \to \mathbb{R}$ eine Rewardfunktion. Sei $w_{\max} = \max_{i \in [t+s]} |r(v_i)|$. Dann gilt bei einem Discountfaktor β mit $1 - \frac{1}{2nn!} < \beta < 1$:*

$$|R_\beta(\pi) - \tilde{R}_\beta(\pi)| < 9n^3 n!(1-\beta) w_{\max}$$

Um dieses Lemma zu beweisen, benötigen wir zunächst folgende technische Hilfsbehauptung zur Abschätzung des Discountfaktors:

Hilfssatz 3.18 *Sei* $i, s \in \mathbb{N}$ *und* $1 \leq s \leq n$, $0 \leq i \leq n$, $\beta \in \mathbb{R}$ *und* $1 - \frac{1}{2nn!} < \beta < 1$ *dann gilt:*

$$\left| \frac{\beta^i}{1-\beta^s} - \frac{1}{s}\left(\frac{1}{1-\beta} + \frac{s-1}{2} - i\right)\right| < 8n^2 n!(1-\beta)$$

Beweis: Sei $i, s \in \mathbb{N}$ und $1 \leq s \leq n$, $0 \leq i \leq n$, $\beta \in \mathbb{R}$ und $1 - \frac{1}{2nn!} < \beta < 1$. Wir setzen $\epsilon = 1 - \beta$.[6] Wir beginnen mit der rechten Seite der zu zeigenden Ungleichung. Zur Vereinfachung beginnen wir mit dem $(1-\beta^s)$-Vielfachen:

$$\beta^i - \frac{1-\beta^s}{s}\left(\frac{1}{1-\beta} + \frac{s-1}{2} - i\right)$$

$$= (1-\epsilon)^i - \frac{1-(1-\epsilon)^s}{s}\left(\frac{1}{\epsilon} + \frac{s-1}{2} - i\right)$$

$$= \sum_{k=0}^{i}\binom{s}{k}(-\epsilon)^k + \frac{1}{s}\sum_{k=1}^{s}\binom{s}{k}(-\epsilon)^k\left(\frac{1}{\epsilon} + \frac{s-1}{2} - i\right)$$

$$= \sum_{k=0}^{i}\binom{s}{k}(-\epsilon)^k + \frac{1}{s\epsilon}\sum_{k=1}^{s}\binom{s}{k}(-\epsilon)^k + \frac{s-1-2i}{2s}\sum_{k=1}^{s}\binom{s}{k}(-\epsilon)^k$$

und, da $1 + \frac{1}{s\epsilon}s(-\epsilon) = 0$:

$$= \sum_{k=1}^{i}\binom{s}{k}(-\epsilon)^k + \frac{1}{s\epsilon}\sum_{k=2}^{s}\binom{s}{k}(-\epsilon)^k + \frac{s-1-2i}{2s}\sum_{k=1}^{s}\binom{s}{k}(-\epsilon)^k$$

und, da $i(-\epsilon) + \frac{1}{s\epsilon}\frac{s(s-1)}{2}(-\epsilon)^2 + \frac{s-1-2i}{2s}s(-\epsilon) = 0$:

$$= \sum_{k=2}^{i}\binom{s}{k}(-\epsilon)^k + \frac{1}{s\epsilon}\sum_{k=3}^{s}\binom{s}{k}(-\epsilon)^k + \frac{s-1-2i}{2s}\sum_{k=2}^{s}\binom{s}{k}(-\epsilon)^k$$

Aus

$$\left|\beta^i - \frac{1-\beta^s}{s}\left(\frac{1}{1-\beta} + \frac{s-1}{2} - i\right)\right|$$

$$= \left|\sum_{k=2}^{i}\binom{s}{k}(-\epsilon)^k + \frac{1}{s\epsilon}\sum_{k=3}^{s}\binom{s}{k}(-\epsilon)^k + \frac{s-1-2i}{2s}\sum_{k=2}^{s}\binom{s}{k}(-\epsilon)^k\right|$$

[6] Grund für diese Substitution $\epsilon = 1 - \beta$ ist, dass das Abschätzen der Größe von Polynomen über β mit β nahe bei 1 nur möglich ist, wenn man alle Koeffizienten genau abschätzt. Dagegen lässt sich die Größe von Polynomen über ϵ mit ϵ positiv und nahe bei 0 anhand der ersten Koeffizienten abschätzen, wenn die Größe aller Koeffizienten beschränkt werden kann.

erhält man mit $\sum_{k=2}^{s}\binom{s}{k}\epsilon^k \leq \sum_{k=2}^{n}\binom{n}{k}\epsilon^k < n!n\epsilon^2$:

$$< \left(1 + \frac{1}{s\epsilon} + \frac{|s-1-2i|}{2s}\right) n!n\epsilon^2 \quad \leq \quad 4n!n^2\epsilon^2$$

Der zur Vereinfachung eingeführte zusätzlich Faktor $1-\beta^s$ lässt sich wie folgt abschätzen:

$$1-\beta^s > -\sum_{l=1}^{s}\binom{s}{l}(-\epsilon)^l = s\epsilon - \sum_{l=2}^{s}\binom{s}{l}(-\epsilon)^l > \epsilon - nn!\epsilon^2 > \frac{1}{2}\epsilon$$

Also ergibt sich mit $\frac{1}{1-\beta^s} < \frac{2}{\epsilon}$:

$$\left|\frac{\beta^i}{1-\beta^s} - \frac{1}{s}\left(\frac{1}{1-\beta} + \frac{s-1}{2} - i\right)\right| \; < \; \frac{2}{\epsilon}\, 4n^2n!\epsilon^2 \; = \; 8n^2n!(1-\beta)$$

□

Beweis von Lemma 3.17: Sei ein Spielgraph (V, E) und eine Partie $\pi = v_0 \dots v_{t-1}(v_t \dots v_{t+s-1})^\omega$ gegeben, wobei $t \geq 0$, $s \geq 1$, $v_i \in V$ für alle $i \in [t+s]$ und alle v_i paarweise verschieden sind. Sei $r : V \to \mathbb{R}$ eine Rewardfunktion. Sei $\beta \in (0,1)$ ein Discountfaktor mit $1 - \frac{1}{2nn!} < \beta < 1$.

Als Abkürzungen definieren wir $w_i = r(v_i)$ für alle $i \in [t+s]$ und $w_{\max} = \max_{i\in[t+s]} |w_i|$.

Wir erhalten durch Einsetzen der Definition:

$$|R_\beta(\pi) - \tilde{R}_\beta(\pi)|$$

$$= \left|\sum_{i=0}^{t-1}\beta^i w_i + \sum_{i=t}^{t+s-1}\frac{\beta^i}{1-\beta^s}w_i - \sum_{i=0}^{t-1}w_i - \sum_{i=t}^{t+s-1}\frac{1}{s}\left(\frac{1}{1-\beta} + \frac{s-1}{2} - i\right)w_i\right|$$

Mit der Dreiecksungleichung ergibt sich:

$$\leq \left|\sum_{i=0}^{t-1}\beta^i w_i - \sum_{i=0}^{t-1}w_i\right| + \left|\sum_{i=t}^{t+s-1}\left(\frac{\beta^i}{1-\beta^s} - \frac{1}{s}\left(\frac{1}{1-\beta} + \frac{s-1}{2} - i\right)\right)w_i\right|$$

Wir betrachten die beiden Summanden dieser Summe einzeln. Für den ersten Summanden gilt:

$$\left|\sum_{i=0}^{t-1}\beta^i w_i - \sum_{i=0}^{t-1}w_i\right| \; = \; \left|\sum_{i=0}^{t-1}w_i\sum_{k=1}^{i}\binom{i}{k}(\beta-1)^k\right| \leq n^2n!(1-\beta)w_{\max}$$

Für den zweiten Summanden gilt mit Hilfssatz 3.18:

$$\left|\sum_{i=t}^{t+s-1}\left(\frac{\beta^i}{1-\beta^s} - \frac{1}{s}\left(\frac{1}{1-\beta} + \frac{s-1}{2} - i\right)\right)w_i\right|$$

$$< \left| \sum_{i=t}^{t+s-1} 8n^2 n!(1-\beta)w_i \right| \leq 8n^3 n!(1-\beta)w_{\max}$$

Wie wir gezeigt haben gilt:

$$|R_\beta(\pi) - \tilde{R}_\beta(\pi)|$$

$$\leq \left| \sum_{i=0}^{t-1} \beta^i w_i - \sum_{i=0}^{t-1} w_i \right| + \left| \sum_{i=t}^{t+s-1} \left(\frac{\beta^i}{1-\beta^s} - \frac{1}{s}\left(\frac{1}{1-\beta} + \frac{s-1}{2} - i \right) \right) w_i \right|$$

und mit Abschätzung beider Summanden:

$$< n^2 n!(1-\beta)w_{\max} + 8n^3 n!(1-\beta)w_{\max} \leq 9n^3 n!(1-\beta)w_{\max}$$

und somit:

$$|R_\beta(\pi) - \tilde{R}_\beta(\pi)| < 9n^3 n!(1-\beta)w_{\max}$$

□

Damit approximative Ergebnisse einsetzbar sind, muss der Discountfaktor so gewählt werden, dass ihre Abweichung geringer als die Hälfte der Differenz der Ergebnisse zweier beliebiger Partien mit verschieden exakten Bewertungen ist. Der folgende Satz gibt eine untere Schranke für die Differenz der Ergebnisse zweier verschiedener exakter Bewertungen an.[7]

Satz 3.19 *Sei ein Spielgraph (V, E), eine Gewichtsfunktion $r : V \to \mathbb{Z}$ gegeben und $r_{\max} = \max_{v \in V} |r(v)|$. Sei $\beta \in \mathbb{R}$ mit $1 - \frac{1}{2nr_{\max}} < \beta < 1$. Seien $\pi, \pi' \in V^\omega$ Partien auf diesem Spielgraph. Dann gilt:*

$$\tilde{R}_\beta(\pi) = \tilde{R}_\beta(\pi') \text{ oder } |\tilde{R}_\beta(\pi) - \tilde{R}_\beta(\pi')| > \frac{1}{2n!}$$

Die entscheidende Folgerung aus diesem Satz ist, dass man den Größenvergleich der Ergebnisse der exakten Bewertung zweier Partien an Hand des Größenvergleichs der Ergebnisse der approximativen Bewertungen bestimmen kann:

Folgerung 3.20 *Sei ein Spielgraph (V, E), eine Gewichtsfunktion $r : V \to \mathbb{Z}$ und $r_{\max} = \max_{v \in V} |r(v)|$. Sei $\beta \in \mathbb{R}$ mit $1 - \frac{1}{2nn!r_{\max}} < \beta < 1$. Seien $\pi, \pi' \in V^\omega$ Partien auf diesem Spielgraph. Dann gilt:*

$$R_\beta(\pi) < R_\beta(\pi') \iff \tilde{R}_\beta(\pi) < \tilde{R}_\beta(\pi')$$

[7] Dieser Satz ist auf den Fall eingeschränkt, dass die Gewichte ganze Zahlen sind. Da wir im Folgenden nur ganzzahlige Gewichte benötigen, beschränken wir uns auf diesen einfacher zu beweisenden Fall. Eine Abschätzung ist offensichtlich auch für reelle Gewichtsfunktionen möglich, da nur endlich viele verschiedene Partien zu einem gegebenen Spielgraphen existieren.

Beweis von Satz 3.19: Sei ein Spielgraph (V, E), eine Gewichtsfunktion $r : V \to \mathbb{Z}$ gegeben und $r_{\max} = \max_{v \in V} |r(v)|$. Sei $\beta \in \mathbb{R}$ mit $1 - \frac{1}{2nr_{\max}} < \beta < 1$. Seien $\pi, \pi' \in V^\omega$ Partien auf diesem Spielgraph.

Für die Partie π existieren dann $s, t \in \mathbb{N}$ mit $t \geq 0$, $s \geq 1$ und $v_i \in V$ für $i \in [t + s]$ mit $\pi = v_0 \dots v_{t-1}(v_t \dots v_{t+s-1})^\omega$, wobei alle v_i paarweise verschieden sind. Für das approximative Ergebnis der Partie gilt:

$$\tilde{R}_\beta(\pi) = \sum_{i=0}^{t-1} w_i + \sum_{i=t}^{t+s-1} \frac{1}{s} \left(\frac{1}{1-\beta} + \frac{s-1}{2} - i \right) w_i$$

Setzt man $a = \sum_{i=t}^{t+s-1} \frac{n!}{s} \in \mathbb{Z}$ und $b = 2n! \sum_{i=0}^{t-1} r(v_i) + \sum_{i=t}^{t+s-1} \frac{n!}{s}(s - 1 - 2i) \in \mathbb{Z}$, so erhält man:

$$\tilde{R}_\beta(\pi) = \frac{a}{n!(1-\beta)} + \frac{b}{2n!}$$

Für die Partie π' existieren entsprechend $s', t' \in \mathbb{N}$ mit $t' \geq 0$, $s' \geq 1$ und $v'_i \in V$ für $i \in [t' + s']$ mit $\pi' = v'_0 \dots v'_{t'-1}(v'_t \dots v'_{t'+s'-1})^\omega$, wobei alle v'_i paarweise verschieden sind. Setzt man $a' = \sum_{i=t'}^{t'+s'-1} \frac{n!}{s'} \in \mathbb{Z}$ und $b' = 2n! \sum_{i=0}^{t'-1} r(v'_i) + \sum_{i=t'}^{t'+s'-1} \frac{n!}{s'}(s' - 1 - 2i) \in \mathbb{Z}$, so erhält man für die Partie π' entsprechend:

$$\tilde{R}_\beta(\pi') = \frac{a'}{n!(1-\beta)} + \frac{b'}{2n!}$$

Betrachten wir zunächst den Fall $a = a'$. Ist $b = b'$, so gilt auch $\tilde{R}_\beta(\pi) = \tilde{R}_\beta(\pi')$, und ist $b \neq b'$ dann gilt:

$$|\tilde{R}_\beta(\pi) - \tilde{R}_\beta(\pi')| = \frac{1}{2n!} \cdot |b - b'| \geq \frac{1}{2n!}$$

Es verbleibt der Fall $a \neq a'$. Mit $|a - a'| \in \mathbb{Z}$ gilt:

$$\frac{1}{n!(1-\beta)} |a - a'| \;\geq\; \frac{1}{n!(1-\beta)}$$

und, da $\frac{1}{1-\beta} > 2nr_{\max}$, auch

$$\frac{1}{n!(1-\beta)} \;>\; \frac{2nr_{\max}}{n!}$$

Nach Definition gilt $|b| \leq nr_{\max}$ und $|b'| \leq nr_{\max}$. Somit ist $|b - b'| \leq 2nr_{\max}$ und damit:

$$\frac{2nr_{\max}}{n!} \;\geq\; \frac{1}{n!} |b - b'|$$

Unter Ausnutzung der Transitivität erhält man:

$$\frac{1}{n!(1-\beta)}|a-a'| \;>\; \frac{1}{n!}|b-b'|$$

Also gilt:

$$|\tilde{R}_\beta(\pi) - \tilde{R}_\beta(\pi')| = \left|\frac{1}{n!(1-\beta)}(a-a') + \frac{1}{2n!}(b-b')\right|$$

und, da, wie gerade gezeigt, der erste Summand vom Betrag her mehr als doppelt so groß wie der zweite ist:

$$\left|\frac{1}{n!(1-\beta)}(a-a') + \frac{1}{2n!}(b-b')\right| \;\geq\; \left|\frac{1}{n!(1-\beta)}|a-a'| - \frac{1}{2n!}|b-b'|\right|$$

$$> \;\frac{1}{2n!}|b-b'| \;\geq\; \frac{1}{2n!}$$

Insgesamt gilt somit:

$$\tilde{R}_\beta(\pi) = \tilde{R}_\beta(\pi') \text{ oder } \; |\tilde{R}_\beta(\pi) - \tilde{R}_\beta(\pi')| > \frac{1}{2n!}$$

□

3.7.2 Die diskrete Bewertungsfunktion

Von der im diskreten Algorithmus verwendeten Bewertungsfunktion lässt sich leicht ein Bezug zur approximativen Bewertungsfunktion herstellen. Dieser erlaubt es zu zeigen, dass Verbesserungsschritte des diskreten Algorithmus auch Verbesserungsschritte im Bezug auf die exakte Bewertung sind. Weiter lässt sich zeigen, dass Strategien optimaler diskreter Bewertungen bereits Gewinnstrategien sind. Dies ist letztlich eine zweite Möglichkeit, die Korrektheit des diskreten Algorithmus basierend auf der Korrektheit von Puris Algorithmus zu zeigen.

Das folgende Lemma zeigt, dass die Ordnung $\prec$ auf den Partieprofilen einer Bewertung und damit auf den Partien eine Vergröberung der Ordnung $<$ auf den approximativen Ergebnissen dieser Partien ist.

Lemma 3.21 *Sei $G = (V, E)$ ein Spielgraph mit einer Aufteilung der Knotenmenge $V = V_0 \dot\cup V_1$ auf beide Spieler und eine injektive Färbungsfunktion $c : V \to [2|V|]$ gegeben. Sei $r : V \to \mathbb{R} : r(v) = (-4|V|)^{2c(v)}$ eine Rewardfunktion und $w_{\max} = \max_{v\in V} r(v)$ ihr größtes Gewicht. Sei $\beta \in \mathbb{R}$ mit $1 - \frac{1}{12|V|\cdot|V|!\cdot|w_{\max}|} < \beta < 1$. Sei $\varphi : V \to \mathcal{D}$ eine Bewertung des Spielgraphen. Seien $\pi, \pi' \in V^\omega$ Partien verträglich mit φ. Dann gilt:*

$$\varphi(\pi_0) \prec \varphi(\pi'_0) \implies \tilde{R}_\beta(\pi) < \tilde{R}_\beta(\pi'),$$

wobei π_0 der Anfangsknoten der Partie π ist und π'_0 der Anfangsknoten der Partie π'.

Beweis: Sei $G = (V, E)$ ein Spielgraph mit einer Aufteilung der Knotenmenge $V = V_0 \dot\cup V_1$ auf beide Spieler und eine injektive Färbungsfunktion $c : V \to [2|V|]$ gegeben. Wir setzen $n = |V|$. Sei $r : V \to \mathbb{R} : r(v) = (-4n)^{2c(v)}$ eine Rewardfunktion und $w_{\max} = \max_{v\in V} r(v)$ ihr größtes Gewicht. Sei $\beta \in \mathbb{R}$ mit $1 - \frac{1}{12nn!|w_{\max}|} < \beta < 1$. Sei $\varphi : V \to \mathcal{D}$ eine Bewertung des Spielgraphen. Seien $\pi, \pi' \in V^\omega$ Partien verträglich mit φ.

Für die Partie π existieren dann $s, t \in \mathbb{N}$ mit $t \geq 0$, $s \geq 1$ und $v_i \in V$ für $i \in [t + s]$ mit $\pi = v_0 \dots v_{t-1}(v_t \dots v_{t+s-1})^\omega$, wobei alle v_i paarweise verschieden sind. Wir setzen $w_i = r(v_i)$ für alle $i \in [t + s]$. Für die Partie π' existieren entsprechend $s', t' \in \mathbb{N}$ mit $t' \geq 0$, $s' \geq 1$ und $v'_i \in V$ für $i \in [t' + s']$ mit $\pi' = v'_0 \dots v'_{t'-1}(v'_t \dots v'_{t'+s'-1})^\omega$, wobei alle v'_i paarweise verschieden sind. Wir setzen $w'_i = r(v'_i)$ für alle $i \in [t' + s']$.

Sei $\varphi(v_0) = (u, P, e)$ und $\varphi(v'_0) = (u', P', e')$. Sei die Voraussetzung $\varphi(v_0) \prec \varphi(v'_0)$ gegeben. Es lassen sich die Fälle $u \prec u'$ und $u = u'$ unterscheiden. Zu zeigen ist jeweils:

$$\tilde R_\beta(\pi') - \tilde R_\beta(\pi) > 0.$$

Fall $u \prec u'$: Sei x der relevantere der Knoten u und u'. Sei $w_x = r(x)$ das Gewicht von x. Da $u \prec u'$ gilt nach Definition $(u < u' \wedge u' \in V_+) \vee (u' < u \wedge u \in V_-)$, und daher $(x = u' \wedge x \in V_+) \vee (x = u \wedge x \in V_-)$.
Es gilt mit (3.14) und (3.15):

$$\begin{aligned}\tilde R_\beta(\pi') - \tilde R_\beta(\pi) = {} & \sum_{i=0}^{t'-1} w'_i + \sum_{i=t'}^{t'+s-1} \tfrac{1}{s}\left(\tfrac{1}{1-\beta} + \tfrac{s-1}{2} - i\right) w'_i \\ & - \sum_{i=0}^{t-1} w_i - \sum_{i=t}^{t+s-1} \tfrac{1}{s}\left(\tfrac{1}{1-\beta} + \tfrac{s-1}{2} - i\right) w_i\end{aligned}$$

(durch vorziehen der w_x-Summanden)

$$\begin{aligned} = {} & \tfrac{1}{s(1-\beta)}|w_x| + \sum_{i=0}^{t'-1} w'_i + \sum_{\substack{i=t' \\ v'_i \neq x}}^{t'+s-1} \tfrac{1}{s(1-\beta)} w'_i + \sum_{i=t'}^{t'+s-1} \tfrac{1}{s}\left(\tfrac{s-1}{2} - i\right) w'_i \\ & - \sum_{i=0}^{t-1} w_i - \sum_{\substack{i=t \\ v_i \neq x}}^{t+s-1} \tfrac{1}{s(1-\beta)} w_i - \sum_{i=t}^{t+s-1} \tfrac{1}{s}\left(\tfrac{s-1}{2} - i\right) w_i \end{aligned}$$

(mit Dreiecksungleichung)

$$\begin{aligned} \geq {} & \tfrac{1}{s(1-\beta)}|w_x| - \left| \sum_{\substack{i=t' \\ v'_i \neq x}}^{t'+s-1} \tfrac{1}{s(1-\beta)} w'_i - \sum_{\substack{i=t \\ v_i \neq x}}^{t+s-1} \tfrac{1}{s(1-\beta)} w_i \right| \\ & - \left| \sum_{i=0}^{t'-1} w'_i + \sum_{i=t'}^{t'+s-1} \tfrac{1}{s}\left(\tfrac{s-1}{2} - i\right) w'_i - \sum_{i=0}^{t-1} w_i - \sum_{i=t}^{t+s-1} \tfrac{1}{s}\left(\tfrac{s-1}{2} - i\right) w_i \right| \end{aligned} \tag{3.16}$$

Von dieser Summe schätzen wir den zweiten Summanden ab.

$$\left| \sum_{\substack{i=t' \\ v'_i \neq x}}^{t'+s-1} \frac{1}{s(1-\beta)} w'_i - \sum_{\substack{i=t \\ v_i \neq x}}^{t+s-1} \frac{1}{s(1-\beta)} w_i \right|$$

$$= \frac{1}{s(1-\beta)} \left| \sum_{\substack{i=t' \\ v'_i \neq x}}^{t'+s-1} w'_i - \sum_{\substack{i=t \\ v_i \neq x}}^{t+s-1} w_i \right| \leq \frac{1}{s(1-\beta)} \left| \frac{2s\, w_x}{4n^2} \right| \leq \frac{1}{s(1-\beta)} \left| \frac{w_x}{2n} \right|$$

Für den dritten Summanden von (3.16) ergibt sich:

$$\left| \sum_{i=0}^{t'-1} w'_i + \sum_{i=t'}^{t'+s-1} \frac{1}{s} \left(\frac{s-1}{2} - i \right) w'_i - \sum_{i=0}^{t-1} w_i - \sum_{i=t}^{t+s-1} \frac{1}{s} \left(\frac{s-1}{2} - i \right) w_i \right|$$

$$\leq (t' + n + t + n)|w_{\max}| \leq 4n|w_{\max}| \leq \frac{1}{3n(1-\beta)},$$

wobei letztere Ungleichung aus $1 - \frac{1}{12nn!|w_{\max}|} < \beta$ folgt.
Insgesamt erhält man somit für (3.16):

$$\begin{aligned} &\tfrac{1}{s(1-\beta)}|w_x| - \left| \sum_{\substack{i=t' \\ v'_i \neq x}}^{t'+s-1} \tfrac{1}{s(1-\beta)} w'_i - \sum_{\substack{i=t \\ v_i \neq x}}^{t+s-1} \tfrac{1}{s(1-\beta)} w_i \right| \\ &- \left| \sum_{i=0}^{t'-1} w'_i + \sum_{i=t'}^{t'+s-1} \tfrac{1}{s} \left(\tfrac{s-1}{2} - i \right) w'_i - \sum_{i=0}^{t-1} w_i - \sum_{i=t}^{t+s-1} \tfrac{1}{s} \left(\tfrac{s-1}{2} - i \right) w_i \right| \\ &\geq \frac{1}{s(1-\beta)}|w_x| - \frac{1}{s(1-\beta)} \left| \frac{w_x}{2n} \right| - \frac{1}{3n(1-\beta)} > 0 \end{aligned}$$

Durch Transitivität erhält man:

$$\tilde{R}_\beta(\pi') - \tilde{R}_\beta(\pi) > 0.$$

Fall $\mathbf{u = u'}$: Wird nach uniformen Strategien auf einem Graphen gespielt wird, sind die Schleifen zweier Partien bereits identisch, wenn sie einen gemeinsamen Knoten haben. Somit folgt mit aus $u = u'$ auch $s = s'$ und $\{v_t, \dots v_{t+s-1}\} = \{v'_{t'}, \dots v'_{t'+s-1}\}$. Es gilt:

$$\begin{aligned} \tilde{R}_\beta(\pi') - \tilde{R}_\beta(\pi) = &\sum_{i=0}^{t'-1} w'_i + \sum_{i=t'}^{t'+s-1} \tfrac{1}{s} \left(\tfrac{1}{1-\beta} + \tfrac{s-1}{2} - i \right) w'_i \\ &- \sum_{i=0}^{t-1} w_i - \sum_{i=t}^{t+s-1} \tfrac{1}{s} \left(\tfrac{1}{1-\beta} + \tfrac{s-1}{2} - i \right) w_i \end{aligned}$$

und, da die Schleifen gleich sind:

$$= \sum_{i=0}^{t'-1} w'_i - \sum_{i=t'}^{t'+s-1} \frac{i}{s} w'_i - \sum_{i=0}^{t-1} w_i + \sum_{i=t}^{t+s-1} \frac{i}{s} w_i.$$

Die Knoten in $P \cap P'$ kommen jeweils in beiden Partien im Weg zur Schleife vor. Somit gilt für ihre Gewichte: $\sum\limits_{\substack{i=0 \\ v'_i \in P \cap P'}}^{t'-1} w'_i = \sum\limits_{\substack{i=0 \\ v_i \in P \cap P'}}^{t-1} w_i$ Damit ergibt sich:

$$\sum_{i=0}^{t'-1} w'_i - \sum_{i=t'}^{t'+s-1} \frac{i}{s} w'_i - \sum_{i=0}^{t-1} w_i + \sum_{i=t}^{t+s-1} \frac{i}{s} w_i$$

$$= \sum_{\substack{i=0 \\ v'_i \notin P \cap P'}}^{t'-1} w'_i - \sum_{i=t'}^{t'+s-1} \frac{i}{s} w'_i - \sum_{\substack{i=0 \\ v_i \notin P \cap P'}}^{t-1} w_i + \sum_{i=t}^{t+s-1} \frac{i}{s} w_i.$$

Durch Transitivität erhalten wir:

$$\tilde{R}_\beta(\pi') - \tilde{R}_\beta(\pi)$$

$$= \sum_{\substack{i=0 \\ v'_i \notin P \cap P'}}^{t'-1} w'_i - \sum_{i=t'}^{t'+s-1} \frac{i}{s} w'_i - \sum_{\substack{i=0 \\ v_i \notin P \cap P'}}^{t-1} w_i + \sum_{i=t}^{t+s-1} \frac{i}{s} w_i. \tag{3.17}$$

Wir unterscheiden die Unterfälle $P \prec P'$ und $P = P'$.

Unterfall $\mathbf{u = u' \wedge P \prec P'}$: Da $P \prec P'$ gilt nach Definition: $\max_<(P \triangle P') \in P' \triangle V_-$. Wir setzen $x = \max_<(P \triangle P')$.

Da $x > u$ (nach Definition der zweiten Bewertungskomponente) kommt der Knoten x nicht in der Schleife vor: $x \in \{v_0, \ldots, v_{t-1}\} \cup \{v'_0, \ldots, v'_{t'-1}\}$. Somit kommt das Gewicht von u genau einmal (einmal, weil $x \in P \triangle P'$) als Summand in 3.17 vor. Da $x \in P \triangle P'$ und $x \in P' \triangle V_-$, gilt $x \in P \iff x \in V_-$. Setzt man $w_x = r(x)$, so gilt nach Definition von r auch $x \in V_- \iff w_x < 0$, und somit auch $x \in P \iff w_x < 0$. Nach Definition der zweiten Bewertungskomponente gilt $x \in P \iff x \in \{v_0, \ldots, v_{t-1}\}$. Damit gilt $x \in \{v_0, \ldots, v_{t-1}\} \iff w_x < 0$. In der folgenden Summe lässt sich der w_x enthaltende Summand wie folgt herausziehen:

$$\sum_{\substack{i=0 \\ v'_i \notin P \cap P'}}^{t'-1} w'_i - \sum_{i=t'}^{t'+s-1} \frac{i}{s} w'_i - \sum_{\substack{i=0 \\ v_i \notin P \cap P'}}^{t-1} w_i + \sum_{i=t}^{t+s-1} \frac{i}{s} w_i$$

$$= |w_x| + \sum_{\substack{i=0 \\ v'_i \notin (P \cap P') \cup \{x\}}}^{t'-1} w'_i - \sum_{i=t'}^{t'+s-1} \frac{i}{s} w'_i - \sum_{\substack{i=0 \\ v_i \notin (P \cap P') \cup \{x\}}}^{t-1} w_i + \sum_{i=t}^{t+s-1} \frac{i}{s} w_i$$

Da x der relevanteste Knoten unter denen ist, die genau in einer der beiden Partien vorkommenden, gilt für sein Gewicht: $|w_i| \leq \frac{|w_x|}{4n^2}$ für alle $i \in [s+t]$ mit $v_i \notin (P \cap P') \cup \{x\}$ und $|w'_i| \leq \frac{|w_x|}{4n^2}$ für alle $i \in [s+t']$ mit $v'_i \notin (P \cap P') \cup \{x\}$. Daher gilt:

$$|w_x| + \sum_{\substack{i=0 \\ v_i \neq x}}^{t'-1} w'_i - \sum_{i=t'}^{t'+s-1} \frac{i}{s} w'_i - \sum_{\substack{i=0 \\ v_i \neq x}}^{t-1} w_i + \sum_{i=t}^{t+s-1} \frac{i}{s} w_i$$

$$> |w_x| - (t+n+t'+n)\frac{|w_x|}{4n^2} \;\geq\; 0$$

Durch Transitivität erhält man somit:

$$\tilde{R}_\beta(\pi') - \tilde{R}_\beta(\pi) > 0.$$

Unterfall $\mathbf{u = u' \wedge P = P'}$: Nach Definition von e und e' gilt $u = v_e = v'_{e'}$ und damit auch $w_e = w'_{e'}$. Somit gilt mit (3.17):

$$\tilde{R}_\beta(\pi') - \tilde{R}_\beta(\pi)$$

$$= \sum_{\substack{i=0 \\ v'_i \notin P \cap P'}}^{t'-1} w'_i - \sum_{i=t'}^{t'+s-1} \frac{i}{s} w'_i - \sum_{\substack{i=0 \\ v_i \notin P \cap P'}}^{t-1} w_i + \sum_{i=t}^{t+s-1} \frac{i}{s} w_i.$$

$$= \frac{e-e'}{s} w_e + \sum_{\substack{i=0 \\ v'_i \notin P \cap P'}}^{t'-1} w'_i - \sum_{\substack{i=t' \\ i \neq e'}}^{t'+s-1} \frac{i}{s} w'_i - \sum_{\substack{i=0 \\ v_i \notin P \cap P'}}^{t-1} w_i + \sum_{\substack{i=t \\ i \neq e}}^{t+s-1} \frac{i}{s} w_i$$

Aus $\varphi(\pi) \prec \varphi(\pi')$ folgt für diesen Unterfall: $(u \in V_- \wedge e < e') \vee (u \in V_+ \wedge e > e')$. Ist $u \in V_-$, gilt $w_e < 0$, und ist $u \in V_+$, gilt $w_e > 0$. Daraus folgt: $\frac{e-e'}{s} w_e > 0$. Da aus $e \neq e'$ auch $|e - e'| \geq 1$ folgt, gilt: $\frac{e-e'}{s} w_e > \frac{|w_e|}{s}$. Somit gilt:

$$\frac{e-e'}{s} w_e + \sum_{\substack{i=0 \\ v'_i \notin P \cap P'}}^{t'-1} w'_i - \sum_{\substack{i=t' \\ i \neq e'}}^{t'+s-1} \frac{i}{s} w'_i - \sum_{\substack{i=0 \\ v_i \notin P \cap P'}}^{t-1} w_i + \sum_{\substack{i=t \\ i \neq e}}^{t+s-1} \frac{i}{s} w_i$$

$$\geq \frac{|w_e|}{s} - \left| \sum_{\substack{i=0 \\ v'_i \notin P \cap P'}}^{t'-1} w'_i - \sum_{\substack{i=t' \\ i \neq e'}}^{t'+s-1} \frac{i}{s} w'_i - \sum_{\substack{i=0 \\ v_i \notin P \cap P'}}^{t-1} w_i + \sum_{\substack{i=t \\ i \neq e}}^{t+s-1} \frac{i}{s} w_i \right|$$

Da nur Knoten in P $(= P')$ relevantere Knoten als v_e $(= v'_{e'})$ enthalten sind, gilt für die Gewichte: $|w_i| \leq \frac{|w_e|}{4n^2}$ für alle $i \in [s+t] \setminus \{e\}$ mit $v_i \notin P \cap P'$ und es gilt: $|w'_i| \leq \frac{|w_e|}{4n^2}$ für alle $i \in [s+t'] \setminus \{e'\}$ mit $v'_i \notin P \cap P'$. Somit gilt:

$$\frac{|w_e|}{s} - \left| \sum_{\substack{i=0 \\ v'_i \notin P \cap P'}}^{t'-1} w'_i - \sum_{\substack{i=t' \\ i \neq e'}}^{t'+s-1} \frac{i}{s} w'_i - \sum_{\substack{i=0 \\ v_i \notin P \cap P'}}^{t-1} w_i + \sum_{\substack{i=t \\ i \neq e}}^{t+s-1} \frac{i}{s} w_i \right|$$

$$> \frac{|w_e|}{s} - (t + n + t' + n)\frac{|w_e|}{4n^2} \geq 0$$

Durch Transitivität erhält man somit:

$$\tilde{R}_\beta(\pi') - \tilde{R}_\beta(\pi) > 0.$$

□

Aus Folgerung 3.20 und Lemma 3.21 ergibt sich unmittelbar:

Satz 3.22 *Sei ein Spielgraph mit einer Paritätsbedingung gegeben. Dann ist jeder Verbesserungsschritt des diskreten Algorithmus auch eine Verbesserung im Bezug auf die reelle Bewertung, die sich für das zugehörige DPG mit hinreichend großem Discountfaktor ergibt.*

Wir haben damit eine Brücke zwischen dem kontinuierlichen und dem diskreten Zugang zu den Paritätsspielen geschlagen.

Kapitel 4

Anwendung auf das Model-Checking für den modalen μ-Kalkül

Das Model-Checking-Problem besteht darin, für ein Transitionssystem und eine μ-Kalkülformel, die Knoteneigenschaften beschreibt, zu bestimmen, für welche Knoten des Transitionssystems diese Eigenschaften zutreffen. Ein bekanntes Verfahren zu ihrer Bestimmung ist das so genannte ‚Abrollen' der Fixpunkte (siehe z.B. [Eme96]), das sich auf den Fixpunktsatz von Tarski und Knaster [Tar55] stützt. Wir zeigen in diesem Kapitel, wie sich der im vorigen Kapitel beschriebene Strategiesynthesealgorithmus zur Lösung von Model-Checking-Problemen verwenden lässt. Damit wird der Vergleich mit anderen Model-Checkern möglich.

Dieses Vorgehen ist auch für die Untersuchung des dSV-Algorithmus anhand von Beispielen wichtig. In der Literatur wurden bereits viele Probleme als Model-Checking-Probleme modelliert. Die im folgenden beschriebene Transformation erlaubt sie als Paritätsspiele aufzufassen, und sie somit als Eingaben für den dSV-Algorithmus zu verwenden.

Wir definieren zunächst das Model-Checking-Problem für den modalen μ-Kalkül und geben dann ein Verfahren an, das ein Model-Checking-Problem in ein Paritätsspiel transformiert. Dabei nutzen wir hier die weniger restriktive aber äquivalente Variante der Paritätsspiele, die Knoten ohne ausgehende Kanten zulässt und auch keinen bipartiten Spielgraphen erfordert. Das Verfahren baut auf der von Emerson, Jutla und Sistla in [EJS93] beschriebenen Transformation auf, die eine Zwischendarstellung des Problems als Leerheitsproblem für einen Paritätsbaumautomaten verwendet. Das hier dargestellte Verfahren erzeugt den Spielgraphen ohne eine Zwischendarstellung als Baumautomat. Auf diese Weise bleibt wie auch in [SS98] die Struktur des Problems (d.h. des Transitionsgraphen und der Formel) besser erhalten.

4.1 Der modale μ-Kalkül

Die Formeln des modalen μ-Kalküls werden über kantenbeschrifteten Graphen interpretiert. Zunächst führen wir die Syntax der Formeln des μ-Kalküls ein und geben dann ihre Interpretation über einem kantenbeschrifteten Transitionssystem an.

4.1.1 Syntax

Die Formeln des μ-Kalküls werden auf einer gegebenen Menge von Variablen $\mathcal{V}$ und einer Menge von Beschriftungen L aufgebaut. Formeln sind:

- tt und ff,
- x für jedes $x \in \mathcal{V}$,
- $(\varphi_1 \vee \varphi_2)$ und $(\varphi_1 \wedge \varphi_2)$, falls φ_1 und φ_2 Formeln sind,
- $[l]\,\varphi$ und $\langle l \rangle\,\varphi$, falls $l \in L$ eine Beschriftung und φ eine Formel ist,
- $\mu x.\varphi$ und $\nu x.\varphi$, falls $x \in \mathcal{V}$ eine Variable und φ eine Formel ist.

Die Menge dieser Formeln bezeichnen wir auch mit $\Phi(\mathcal{V}, L)$. Die Menge aller Teilformeln einschließlich ihrer selbst, aus denen eine Formel φ zusammengesetzt ist, wird mit $SF(\varphi)$ bezeichnet. Um Klammern einsparen zu können, legen wir fest, dass einstellige Operatoren die höchste Priorität haben, gefolgt von $\wedge$ mit kleinerer und $\vee$ mit kleinster Priorität. Bei gleicher Priorität wird Linksassoziativität unterstellt.

Formeln der Form $\mu x.\varphi$ nennen wir μ-Formeln und solche der Form $\nu x.\varphi$ entsprechend ν-Formeln.

Ein Beispiel für eine μ-Kalkülformel ist:

$$\nu x.\mu y.([a]\,(x \vee y) \wedge [b]\,y) \tag{4.1}$$

Die Menge $FV(\varphi)$ der *freien Variablen* einer Formel φ definieren wir wie folgt:

- $FV(\mathrm{tt}) = FV(\mathrm{ff}) = \emptyset$,
- $FV(x) = \{x\}$ für jedes $x \in \mathcal{V}$,
- $FV(\varphi_1 \vee \varphi_2) = FV(\varphi_1 \wedge \varphi_2) = FV(\varphi_1) \cup FV(\varphi_2)$, falls φ_1 und φ_2 Formeln sind,
- $FV([l]\,\varphi) = FV(\langle l \rangle\,\varphi) = FV(\varphi)$, falls $l \in L$ eine Beschriftung und φ eine Formel ist,
- $FV(\mu x.\varphi) = FV(\nu x.\varphi) = FV(\varphi) \setminus \{x\}$, falls $x \in \mathcal{V}$ eine Variable und φ eine Formel ist.

Eine Formel φ heißt *geschlossen*, wenn $FV(\varphi) = \emptyset$. Die oben angegebene Beispielformel 4.1 ist somit eine geschlossene Formel.

Zur Beschreibung der Komplexität einer Formel φ verwendet man das Maß der *Alternationstiefe* $AT(\varphi)$. Für eine gegebene Formel unterscheiden wir die Fälle:

- φ enthält nicht zugleich μ- und ν-Subformeln. Dann ist $AT(\varphi) = 0$.
- φ ist keine μ- oder ν-Formel und enthält zugleich μ- und ν-Subformeln. Dann ist $AT(\varphi) = \max\{AT(g) | g \text{ ist eine } \mu\text{- oder } \nu\text{-Subformel}\}$.
- φ ist eine μ-Formel und enthält echte Subformeln, die ν-Formeln sind. Dann ist $AT(\varphi) = 1 + \max\{AT(g) | \text{ ist ein } \nu\text{-Subformel}\}$
- φ ist eine ν-Formel und enthält echte Subformeln, die μ-Formeln sind. Dann ist $AT(\varphi) = 1 + \max\{AT(g) | \text{ ist ein } \mu\text{-Subformel}\}$

Anschaulich betrachtet ist die Alternationstiefe einer Formel die maximale Anzahl von Wechseln zwischen μ und ν, die auf einem unter allen Zweigen des syntaktischen Baumes der Formel vorkommt. Die Beispielformel 4.1 besitzt einen μ-ν-Wechsel und hat somit die Alternationstiefe 1.

Die hier verwendete Definition der Alternationstiefe ist rein syntaktisch. Es existieren auch Definitionen wie in [EL86, Niw86], die stärkere, semantische Eigenschaften verlangen. Für die Umsetzung eines Model-Checking-Problems in ein Paritätsspiel ist die syntaktische Definition interessanter, da sie in Beziehung zur Anzahl der benötigten Farben im Paritätsspiel steht.

4.1.2 Semantik

Die Formeln werden über einem Transitionssystem $T = (S, L, \rightarrow)$ mit Kantenbeschriftungen interpretiert, wobei S eine endliche Menge von Zuständen, L die endliche Menge von Beschriftungen und $\rightarrow \subseteq S \times L \times S$ die Kantenrelation ist. Ein Beispiel für ein einfaches Transitionssystem ist in Abbildung 4.1 gegeben.

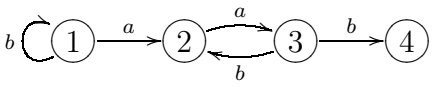


Abbildung 4.1: Transitionssystem.

Sei eine Variablenbelegung $B : \mathcal{V} \rightarrow 2^S$ gegeben. Eine Belegung, die von B nur in der Variablen x abweicht und für x den Wert $S' \subseteq S$ hat, bezeichnen wir mit $B_{x \mapsto S'}$. Sei ein Transitionssystem $T = (S, L, \rightarrow)$ gegeben. Dann wird eine Formel φ mit der Belegung B über dem Transitionssystem T interpretiert, indem man ihr die Menge $\|\varphi\|_B^T$ der Zustände zuordnet, in denen φ gelten soll.

- $\|\mathrm{tt}\|_B^T = S$ und $\|\mathrm{ff}\|_B^T = \emptyset$,
- $\|x\|_B^T = B(x)$,
- $\|\varphi_1 \vee \varphi_2\|_B^T = \|\varphi_1\|_B^T \cup \|\varphi_2\|_B^T$ und $\|\varphi_1 \wedge \varphi_2\|_B^T = \|\varphi_1\|_B^T \cap \|\varphi_2\|_B^T$,
- $\|[l]\,\varphi\|_B^T = \{s \in S | \forall s' \in S : s \xrightarrow{l} s' \Rightarrow s' \in \|\varphi\|_B^T\}$,
- $\|\langle l \rangle\,\varphi\|_B^T = \{s \in S | \exists s' \in S : s \xrightarrow{l} s' \wedge s' \in \|\varphi\|_B^T\}$,
- $\|\mu x.\varphi\|_B^T = \bigcap\{S' \subseteq S \mid \|\varphi\|_{B_{x \mapsto S'}}^T \subseteq S'\}$
- $\|\nu x.\varphi\|_B^T = \bigcup\{S' \subseteq S \mid S' \subseteq \|\varphi\|_{B_{x \mapsto S'}}^T\}$

Die Interpretation einer Formel ist so definiert, dass sie nur von ihren freien Variablen abhängt; somit ist die Interpretation geschlossener Formeln unabhängig von einer Variablenbelegung. Für eine geschlossene Formel φ schreiben wir daher auch $\|\varphi\|^T$.

Als *Model-Checking-Problem* bezeichnet man die Aufgabe, zu einem gegebenen Transitionssystem T und einer darüber interpretierbaren, geschlossenen Formel φ die Menge $\|\varphi\|^T$ der Knoten des Transitionssystem zu bestimmen, in welchen die Formel gilt.

Anschaulich formuliert, beschreibt die Formel 4.1 „die Menge der Knoten, von denen aus auf jedem unendlichen Pfad immer wieder die Beschriftung ‚a' auftritt". In dem Transitionssystem 4.1 gilt diese Formel somit in den Zuständen 2, 3 und 4.

Abweichend von obiger Definition gibt es eine Variante des modalen μ-Kalküls mit Propositionen, die über einer Kripkestruktur interpretiert wird. Die Kripkestruktur ist ein Graph und eine Menge von Propositionen, denen jeweils eine Menge von Knoten des Graphen zugeordnet ist, in denen sie gelten. Es lässt sich jedoch leicht eine Kripkestruktur zu jedem beschrifteten Transitionssystem und umgekehrt konstruieren. Entsprechend werden in den Formeln Propositionen verwendet.

4.2 Reduktion des Model-Checking-Problems

Aus einem Model-Checking-Problem, d.h. einem Transitionssystem und einer μ-Kalkülformel, lässt sich, wie wir in diesem Abschnitt zeigen, ein Paritätsspiel konstruieren, aus dessen Gewinnmenge für Spieler 0 sich die Lösung des Model-Checking-Problems extrahieren lässt.

Dazu definieren wir zunächst ein Transitionssystem, das durch die gegebene Formel induziert wird. Danach definieren wir das synchrone Produkt zweier beschrifteter Transitionssysteme. Schließlich zeigen wir, dass sich das Produkt

aus induziertem und gegebenem Transitionssystem als ein Paritätsspiel auffassen lässt, aus dessen Gewinnbereich für Spieler 0 sich ablesen lässt, in welchen Knoten des Transitionssystems die Formel gilt.

4.2.1 Transitionssysteme aus Formeln

Durch eine Formel φ wird ein Transitionssystem T_φ mit beschrifteten Kanten induziert. Zur Vereinfachung nehmen wir an, dass alle Variablennamen verschieden gewählt sind (durch gebundene Umbenennung kann dies anderenfalls stets erreicht werden). Dabei sind die Zustände dieses Transitionssystems die Subformeln der gegebenen Formel, und die Beschriftungen sind λ und die in der Formel vorkommenden Beschriftungen. Aus den vorkommenden Subformeln ergeben sich auch die Transitionen:

- Aus tt, ff und x entstehen keine Transitionen.
- $\varphi_1 \vee \varphi_2$ liefert die Transitionen $\varphi_1 \vee \varphi_2 \xrightarrow{\lambda} \varphi_1$ und $\varphi_1 \vee \varphi_2 \xrightarrow{\lambda} \varphi_2$.
- $\varphi_1 \wedge \varphi_2$ liefert die Transitionen $\varphi_1 \wedge \varphi_2 \xrightarrow{\lambda} \varphi_1$ und $\varphi_1 \wedge \varphi_2 \xrightarrow{\lambda} \varphi_2$.
- $[l]\,\varphi$ liefert die Transition $[l]\,\varphi \xrightarrow{l} \varphi$.
- $\langle l\rangle\,\varphi$ liefert die Transition $\langle l\rangle\,\varphi \xrightarrow{l} \varphi$.
- $\mu x.\,\varphi$ liefert die Transitionen $\mu x.\,\varphi \xrightarrow{\lambda} \varphi$ und $x \xrightarrow{\lambda} \mu x.\,\varphi$.
- $\nu x.\,\varphi$ liefert die Transitionen $\nu x.\,\varphi \xrightarrow{\lambda} \varphi$ und $x \xrightarrow{\lambda} \nu x.\,\varphi$.

Das durch die Formel 4.1 induzierte Transitionssystem ist in Abbildung 4.2 dargestellt.

4.2.2 Synchrones Produkt von Transitionssystemen

Aus zwei gegebenen beschrifteten Transitionssystemen $T_1 = (S_1, L_1, \to_1)$ und $T_2 = (S_2, L_2, \to_2)$ lässt sich ein Produkt-Transitionssystem $T = T_1 \otimes T_2$ definieren, wobei

$$T = (S_1 \times S_2, L_1 \cup L_2, \to)$$

und für alle $s_1, s_1' \in S_1$ und $s_2, s_2' \in S_2$ und $l \in L_1 \cup L_2$ gilt:

$$\text{falls } l \in L_1 \cap L_2 : \ (s_1, s_2) \xrightarrow{l} (s_1', s_2') \iff s_1 \xrightarrow{l}_1 s_1' \wedge s_2 \xrightarrow{l}_2 s_2'$$

und

$$\text{falls } l \in L_1 \setminus L_2 : \ (s_1, s_2) \xrightarrow{l} (s_1', s_2) \iff s_1 \xrightarrow{l}_1 s_1'$$

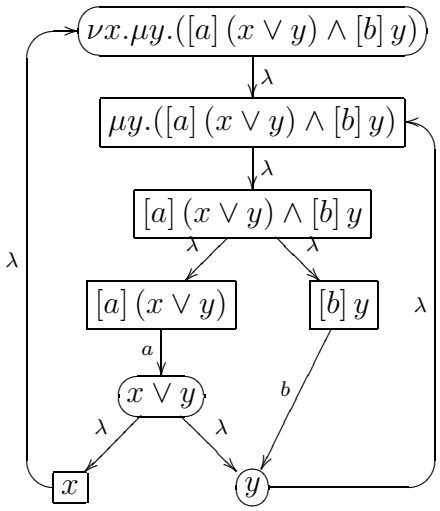


Abbildung 4.2: Durch φ induziertes Transitionssystem.

und

$$\text{falls } l \in L_2 \setminus L_1 : \ (s_1, s_2) \xrightarrow{l} (s_1, s_2') \iff s_2 \xrightarrow{l}_2 s_2'$$

Ein solches Produkt-Transitionssystem wird auch als *synchrones Produkt* bezeichnet.

4.2.3 Ein Paritätsspiel auf einem Produkt-Transitionssystem

Zu einem gegebenen Transitionssystem $T = (S, L, \rightarrow)$ und einer Formel φ und einer Variablenbelegung B mit $\mathcal{V}(\varphi) \subseteq \mathrm{dom}(B)$ lässt sich das Produkt-Transitionssystem $T \otimes T_\varphi = (S_{T,\varphi,B}, L_{T,\varphi,B}, \rightarrow_{T,\varphi,B})$ bestimmen. Durch Weglassen der Kantenbeschriftungen ergibt sich der Spielgraph

$$G_{T,\varphi,B} = (S_{T\otimes T_\varphi}, \rightarrow_{T\otimes T_\varphi}).$$

Ist φ eine geschlossene Formel, so schreiben wir auch $G_{T,\varphi}$ und setzen eine beliebige aber feste Variablenbelegung voraus.

In diesem Spiel gehören Spieler 0 die Knoten der Form:

$$(s, \mathrm{ff}),\ (s, \varphi_1 \vee \varphi_2),\ (s, \langle a \rangle \varphi),\ (s, \nu x.\varphi)$$

und Spieler 1 Knoten der Form:

$$(s, \mathrm{tt}),\ (s, \varphi_1 \wedge \varphi_2),\ (s, [a]\,\varphi),\ (s, \mu x.\varphi).$$

Ein Knoten der Art (s, x) gehört genau dann zu Spieler 1, wenn $s \in B(x)$ und sonst zu Spieler 0.

Um eine Färbung auf $G_{T,\varphi,B}$ zu definieren, definieren wir zunächst eine Färbung der Knoten von $T_\varphi = (S_\varphi, L_\varphi, \rightarrow_\varphi)$. Sei

$$S^\mu_\varphi = \{\psi \in SF(\varphi) \mid \exists \psi' \in SF(\psi)\, \exists y \in \mathcal{V}(\psi) : \psi = \mu y.\, \psi'\}$$
$$S^\nu_\varphi = \{\psi \in SF(\varphi) \mid \exists \psi' \in SF(\psi)\, \exists y \in \mathcal{V}(\psi) : \psi = \nu y.\, \psi'\}$$

Dann erhalten wir die Färbung $c' : S_\varphi \rightarrow \mathbb{N}$ durch

$$c'(\psi) = 0, \text{ falls } \psi \notin S^\mu_\varphi \cup S^\nu_\varphi$$
$$c'(\psi) = \max(\{0\} \cup \{f | \exists \psi' \in SF(\psi) : \psi' \in S^\mu_\psi \wedge f = c(\psi') + 1\}), \text{ falls } \psi \in S^\nu_\varphi.$$
$$c'(\psi) = \max(\{1\} \cup \{f | \exists \psi' \in SF(\psi) : \psi' \in S^\nu_\psi \wedge f = c(\psi') + 1\}), \text{ falls } \psi \in S^\mu_\varphi.$$

Die Farben für Knoten des Spielgraphen entsprechen jeweils der Farbe der Formel des Knotens:

$$c((s, \psi)) = c'(\psi).$$

Beispiel

Für die Knoten des durch die Formel 4.1 induzierten Transitionssystems aus Abbildung 4.2 ergeben sich folgende Farben: Der Knoten $\nu x.\mu y.([a]\,(x \vee y) \wedge [b]\,y)$ hat die Farbe 2, der Knoten $\mu y.([a]\,(x \vee y) \wedge [b]\,y)$ hat die Farbe 1 und alle anderen Knoten die Farbe 0.

In Abbildung 4.3 wird der Spielgraph, der sich aus der Formel 4.1 und dem Transitionssystem in Abbildung 4.1 ergibt, dargestellt.

Betrachten wir diesen Spielgraph, so ergeben sich folgende Gewinnbereiche (in Abbildung 4.4 sind die Strategien durch durchgezogene und nicht benutzte Alternativen durch gestrichelte Kanten dargestellt):

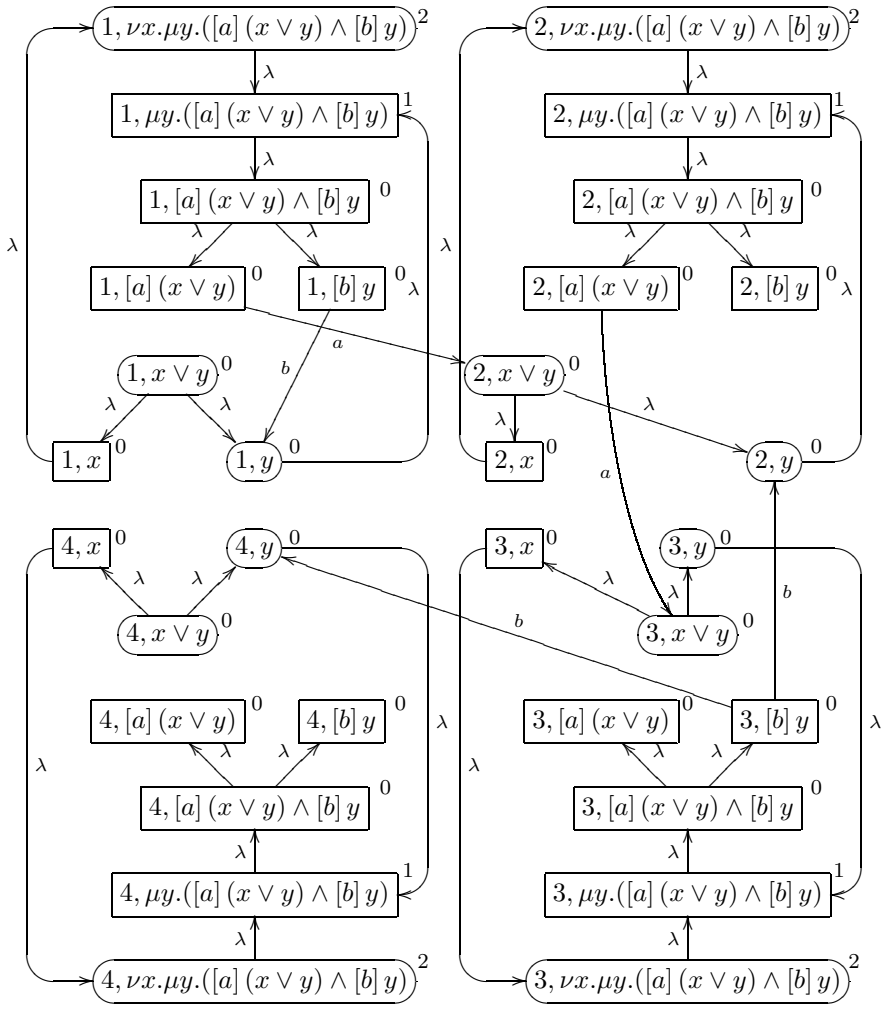
Jede Partie, die in einem Knoten mit 4 in der ersten Komponente beginnt, endet in $(4, [a]\,(x \vee y))$ oder $(4, [b]\,y)$. Diese beiden Knoten gehören Spieler 1, wodurch Spieler 0 die Partien gewinnt. Jede Partie, die in einem Knoten mit 3 in der ersten Komponente beginnt, endet in $(3, [a]\,(x \vee y))$ oder erreicht $(3, [b]\,y)$. Endet sie in $(3, [a]\,(x \vee y))$, so gewinnt Spieler 0.

Jede Partie, die in einem Knoten mit 2 in der ersten Komponente beginnt, endet in $(2, [b]\,y)$ oder $(3, [a]\,(x \vee y))$ oder erreicht $(3, [b]\,y)$. Endet sie in $(2, [b]\,y)$ oder $(3, [a]\,(x \vee y))$, so gewinnt Spieler 0.

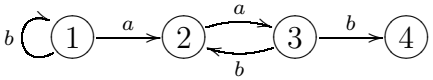
Betrachten wir Partien, die Knoten $(3, [b]\,y)$ erreichen. Zieht Spieler 1 von $(3, [b]\,y)$ nach $(4, y)$, so gewinnt Spieler 0. Zieht Spieler 1 von $(3, [b]\,y)$ nach $(2, y)$, so sind zwei verschiedene unendliche Partien (alle endlichen Partien haben wir bereits betrachtet) denkbar:

$(3, [b]\,y), (2, y),\ (2, \mu y.([a]\,(x \vee y) \wedge [b]\,y)),\ (2, [a]\,(x \vee y) \wedge [b]\,y),\ (2, [a]\,(x \vee y)),\ (3, x \vee y),\ (3, y), (3, \mu y.([a]\,(x \vee y) \wedge [b]\,y)),\ (3, [a]\,(x \vee y) \wedge [b]\,y),\ \ldots$

oder



Produktsystem zum Transitionssystem wie in Abbildung 4.1:



und der Formel (4.1):

$$\nu x.\mu y.([a]\,(x \vee y) \wedge [b]\,y)$$

Abbildung 4.3: Produktsystem als Spielgraph.

$(3, [b]\,y)$, $(2, y), (2, \mu y.([a]\,(x \vee y) \wedge [b]\,y))$, $(2, [a]\,(x \vee y) \wedge [b]\,y)$, $(2, [a]\,(x \vee y))$, $(3, x \vee y)$, $(3, x)$, $(3, \nu x.\mu y.([a]\,(x \vee y) \wedge [b]\,y))$, $(3, \mu y.([a]\,(x \vee y) \wedge [b]\,y))$, $(3, [a]\,(x \vee y) \wedge [b]\,y)$, ...

Die beiden Alternativen ergeben sich dadurch, dass Spieler 0 sich in Knoten $(3, x \vee y)$ zwischen den Nachfolgern $(3, y)$ und $(3, x)$ entscheiden kann. Wählt er wie in der ersten Variante $(3, y)$, so sind in dieser Schleife die Knoten mit größter Farbe: $(2, \mu y.([a]\,(x \vee y) \wedge [b]\,y))$ und $(3, \mu y.([a]\,(x \vee y) \wedge [b]\,y))$. Sie haben die Farbe 1, wodurch Spieler 1 eine solche Partie gewinnen würde. Wählt Spieler 0 jedoch den Nachfolger $(3, x)$ wie in der zweiten Variante, so hat in dieser Schleife der Knoten $(3, \nu x.\mu y.([a]\,(x \vee y) \wedge [b]\,y))$ die größte Farbe; diese Farbe ist 2, wodurch Spieler 0 die Partie gewinnt. Also wählt Spieler 0 die zweite Variante. Damit gewinnt Spieler 0 alle Partien, deren Anfangsknoten $2, 3$ oder 4 in der ersten Komponente hat.

Der einzige Nachfolger von $(1, [a]\,(x \vee y))$ ist $(2, x \vee y)$. Damit gewinnt Spieler 0 auch Partien mit diesem Anfangsknoten. Alle anderen Partien, die in Knoten mit 1 in der ersten Komponente beginnen, erreichen den Knoten $(1, [a]\,(x \vee y) \wedge [b]\,y)$. Hier wählt Spieler 1 den Nachfolger $(1, [b]\,y)$, wodurch die folgende Schleife entsteht:

$(1, [a]\,(x \vee y) \wedge [b]\,y)$, $(1, [b]\,y)$, $(1, y)$, $(1, \mu y.([a]\,(x \vee y) \wedge [b]\,y))$, ...

Der Knoten mit größter Farbe in dieser Schleife ist $(1, \mu y.([a]\,(x \vee y) \wedge [b]\,y))$ mit der Farbe 1. Damit gewinnt Spieler 1 diese Partien.

Im folgenden Hilfssatz wird gezeigt, in welcher Weise die Gewinnbereiche eines Paritätsspiels, das aus einer geschlossenen Formel und einem beschrifteten Transitionssystem entstanden ist, mit dem Gelten der Formel und ihrer Teilformeln in den Knoten des Transitionssystems zusammenhängt.

Hilfssatz 4.1 *Sei L eine Menge von Beschriftungen, $T = (S, L, \rightarrow)$ ein Transitionssystem, $\mathcal{V}$ eine Variablenmenge und $\varphi \in \Phi(\mathcal{V}, L)$ eine Formel mit Beschriftungen aus L. Dann gilt für jede Variablenbelegung $B : \mathcal{V} \rightarrow 2^S$:*

$$\|\varphi\|_B^T = proj_1\big(W_0(G_{T,\varphi,B}) \cap (S \times \{\varphi\})\big),$$

wobei $proj_1$ die Projektion auf die erste Komponente ist.

Für das Beispiel aus Abbildung 4.3 gilt, wenn T das Transitionssystem, φ die Formel und $B : \{x, y\} \rightarrow 2^{\{1,2,3,4\}}$ eine beliebige Belegung bezeichen:

$$\|\varphi\|_B^T = \{2, 3, 4\} = proj_1\big(W_0(G_{T,\varphi,B}) \cap (\{1, 2, 3, 4\} \times \{\varphi\})\big).$$

Die Belegung ist für die Gewinnbereiche unerheblich, da die Knoten, deren Spielerzugehörigkeit sie beeinflusst alle genau eine ausgehende Kante haben.

Beweis von Hilfssatz 4.1: Sei L eine Menge von Beschriftungen, $T = (S, L, \rightarrow)$ ein Transitionssystem, $\mathcal{V}$ eine Variablenmenge und $\varphi \in \Phi(\mathcal{V}, L)$ eine Formel mit

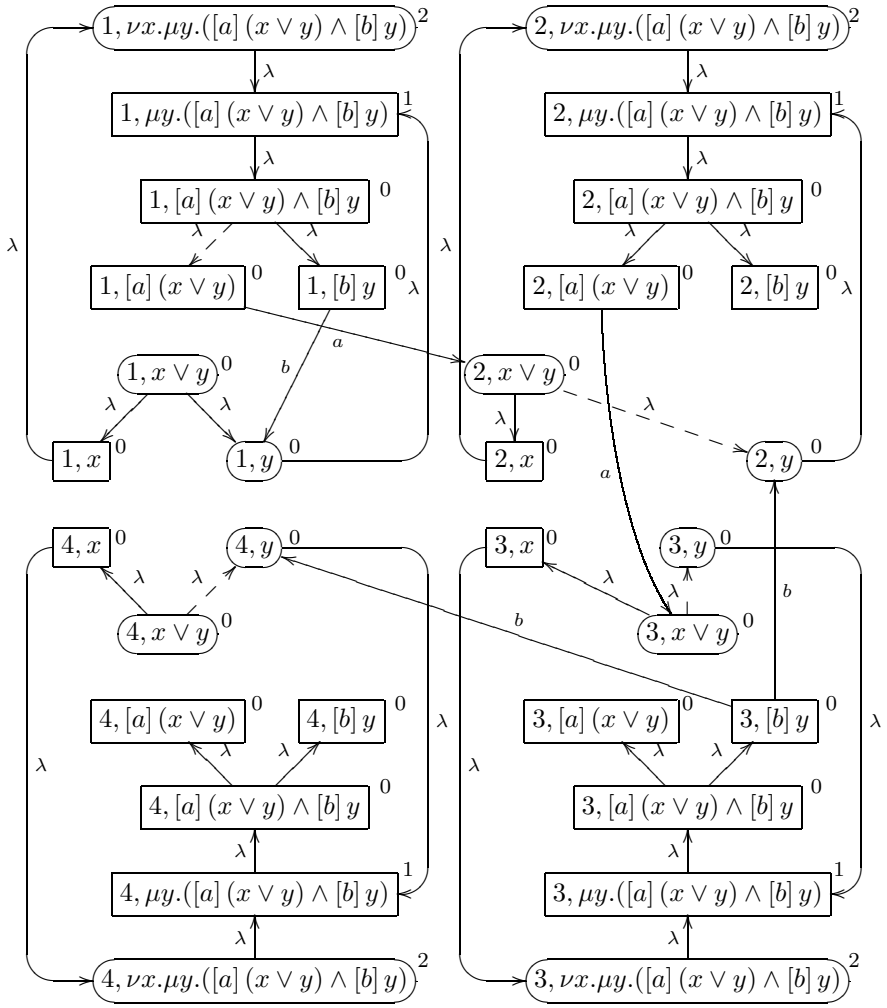


Abbildung 4.4: Produktsystem als Spielgraph mit Gewinnstrategien.

Beschriftungen aus L und $B : \mathcal{V} \to 2^S$ eine Variablenbelegung. Wir führen den Beweis durch Induktion über den Aufbau der Formel φ.

Induktionsanfang: Unmittelbar aus der Konstruktion des Spiels ergibt sich:

- Wenn $\varphi = \mathrm{tt}$ ist, gilt $\|\varphi\|_B^T = S = proj_1\big(W_0(G_{T,\varphi,B}) \cap (S \times \{\varphi\})\big)$.
- Wenn $\varphi = \mathrm{ff}$ ist, gilt $\|\varphi\|_B^T = \emptyset = proj_1\big(W_0(G_{T,\varphi,B}) \cap (S \times \{\varphi\})\big)$.
- Wenn $\varphi = x$ ist, gilt $\|\varphi\|_B^T = B(x) = proj_1\big(W_0(G_{T,\varphi,B}) \cap (S \times \{\varphi\})\big)$.

Induktionsvoraussetzung: Für alle $\varphi' \in SF(\varphi) \setminus \{\varphi\}$ und jede Variablenbelegung $B' : \mathcal{V} \to 2^S$ gilt:

$$\|\varphi'\|_{B'}^T = proj_1\big(W_0(G_{T,\varphi',B'}) \cap (S \times \{\varphi'\})\big).$$

Induktionsschritt:

- Wenn $\varphi = [a]\,\varphi'$ ist, dann ist $G_{T,\varphi',B}$ ein Teilspiel von $G_{T,\varphi,B}$, wie in Hilfssatz 2.8 (ein Teilgraph in dem einige Knoten ohne eingehende Kanten fehlen), so dass $W_0(G_{T,\varphi,B}) \cap (S \times SF(\varphi')) = W_0(G_{T,\varphi',B})$. Für einen beliebigen Knoten $s \in S$ gewinnt Spieler 0 auf (s,φ) gdw. alle ausgehenden Kanten zu Gewinnknoten führen. Alle Nachfolger sind stets Knoten im Spielgraph $G_{T,\varphi',B}$. Somit erhält man für die übrigen Knoten:

$$\begin{aligned}
\|\varphi\|_B^T &= \{s \in S | \forall s' \in S : s \xrightarrow{a} s' \;\Rightarrow\; s' \in \|\varphi'\|_B^T\} \\
&= \{s \in S | \forall s' \in S : (s,\varphi) E (s',\varphi') \Rightarrow s' \in \|\varphi'\|_B^T\} \\
&= \{s \in S | \forall s' \in S : (s,\varphi) E (s',\varphi') \Rightarrow (s',\varphi') \in W_0(G_{T,\varphi',B})\} \\
&= proj_1\big(W_0(G_{T,\varphi,B}) \cap (S \times \{\varphi\})\big).
\end{aligned}$$

- Wenn $\varphi = \varphi_1 \wedge \varphi_2$ ist, dann sind $G_{T,\varphi_1,B}$ und $G_{T,\varphi_2,B}$ Teilspiele von $G_{T,\varphi,B}$, wie in Hilfssatz 2.8, so dass $W_0(G_{T,\varphi,B}) \cap (S \times SF(\varphi_i)) = W_0(G_{T,\varphi_i,B})$ für $i \in \{1,2\}$ gilt.

  Für einen beliebigen Knoten $s \in S$ gewinnt Spieler 0 auf (s,φ) gdw. die beiden über ausgehende Kanten erreichbaren Knoten (s,φ_1) und (s,φ_2) Gewinnknoten sind. Dies ist gilt jedoch gdw. $\varphi_1 \wedge \varphi_2$ gilt. Somit ergibt sich mit der Induktionsvoraussetzung:

$$\begin{aligned}
\|\varphi\|_B^T &= \|\varphi_1\|_B^T \cap \|\varphi_2\|_B^T \\
&= proj_1\big(W_0(G_{T,\varphi_1,B}) \cap (S \times \{\varphi_1\})\big) \cap proj_1\big(W_0(G_{T,\varphi_2,B}) \cap (S \times \{\varphi_2\})\big) \\
&= proj_1\big(W_0(G_{T,\varphi,B}) \cap (S \times \{\varphi\})\big).
\end{aligned}$$

- Fall $\varphi = \mu x.\varphi'$: Für eine beliebige Belegung $B' : \mathcal{V} \to 2^S$ sei V die Knotenmenge und E die Kantenmenge von $G_{T,\varphi,B'}$. Sei $n = |V|$ und $E' = E \setminus (V \times (S \times \{\varphi\}))$ eine Teilmenge der Kanten. Für den Spielgraph $G'_{T,\varphi,B'} = (V, E')$

mit der gleichen Paritätsbedinung wie $G_{T,\varphi,B'}$ zeigen wir zunächst die folgende Hilfsbehauptung:

$$\|\varphi'\|^T_{B'} = proj_1(W_0(G'_{T,\varphi,B'}) \cap (S \times \{\varphi\})) \tag{4.2}$$

Der Spielgraph $G'_{T,\varphi,B'}$ ist ein Teilspiel von $G_{T,\varphi,B'}$, wie in Hilfssatz 2.8, so dass $W_0(G'_{T,\varphi,B'}) \cap (S \times SF(\varphi')) = W_0(G_{T,\varphi',B'})$.

Da jeder Knoten $(s,\varphi) \in S \times \{\varphi\}$ genau den einen Nachfolger $(s,\varphi') \in V$ hat, gilt:

$$proj_1(W_0(G_{T,\varphi',B'}) \cap (S \times \{\varphi'\})) = proj_1(W_0(G'_{T,\varphi,B'}) \cap (S \times \{\varphi\})).$$

Nach Induktionsvoraussetzung gilt:

$$\|\varphi'\|^T_{B'} = proj_1(W_0(G_{T,\varphi',B'}) \cap (S \times \{\varphi'\})).$$

Durch Transitivität ergibt sich die Hilfsbehauptung.

Wir rollen den kleinsten Fixpunkt φ von $\varphi'(x)$ ab:

$$\begin{aligned} S_0 &= \|\varphi'\|^T_{B_{x\mapsto\emptyset}} \\ S_i &= \|\varphi'\|^T_{B_{x\mapsto S_{i-1}}} \text{ für } i = 1,\dots n \end{aligned}$$

Dann gilt nach dem Satz von Tarski und Knaster [Tar55]:

$$\|\varphi\|^T_B = \|\varphi'\|^T_{B_{x\mapsto S_{n-1}}}. \tag{4.3}$$

Mit (4.2) gilt außerdem:

$$\begin{aligned} S_0 &= proj_1(W_0(G'_{T,\varphi,B_{x\mapsto\emptyset}}) \cap (S \times \{\varphi\})) \\ S_i &= proj_1(W_0(G'_{T,\varphi,B_{x\mapsto S_{i-1}}}) \cap (S \times \{\varphi\})) \text{ für } i = 1,\dots n \end{aligned}$$

Die Folge

$$\left(G'_{T,\varphi,B_{x\mapsto\emptyset}},\ G'_{T,\varphi,B_{x\mapsto S_0}},\ \dots,\ G'_{T,\varphi,B_{x\mapsto S_{n-1}}}\right)$$

von Spielgraphen ist genau die Auffaltung von $G_{T,\varphi,B}$. Die notwendigen Knotenaufteilungen werden dabei durch die Wahl der Bewertungsfunktion für die Variable x festgelegt. Nach Hilfssatz 2.12 über die Auffaltung gilt:

$$W_0(G'_{T,\varphi,B_{x\mapsto S_{n-1}}}) = W_0(G_{T,\varphi,B}). \tag{4.4}$$

Damit ergibt sich:

$$\begin{aligned} \|\varphi\|^T_B &\overset{(4.3)}{=} \|\varphi'\|^T_{B_{x\mapsto S_{n-1}}} \\ &\overset{(4.2)}{=} proj_1(W_0(G'_{T,\varphi,B_{x\mapsto S_{n-1}}}) \cap (S \times \{\varphi\})) \\ &\overset{(4.4)}{=} proj_1\big(W_0(G_{T,\varphi,B}) \cap (S \times \{\varphi\})\big). \end{aligned}$$

- Für $\varphi = \langle a \rangle \varphi'$, $\varphi = \varphi_1 \vee \varphi_2$ und $\varphi = \nu x.\varphi'$ ist der Beweis jeweils dual zu $\varphi = [a] \varphi'$, $\varphi = \varphi_1 \wedge \varphi_2$ und $\varphi = \mu x.\varphi'$.

□

Für den Fall geschlossener Formeln erhält man folgende einfachere Formulierung des Satzes:

Satz 4.2 *Sei $T = (S, L, \rightarrow)$ ein Transitionssystem, wobei S die Menge der Knoten und L die Menge seiner Beschriftungen ist, und φ eine geschlossene μ-Kalkülformel mit Beschriftungen aus L. Dann gilt:*

$$\|\varphi\|^T = proj_1\big(W_0(G_{T,\varphi}) \cap (S \times \{\varphi\})\big),$$

wobei $proj_1$ die Projektion auf die erste Komponente ist.

Der Beweis dieses Satzes folgt unmittelbar durch Anwendung des vorangehenden Hilfssatzes 4.1.

4.3 Diskussion

Es wurde eine direkte Transformation von Model-Checking-Problemen (d.h.: jeweils Transitionssystem und μ-Kalkülformel) in Paritätsspiele vorgestellt. Im Gegensatz dazu erfordert die in [EJS93] vorgestellte Methode einen Zwischenschritt über Paritätsbaumautomaten. Für diesen Zwischenschritt muss der Produktgraph derart transformiert werden, dass jeder Knoten höchstens zwei Nachfolger hat. Dieses führt zu einer Vervielfachung der Knoten, deren Anzahl im ungünstigsten Fall quadratisch anwächst.

Kapitel 5

Implementierung

Der in Kapitel 2 beschriebene Algorithmus zur Strategiekonstruktion für Paritätsspiele ist in dem System `omega` ([SV00b]) implementiert. Das System `omega` ist eine Plattform zur experimentellen Untersuchung von Algorithmen der ω-Automatentheorie. Es stellt u. a. Algorithmen zur Erzeugung von Paritätsspielen und deren Auswertung zur Verfügung. Eine dieser Möglichkeiten, ein Paritätsspiel zu erzeugen, besteht darin, es, wie in Kapitel 4 beschrieben, aus einem Model-Checking-Problem zu generieren. Im ersten Abschnitt dieses Kapitels wird auf die Strategiesynthese und die Schnittstelle zur direkten Eingabe von Paritätsspielen eingegangen. Der zweite Abschnitt geht auf die Eingabe in Form eines Model-Checking-Problems ein und es werden experimentelle Erfahrungen mit dem dSV-Algorithmus in diesem Kontext diskutiert.

Das System `omega` ist in der Sprache Common Lisp implementiert. Nur der Algorithmus zur Strategiekonstruktion selbst ist aus Performancegründen in optimiertem C geschrieben. Die Benutzungsschnittstelle ist in die Sprache Common Lisp eingebettet. Dies ermöglicht es, den vollen Sprachumfang von Common Lisp für die Konstruktion von Eingaben zu nutzen. So können Generatoren für Paritätsspiele definiert werden, die skalierbare Beispiele erzeugen, oder es können auch Schnittstellen zu anderen Systemen geschaffen werden.

5.1 Strategiesynthese für Paritätsspiele

Im diesem Abschnitt stellen wir die Implementierung des dSV-Algorithmus in `omega` dar. Wir beschreiben Ein- und Ausgabeformate des Programms und gehen auf die implementierten Optimierungen des Algorithmus ein.

5.1.1 Eingabesprache

Die Eingabesprache ist in Common Lisp eingebettet. Common Lisp ist dabei um Funktionen und Makros erweitert, die eine einfache Beschreibung von Paritäts-

spielgraphen erlauben.

Der Synthesealgorithmus kann aus der Common Lisp Umgebung heraus unter dem Funktionsnamen `strategy` aufgerufen werden:

```
(strategy vertices-of-player0 vertices-of-player1 edge-list colors order)
```

Die Parameter `vertices-of-player0` und `vertices-of-player1` sind die Listen der Knoten, die Spieler 0 bzw. 1 gehören. Der Parameter `edge-list` ist die Liste der Kanten des Spielgraphen, wobei jede Kante selbst eine Liste der Form `(quelle ziel)` mit den Knoten `quelle` und `ziel` ist. Der Parameter `colors` ist eine Liste `(color0 color1 ... colorn)`, in welcher jeweils `colori` eine Liste der Knoten mit Farbe i ist, gewöhnlich in aufsteigender Relevanz angegeben.

Der Parameter `order` ist optional; Wird er weggelassen, was dem Wert `:last-best` entspricht, so wird vorausgesetzt, dass die Farben in aufsteigender Relevanz geordnet sind. Ist der Parameter `order` auf `:first-best` gesetzt, so wird vorausgesetzt, dass die Farben in absteigender Relevanz angeordnet sind, genau wie die Knoten in jeder Liste `colori`.

Ein Knoten kann durch ein beliebiges Lisp-Objekt bezeichnet werden. Die Gleichheit zweier Knotenbezeichner wird durch die Funktion `equal` bestimmt.

Beispiel

Wir betrachten den Paritätsspielgraphen $G_2 = (V_0, V_1, E, c)$ with $V_0 = \{2, 4\}$, $V_1 = \{1, 3\}$, $E = \{(1, 2), (2, 1), (2, 3), (3, 4), (4, 3)\}$ und $c(1) = 1$, $c(2) = 1$, $c(3) = 1$, $c(4) = 0$, der in Abbildung 5.1 dargestellt ist. Wir bezeichnen die Knoten in V_0 durch Kreise und die Knoten in V_1 durch Quadrate.

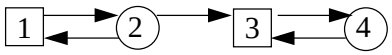


Abbildung 5.1: Beispielgraph G_2.

Die Prozedur zur Berechnung der Gewinnstrategien und der zugehörigen Gewinnstrategien wird wie folgt aufgerufen:[1]

```
(strategy '(2 4)
          '(1 3)
         '((1 2) (2 1) (2 3) (3 4) (4 3))
          '((4) (3 2 1))
          :first-best)
```

Anstatt die Elemente der Parameter explizit im Funktionsaufruf aufzuzählen, können wir eine Funktion benutzen, die die notwendigen Parameter zum Aufruf von `strategy` generiert.

[1] Das Anführungszeichen vor einem Parameter sagt dem Interpreter, dass es sich um eine Konstante handelt; anderenfalls würde die folgende Liste als eine Funktionsaufruf interpretiert werden.

Makrosprache

Für größere Beispiele ist eine algorithmische Definition von Spielgraphen geeigneter. Wir entwickeln im Folgenden eine Notation für solche Definitionen, wiederum in Form eines Lisp Makros, mit dem Namen `parity-game`. Es stellt eine Umgebung zur Verfügung, in der die Makros `vertex`, `edge` und `dedge` zur Verfügung stehen. Mit ihrer Hilfe werden Knoten und Kanten eingeführt. Dabei wird mit `edge` eine gerichtete Kante eingeführt, während mit `dedge` zwei gerichtete Kanten in Hin- und Rückrichtung eingeführt werden.

Beispiel

Wir zeigen die Verwendung dieser Umgebung für die Graphen G_n in Abbildung 5.2.

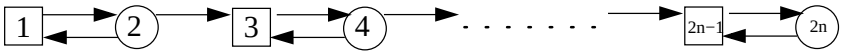


Abbildung 5.2: Beispielgraph G_n.

Die Spezifikation in einer Pascal-artigen Notation sieht wie folgt aus:

```
for v:= 1 to n * 2 do
  v is vertex with:
    if even then v in V_0
            else v in V_1
    if v = 2 * n then color(v) = 0
                 else color(v) = 1
for i:= 1 to n do
  (2 * i - 1, 2 * i) is d-edge
for i:= 1 to n - 1 do
  (2 * i, 2 * i + 1) is edge
```

Der entsprechende Lisp-Ausdruck hat die folgende Form (hier dargestellt unter Benutzung von Infixnotation):

```
(parity-game
  (for (v 1 n*2)
    (vertex (v)
            (if (evenp v) 0 1)
            (if v=n*2 0 1)))
  (for (i 1 n)
       (dedge (i*2-1) (i*2)))
  (for (i 1 n-1)
       (edge (i*2) (i*2+1))))
```

Die Einführung der Kanten ist selbsterklärend. Für die Knoten müssen außer dem Namen auch der Spieler, dem dieser Knoten gehört, und seine Farbe angegeben werden. (Im Beispiel gehören die geraden Knoten Spieler 0 und die anderen Spieler 1, die Farbe von Knoten $2n$ ist 0 und für alle anderen Knoten 1.) In dieser Umgebung ist ein Knoten stets eine Liste. Die bedingten Anweisungen sind wie üblich geschrieben, mit einer if-Anweisung und den zwei Ausdrücken für den Ja- und den Nein-Fall. Außerdem steht eine for-Schleife mit der üblichen Bedeutung zur Verfügung.

Für die weiter unten diskutierten Beispiele benutzen wir ebenfalls dasselbe Format.

5.1.2 Ausgabe

Die Ausgabe enthält die folgenden Daten: die berechneten Gewinnbereiche von Spieler 0 und 1, optimale Strategien für beide Spieler und die Anzahl der Iterationen (Verbesserungsschritte), die in der Durchführung des Algorithmus gebraucht wurden. Letzterer ist der kritische Parameter, der polynomielles von exponentiellem Verhalten des Algorithmus trennt.

Die gelieferten Strategien sind Gewinnstrategien auf den jeweiligen Gewinnbereichen. Auf dem Komplement berechnete Strategien sind offensichtlich keine Gewinnstrategien.

Beispiel

Für das Beispiel aus Abbildung 5.1 erhält man die folgende Ausgabe (die sagt, dass Spieler 0 überall gewinnt, indem er von jedem Knoten aus V_0 die angegebene Kante wählt):

```
(4 3 2 1)     ; Gewinnbereich von Spieler 0
nil           ; Gewinnbereich von Spieler 1
((2 3) (4 3)) ; Gewinnstrategie Spieler 0
nil           ; Gewinnstrategie Spieler 1
2             ; Anzahl der Iterationen
```

Kantentraversierungen

Zusätzliche Informationen stehen in einer weiteren Datei mit folgenden Daten zur Verfügung:

1. Anzahl der Iterationen.

2. Vorhandensein von Strategieänderungen, die zur Änderung eines relevantesten Schleifenknotens führen. (Diese Iterationen führen zu großen Veränderungen, da eine neue Schleife erreicht werden muss.)

3. Anzahl der Knoten von Spieler 0, für die sich die Strategie ändert.

4.–6. Anzahl der Kantentraversierungen in verschiedenen Funktionsaufrufen.

7. Summe aller Kantentraversierungen. (Die Zahl der Kantentraversierungen bildet ein wichtiges Maß für Zeitaufwand des Algorithmus. Eine Kantentraversierung kann als atomare Operation des Algorithmus angesehen werden. Die Anzahl anderer ausgeführter Operationen im Algorithmus ist linear durch die Anzahl der Kantentraversierungen beschränkt.)

Verbesserungsspur

Die komplizierte Eigenschaft des dSV-Algorithmus besteht darin, dass er die Möglichkeit hat, die Strategie für alle Knoten, etwa von Spieler 0, in jedem Iterationsschritt zu ändern. Einen interessanten Überblick über den Ablauf des Algorithmus ergibt sich somit aus einer Matrix, in welcher jede Zeile die Ergebnisse einer gegebenen Iteration für die Knoten von Spieler 0 beschreibt. Wir nennen diese Matrix die *Verbesserungsspur*. Diese wird in einer weiteren Datei in dem folgenden Format zur Verfügung gestellt. Jeder Knoten von Spieler 0 wird in der Reihenfolge aufsteigender Relevanz einer Spalte zugeordnet. Die i-te Zeile enthält das Ergebnis der i-ten Iteration. Ein Eintrag an Position (i, j) in der Matrix sagt, ob und wie sich die Strategie für Knoten j (mit Bezug auf die gegebene Ordnung) in der i-ten Iteration geändert hat. Wir unterscheiden zwei Arten von Einträgen: Stern und Punkt. Damit unterscheiden wir, ob in dem zugehörigen Verbesserungsschritt sich der mit Knoten j assoziierte relevanteste Schleifenknoten in der Iteration i verändert hat. Hat er sich geändert, so ist der Eintrag ein Stern, anderenfalls ein Punkt.

5.1.3 Optimierungen der Implementierung

Die folgenden in Abschnitt 3.6.1 vorgeschlagenen Änderungen wurden zur Verbesserung der Effizienz der Implementierung berücksichtigt:

- Nur Strategien von Knoten, von denen aus Spieler 0 mit der aktuellen Strategie nicht gewinnt, werden verbessert. Um die erste Komponente der Bewertungen zu bestimmen, werden die Knoten bestimmt, die einen Pfad zu dem relevantesten Knoten einer festen Schleife besitzen. Hier werden als relevanteste Knoten nur negative berücksichtigt.

- Es werden nur vereinfachte Bewertungen berechnet. Die zweiten Komponenten werden aufgebaut, indem beginnend mit der leeren Menge in absteigender Relevanz Knoten hinzugefügt werden. Die Bewertungen von Knoten werden immer beginnend bei dem Knoten, der hinzugefügt wird, aktualisiert und danach die Vorgänger dieses Knotens aktualisiert, usw. Unter den

Vorgängern werden Knoten höherer Relevanz und deren Vorgänger ausgelassen, womit man unmittelbar die vereinfachte Bewertung erhält.

- Die Funktion *subvaluation*() (siehe Abbildung 3.8) wird zur Berechnung der zweiten Komponenten der Bewertungen nicht für jede durch *valuation*() (siehe Abbildung 3.7) bestimmte Schleifenkomponente einzeln aufgerufen. Dieses hätte den Nachteil, dass die Knoten der Schleifenkomponenten jeweils nach Relevanz geordnet werden müssten. Stattdessen werden zunächst alle negativen ersten Komponenten der Bewertung berechnet. Danach werden die zweiten Komponenten berechnet, indem die Hauptschleife von *subvaluation*() zur Berechnung der zweiten Komponenten einmal in absteigender Relevanz für alle Knoten ausgeführt wird, die eine negative erste Komponente haben. Die dritte Komponente der Bewertungen wird wie zuvor einzeln pro Schleifenkomponente berechnet.

- Die zweite Komponente aller Profile wird, wie in 3.6.1 beschrieben, durch einen gemeinsamen Baum repräsentiert.

5.1.4 Diskussion von Fallstudien

Die ersten Fallstudien, die mit dem oben beschrieben System durchgeführt wurden, verwenden Beispiele, die aus der Literatur stammen, die die verschiedenen Ansätze zur Lösung von Paritätsspielen untersucht. Zwei dieser Artikel sind [BLV96], basierend auf dem rekursiven Algorithmus von McNaughton [McN93], und [Jur00b], in dem Fortschrittsmaße (progress measures) angewandt werden. In beiden Fällen lassen sich geeignete Familien von Spielgraphen definieren, die zu einer exponentiellen Laufzeit des jeweiligen Algorithmus führen. Die Analyse mit der vorliegenden Implementierung des Algorithmus zeigt, dass für diese Fälle eine lineare Anzahl von Iterationen ausreichend ist, was ein Polynomzeitverhalten für diese Fälle belegt. Wir beschränken uns im Folgenden auf die Darstellung des Beispiels aus [BLV96].

In [BLV96] wird ein Verfahren zur Erzeugung von Paritätsspielen beschrieben. Das dort beschriebene skalierbare Beispiel, für 16 Knoten dargestellt in Abbildung 5.3, soll im Folgenden untersucht werden.

Dieses skalierbare Spiel lässt sich in der Eingabesprache von `omega` wie folgt formulieren:

```
(defun gen-blv96 (gamesize)
  (when (evenp gamesize)
    (when (evenp (/ gamesize 2))
      (let* ((anzahl-spieler0-knoten (/ gamesize 2))
             ;; Anzahl der Kanten pro Knoten wird durch die Anzahl der
             ;; Funktionen fuer C- bzw. D-Knoten festgelegt
             (CFunctVector '#(c1 c2))
             (DFunctVector '#(d1 d2))
             (CFunct (length CFunctVector))
```

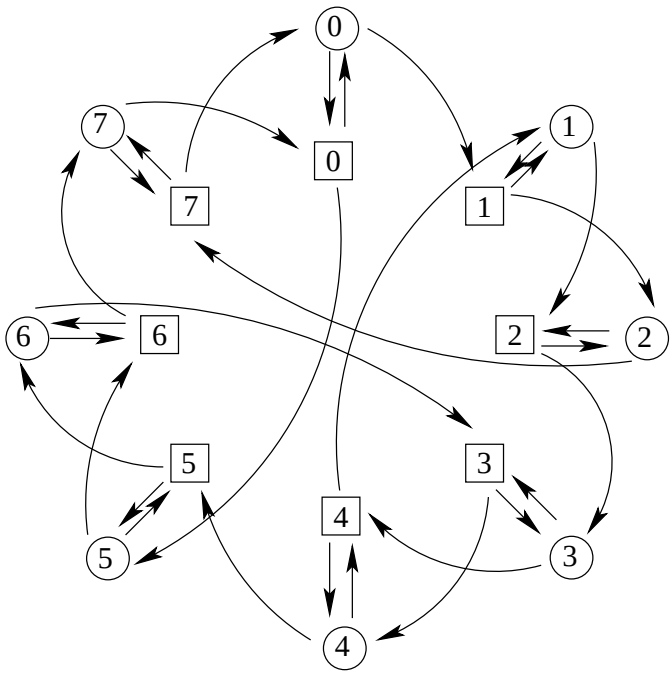


Abbildung 5.3: Beispielgraph aus [BLV96]

```
     (DFunct (length DFunctVector))
     (CSchrittMinus 1)
     (CSchrittPlus 1)
     (DSchrittMinus 1)
     (DSchrittPlus 1)
     (CSchritt (+ CSchrittMinus CSchrittPlus))
     (DSchritt (+ DSchrittMinus DSchrittPlus)))
(parity-game
 ;; Spieler 0-Knoten
 (for (v 0 (- anzahl-spieler0-knoten 1))
    (vertex ('v v) 0 (if (>= (mod v CSchritt) CSchrittMinus)
                         (+ (* 2 (floor (/ v CSchritt))) 1)
                       (* 2 (floor (/ v CSchritt))))))
  )
 ;; Spieler 1-Knoten
 (for (v 0 (- anzahl-spieler0-knoten 1))
    (vertex ('w v) 1 (if (>= (mod v DSchritt) DSchrittMinus)
                         (+ (* 2 (floor (/ v DSchritt))) 1)
                       (* 2 (floor (/ v DSchritt))))))
  )
 ;; Transitionen von Knoten von Spieler 0 aus
 ;; Um das Einfuegen mehrerer gleicher Kanten zu verhindern, wird
 ;; nach Berechnung eines neuen Zielknotens vor dem Kanten-Einfuegen
 ;; ueberprueft, ob die Kante schon existiert (in wertliste enthalten)
 (for (x 0 (- anzahl-spieler0-knoten 1))
   (let ((werteliste '()))
```

```
            (for (f 0 (- CFunct 1))
               (let ((wert (eval `(,(aref CFunctVector f)
                                   ,x ,anzahl-spieler0-knoten))))
                 (when (not (member wert werteliste))
                   (push wert werteliste)
                   (edge 'v x) 'w wert))))))
         )
         ;; Transitionen von Knoten von Spieler 1 aus
         ;; Verwendung wertliste analog zu oben
         (for (x 0 (- anzahl-spieler0-knoten 1))
           (let ((werteliste '()))
             (for (f 0 (- DFunct 1))
               (let ((wert (eval `(,(aref DFunctVector f)
                                   ,x ,anzahl-spieler0-knoten))))
                 (when (not (member wert werteliste))
                   (push wert werteliste)
                   (edge 'w x) 'v wert))))))
         ))
 ))))
```

Um einen nichttrivialen Lauf des Algorithmus zu erhalten, verwenden wir für die Analyse den auf 136 Knoten skalierten Spielgraphen, in welchem die Knoten von Spieler 0 die Nummern $0, 2, \ldots, 134$ tragen. Die Nummern der Knoten von Spieler 0 sind als Namen der Knoten in der berechneten Verbesserungsspur verwendet (Dezimalnotation, in Abb. 5.4 von oben nach unten geschrieben).

```
                                                                  11111111111111111
        1111122222333334444455555666667777788888999990000011111222223333
   02468024680246802468024680246802468024680246802468024680246802468024
--|------------------------------------------------------------------------
 1| * * * * * * * ** * * * * * * * * * * * * * * * * *  * * * * * * * * *
 2|* * * * * * * *  * * * * * * * ******************
 3|                               *                                      *
 4|              * * * * * * * *
 5|*           *                   *                                      *
 6|*         *
 7|        *
 8|      *
 9|    *
10|  *
11|*
```

Abbildung 5.4: Beispiel für eine Verbesserungsspur

Wie in Abbildung 5.4 dargestellt, liefert die Ausgabe Informationen über die Iterationen 1 bis 11. Aus der Verbesserungsspur ergibt sich, dass der Algorithmus bis zur 6. Iteration jeweils mehrere Strategien einzelner Knoten gleichzeitig korrigiert und dann nur noch einzelne Kantenauswahlen für die Knoten $16, 12, 8, 4, 0$

in den verbleibenden Iterationen ändert. In diesem Fall benötigt der Algorithmus 12 Iterationen, wobei die letzte nicht ausgegeben wird, da sie keine Änderungen in der Verbesserungsspur erbringen kann.

Die zusätzlichen Informationen für das oben untersuchte Beispiel sind in der Tabelle in Abbildung 5.5 gegeben. Aus dieser Tabelle ergibt sich, dass in den ersten beiden Iterationen für viele Knoten (je 34) die Strategie geändert wird und in den folgenden Iterationen nur noch für höchstens acht. Umgekehrt zeigt sich an der Anzahl der Summe der Kantentraversierungen (Spalte 7), dass der Aufwand pro Iteration tendenziell steigt. Insbesondere fällt die erste Iteration mit nur 713 Kantentraversierungen auf. Dieses Muster lässt sich auch bei anderen Beispielen erkennen.

In diesen Fällen scheint es in der ersten Iteration zu einem schnellen groben Einstellen auf die Bewertungsfunktion zu kommen, während danach die Strategie in kleiner werdenden Schritten mit immer größerem Aufwand verfeinert wird.

```
#vertices: 136 belonging to player 0: 68
#edges: 272 starting from vertices belonging to player 0: 136
#colors: 68
```

1	2	3	4	5	6	7
1	x	34	1	352	47	713
2	x	34	16	1200	164	1620
3	x	2	86	1186	422	1987
4	x	8	89	659	89	1139
5	x	4	156	1192	142	1853
6	x	2	158	1116	139	1778
7	x	1	160	1164	159	1850
8	x	1	162	1247	149	1927
9	x	1	164	1336	139	2010
10	x	1	166	1431	129	2099
11	x	1	168	1532	119	2195
12		0	0	0	0	171

Abbildung 5.5: Zusätzliche Informationen

5.2 μ-Kalkül-Model-Checking

Aufbauend auf der Implementierung der Strategiekonstruktion für Paritätsspiele ist eine Auswertung von μ-Kalkülformeln über beschrifteten Transitionssystemen möglich.

5.2.1 Eingabesprache

Ein Model-Checking-Problem wird in Form einer μ-Kalkülformel und eines kantenbeschrifteten Transitionssystems, auf welches sich die Formel bezieht, beschrieben.

Die Auswertung eines Model-Checking-Problems erfolgt durch den Aufruf:

`(eval-form graph formula)`,

wobei `graph` ein beschrifteter Transitionsgraph und `formula` eine μ-Kalkülformel ist, die jeweils in dem im Folgenden beschriebenen Format vorliegen. Wir gehen zunächst auf das Format für Formeln und danach auf das Format für Transitionssysteme ein.

μ-Kalkülformeln

Die Formeln des μ-Kalküls lassen sich in der Lisp-eigenen Präfixnotation angeben. Dabei sind alle Lisp-Symbole außer `nil` zulässige Variablennamen und alle Lisp-Symbole außer `nil` und `all` zulässige Beschriftungen:

x	`x`
$\Phi_1 \wedge \Phi_2 \wedge \ldots \Phi_n$	`(and` Φ_1 Φ_2 `...` Φ_2 `)`
$\Phi_1 \vee \Phi_2 \vee \ldots \Phi_n$	`(or` Φ_1 Φ_2 `...` Φ_2`)`
$[a]\,\Phi$	`(box a` Φ`)`
$\langle a \rangle\,\Phi$	`(diamond a` Φ`)`
$\mu x.\Phi$	`(mu x` Φ`)`
$\nu x.\Phi$	`(nu x` Φ`)`

Die Operatoren `and` und `or` haben beliebige Stelligkeit (insbesondere sind auch 0 Argumente zugelassen). Als erstes Argument von `box` und `diamond` ist die spezielle Konstante `all` zugelassen, die abkürzend für alle Beschriftungen steht.

Außerdem gibt es ein Dualisierungsmakro `dual`. Es liefert zu einer gegebenen Formel Φ die duale Formel, in der $\vee - \wedge$, $\langle a \rangle - [a]$ und $\mu - \nu$ vertauscht sind. Dualisierung und Negation sind gleich, wenn alle gebundenen Variablen nur positiv im Bezug auf den Kontext des jeweils bindenden Operators vorkommen. Somit kann das Dualisierungsmakro in diesen Fällen zur Negation genutzt werden.

Es stehen weitere Abkürzungen zur Verfügung, die es erlauben CTL-Operatoren zu verwenden:

```
ff          (or)
tt          (and)
(AF Φ)      (mu x (or (box all x) Φ))
(EF Φ)      (mu x (or (diamond all x) Φ))
(AG Φ)      (nu x (and (box all x) Φ))
(EG Φ)      (nu x (and (diamond all x) Φ))
(AU Φ Ψ)    (mu x (or (and (box all x) Φ) Ψ))
(EU Φ Ψ)    (mu x (or (and (diamond all x) Φ) Ψ))
```

Um Bindungsfehler zu vermeiden, sind die zusätzlichen durch diese Abkürzungen eingeführten Variablen immer so gewählt, dass ihre Namen verschieden von allen anderen in der Formel benutzten sind.

Kantenbeschriftete Transitionssysteme

Das beschriftete Transitionssystem, auf das sich die Formel bezieht, wird durch eine Inzidenzliste dargestellt:

$((s_1\ (a_{11}\ t_{111}\ t_{112}\ \ldots)\ (a_{12}\ t_{121}\ t_{122}\ \ldots))$
$\ (s_2\ (a_{21}\ t_{211}\ t_{212}\ \ldots)\ (a_{22}\ t_{221}\ t_{222}\ \ldots)))$

Dabei ist jeder Knoten t_{jkl} ein Zielknoten, der vom Quellknoten s_j mit der Beschriftung a_{jk} erreicht wird. Dabei muss $s_j \neq s_{j'}$ für $j \neq j'$, und auch $a_{jk} \neq a_{jk'}$ für $k \neq k'$ sein. Außerdem muss jeder Knoten des Systems als Quellknoten aufgeführt werden (auch dann, wenn er keine ausgehenden Kanten hat).

Beispiel: Der Transitionsgraph $(\{1,2,3,4\},\ \{(1,a,2),\ (1,a,3),\ (1,b,4),\ (2,a,1),\ (2,b,3),\ (3,c,3)\})$, dargestellt in Abbildung 5.6, hat somit folgende Darstellung:

```
((1 (a 2 3) (b 4)) (2 (a 1) (b 3)) (3 (c 3)) (4)).
```

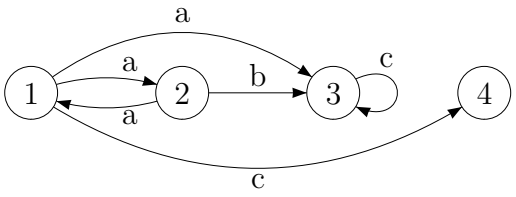


Abbildung 5.6: Spielgraph G_1.

5.2.2 Ausgabe

Das Ergebnis der Auswertung eines Model-Checking-Problems ist die Liste der Knoten des gegebenen Transitionssystems `graph`, in welchen die gegebene μ-Kalkülformel `formula` gilt. Der Aufruf

```
(eval-form graph formula)
```

liefert diese Liste.

Beispiel 1

Die Formel

$$\mu x.\langle a\rangle\, x \vee \langle c\rangle\, \mathrm{tt}$$

gilt in allen Knoten, von denen aus ein mit Buchstaben a beschrifteter Kantenzug zu einem Knoten führt, der eine mit c beschriftete ausgehende Kante besitzt.

Ihre Auswertung auf dem Graph aus Abbildung 5.6 wird durch

```
(eval-form '((1 (a 2 3) (b 4)) (2 (a 1) (b 3)) (3 (c 3)) (4))
           '(mu x (or (diamond a x) (diamond c tt))))
```

formuliert.

Die Ausgabe ist: `(1 2 3)`. Die Formel kann in Knoten 4 nicht gelten, weil sie mindestens eine mit a oder c beschriftete ausgehende Kante fordert. Sie gilt in Knoten 3, weil dieser eine ausgehende Kante mit c-Beschriftung besitzt; d.h. $\langle c\rangle\, \mathrm{tt}$ gilt. Und die Formel gilt in Knoten 1 und 2, weil von diesen Knoten jeweils ein mit as beschrifteter Kantenzug zu 3 führt.

Beispiel 2

Das Alternating-Bit-Protokoll ([Mil89, LT98]) ist ein einfaches Netzwerkprotokoll, das die Übertragung von Nachrichten über einen unsicheren Kanal modelliert. Bei diesem Protokoll kann die Nachricht oder deren Ankunftsbestätigung bei der Übertragung verloren gehen, was zum erneuten Versenden der Nachricht führt. Wir betrachten drei Transitionssysteme die aus einem einzelnen Kanal, zwei und drei hintereinander geschalteten Kanälen bestehen. In Abbildung 5.7 ist das Transitionssystem nach einer Formulierung aus [Tob98] mit Hilfe von CCS-Termen gegeben. Für die Transitionssysteme ergeben sich folgende Anzahlen für Zustände und Transitionen:

Kanäle	1	2	3
Zustände	57	1589	44431
Transitionen	130	6819	280456

Kantenbeschriftungen sind die Aktionen *send* und *receive* und die stille Aktion τ, die nicht nach außen sichtbare Aktionen modelliert. Folgende Eigenschaften dieses Systems wurden (wie auch in [LT98]) untersucht:[2]

[2] Es werden die Abkürzungen $[-]\,\varphi = \bigwedge_{a\in L} [a]\,\varphi$ und $\langle -\rangle\,\varphi = \bigvee_{a\in L} \langle a\rangle\,\varphi$ verwendet, wobei L die Menge der Beschriftungen des Transitionssystems ist.

$$
\begin{array}{lcl}
S_0 & = & send.S_0' \\
S_0' & = & \overline{s_0}.(rack_0.S_1 + rack_1.S_0' + \tau.S_0') \\
S_1 & = & send.S_1' \\
S_1' & = & \overline{s_1}.(rack_1.S_0 + rack_0.S_1' + \tau.S_1') \\
R_0 & = & r_0.\overline{receive}.\overline{sack_0}.R_1 + r_1.\overline{sack_1}.R_0 + \tau.\overline{sack_1}.R_0 \\
R_1 & = & r_1.\overline{receive}.\overline{sack_1}.R_0 + r_-.\overline{sack_0}.R_1 + \tau.\overline{sack_0}.R_1 \\
C & = & s_0.(\overline{r_0}.C + C) + s_1.(\overline{r_1}.C + C) \\
 & & \quad + sack_0.(\overline{rack_0}.C + C) + sack1.(\overline{rack_1}.C + C) \\
ABP & = & (R_0 \mid C \mid S_0) \setminus \{r_0, r_1, s_0, s_1, rack_0, rack_1, sack_0, sack_1\} \\
ABP_2 & = & (ABP[inode/receive] \mid ABP[inode/send]) \setminus \{inode\} \\
ABP_3 & = & (ABP[inode_1/receive] \mid ABP[inode_1/send, inode_2/receive] \\
 & & \quad \mid ABP[inode_2/send]) \setminus \{inode_1, inode_2\}
\end{array}
$$

Abbildung 5.7: Das Alternating Bit Protokoll durch CCS-Terme formuliert.

$$
\begin{array}{lcl}
\varphi_1 & = & \mathrm{EF}\,[-]\,\mathrm{ff} \\
 & & \text{(Es existiert ein Deadlock.)} \\
\varphi_2 & = & \mu x.(\langle\overline{receive}\rangle\,\mathrm{tt} \vee \langle\tau\rangle\,x) \\
 & & \text{(Es kann empfangen werden.)} \\
\varphi_2' & = & \mu x.(\langle send\rangle\,\mathrm{tt} \vee \langle\tau\rangle\,x) \\
 & & \text{(Es kann gesendet werden.)} \\
\varphi_3 & = & \mathrm{AG}\,(\varphi_2 \vee \varphi_2') \\
 & & \text{(Es kann immer wieder gesendet oder empfangen werden.)} \\
\varphi_4 & = & \mathrm{AG}\,([send]\,\neg\varphi_2' \vee [\overline{receive}]\,\neg\varphi_2) \\
 & & \text{(Es kann nicht mehrfach gesendet oder mehrfach empfangen werden.)} \\
\varphi_5 & = & \mathrm{AG}\ \big([send]\,\nu x.(\langle\overline{receive}\rangle\,\mathrm{tt} \vee (\langle send\rangle\,\mathrm{ff} \wedge [-]\,x)) \\
 & & \qquad \wedge [\overline{receive}]\,\nu x.(\langle send\rangle\,\mathrm{tt} \vee (\langle\overline{receive}\rangle\,\mathrm{ff} \wedge [-]\,x))\big) \\
 & & \text{(Nach jedem Senden kann nur empfangen werden und umgekehrt.)} \\
\varphi_6 & = & \nu y.\mu x.[-]\,((\langle send\rangle\,\mathrm{tt} \wedge y) \vee x) \\
 & & \text{(Es kann unendlich oft gesendet werden.)}
\end{array}
$$

Die folgende Tabelle zeigt für die verschiedenen Formeln (1. Spalte) die Auswertungszeiten des Model-Checking-Problems (2. Hauptspalte), der dazu berechnenden Strategiekonstruktion (3. Hauptspalte) und die Anzahl der dafür benötigten Iterationen (4. Hauptspalte). Abhängig von der Formel erreichen die konstruierten Paritätsspiele eine Größe von bis zu 800 000 Knoten. Auffallend sind demgegenüber die geringen Anzahlen von Iterationen.

	Gesamt (sec)			dSVA (sec)			Iterationen		
Kanäle	1	2	3	1	2	3	1	2	3
φ_1	0.05	46.05	35 595.	0.01	1.48	469.	4	21	26
φ_2	0.05	45.61	35 562.	0.01	0.82	456.	4	8	13
φ_2'	0.06	45.12	35 531.	0.01	0.96	437.	4	16	14
φ_3	0.18	272.40	225 868.	0.01	10.69	23 393.	5	16	14
φ_4	0.30	140.21	122 163.	0.01	3.65	4 903.	1	2	2
φ_5	0.31	351.43	302 442.	0.02	5.81	7 395.	1	1	1
φ_6	0.07	55.57	55 604.	0.01	4.57	11 071.	7	2	2

Im Vergleich zu vorhandenen Model-Checking-Systemen wie CWB [CPS93], NCSU-CWB [CS96] und TRUTH [LT98, Tob98] ist zweierlei festzustellen: Die Laufzeiten des dSV-Algorithmus liegen deutlich über den durch die anderen Systeme erzielten Werten. Jedoch zeigt sich (etwa bei φ_5), dass der Aufwand innerhalb einer Iteration die Laufzeit deutlich mehr bestimmt als die Anzahl der Iterationen. Bei dem im Falle des dSV-Algorithmus für die theoretische Analyse kritischen Parameter (Anzahl der Iterationen) ergibt sich kein Anstieg, der ein exponentielles Verhalten nahelegt. Ob diese experimentelle Beobachtung eine abstrakte Eigenschaft des Algorithmus wiedergibt, muss hier offen bleiben.

5.2.3 Optimierungen der Implementierung

Bei der Konstruktion des Spiels, wie in Abschnitt 4.2 beschrieben, sind verschiedene Optimierungen möglich. Im Formel-Transitionssystem lassen sich Zustände einsparen:

- Variablenknoten haben immer genau einen Nachfolger (eine μ- oder ν-Formel). Transitionen zu ihnen können durch Transitionen zu ihren Nachfolgern ersetzt werden.

- μ- oder ν-Knoten können mit ihren Nachfolgern zusammengefasst werden, wenn diese nicht auch μ- oder ν-Knoten sind. Der zusammengefasste Knoten gehört dem Spieler, dem auch der Nachfolger des μ- oder ν-Knotens gehört. Er besitzt jedoch die Farbe des μ- oder ν-Knotens.

- $\wedge$-Knoten können mit ihren Nachfolgern zusammengefasst werden, wenn diese $[]$- oder auch $\wedge$-Knoten sind.

- $\vee$-Knoten können mit ihren Nachfolgern zusammengefasst werden, wenn diese $\langle\rangle$- oder auch $\vee$-Knoten sind.

Weiterhin muss nur der erreichbare Teil des Produktes aus Transitions- und Formelgraph konstruiert werden. Die Implementierung berücksichtigt diese Optimierungen.

Die entwickelte Implementierung kann zum Model-Checking für μ-Kalkül-Spezifikationen benutzt werden, ist jedoch nicht hierfür optimiert. Ist ein Model-Checking-Problem in ein Paritätsspiel transformiert, so muss für dieses Spiel lediglich zu einem Zustand oder zu wenigen Zuständen (die, welche die ganze Formel enthalten) ermittelt werden, welcher Spieler von diesen Zuständen aus gewinnt. Ein optimierter Algorithmus würde somit Strategien nicht auf dem ganzen Spielgraphen berechnen und den Spielgraph und Bewertungen dafür nur soweit wie nötig generieren.

Kapitel 6

Zusammenfassung und Ausblick

In der vorliegenden Arbeit wurde ein neuer Algorithmus, der diskrete Strategieverbesserungsalgorithmus (dSVA), zur Synthese von Gewinnstrategien für Paritätsspiele entwickelt.

Dazu wurde eine Bewertung von Partien eingeführt, die eine erhebliche Verfeinerung der Paritätsbedingung darstellt, während die Paritätsbedingung nur die Werte ‚Gewinnpartie' und ‚Verlustpartie' unterscheidet. Schon Puri ([Pur95]) lieferte mit seiner Transformation von Paritätsspielen auf Discounted-Payoff-Spiele und der dazu definierten Bewertung einer Partie eine Verfeinerung der Paritätsbedingung. Seine verfeinerte Bewertung enthält jedoch einerseits für Paritätsspiele unwesentliche Informationen, und sie bewertet andererseits jede Partie mit einer reellen Zahl, was das Extrahieren der darin enthaltenen unterschiedlichen diskreten Informationen nahezu unmöglich macht. In Kapitel 3 wurden drei wesentliche Parameter einer Partie in Paritätsspielen identifiziert, die als Komponenten eines Tripels Eingang in die definierte Bewertungsfunktion finden:

1. Relevantester unendlich oft auftretender Knoten v.

2. Menge der Knoten der Partie, die relevanter als v sind.

3. Anzahl der ‚Züge' bis zum ersten Auftreten von v.

An diesen Parametern orientiert sich der vorgestellte diskrete Strategieverbesserungsalgorithmus. Sie haben den Vorteil, sich leicht aus einer vorliegenden Partie ermitteln zu lassen.

Es wurde der Zusammenhang dieser Bewertung zu der von Puri verwendeten hergestellt. Dazu wurde eine approximierte Bewertung eingeführt, die es erlaubt, Gewinnstrategien für Discounted-Payoff-Spiele mit großem Discountfaktor zu berechnen, was insbesondere für Mean-Payoff-Spiele interessant ist, die sich als Discounted-Payoff-Spiele mit Discountfaktor $1-\epsilon$ mit verschwindend kleinem ϵ verstehen lassen.

Die Implementierung des dSV-Algorithmus ermöglicht die empirische Analyse des Algorithmus. Die implementierte Transformation von Model-Checking-Problemen in Paritätsspiele macht die Klasse von bereits in diesem Kalkül beschriebenen Problemen zugänglich für die Analyse des Algorithmus.

Die wichtigste Frage, die in dieser Arbeit offenbleibt, ist die der genauen Laufzeitbestimmung des dSV-Algorithmus im schlechtesten Fall. Es ist nicht gelungen, die Lücke zwischen der (trivialen) exponentiellen Schranke, gegeben durch die Menge der möglichen Strategien über dem betrachteten Graphen, und einer linearen unteren Schranke zu schließen.

Dennoch ist der Algorithmus von methodischem Interesse: Das Problem, ob die Lösung von Paritätsspielen in Polynomzeit möglich ist, ist nun möglicherweise auf die Frage reduziert, ob der konkrete dSV-Algorithmus eine Polynomzeit-Abschätzung zulässt.

Eine Weiterentwicklung der hier erzielten Ergebnisse könnte darin bestehen, dass die Bewertungsfunktion verfeinert wird, um so eine Polynomzeit-Abschätzung zu erreichen. Eine solche verfeinerte Bewertungsfunktion sollte auch einen wichtigen Parameter berücksichtigen, der im vorliegenden Ansatz nicht zum Zuge kommt, nämlich die Anzahl der Farben (Prioritäten) des Paritätsspiels, die ja in engem Zusammenhang zur Alternationstiefe von Formeln des μ-Kalküls steht.

Ein Problem mit weiterer Perspektive ist die Frage, ob der dSV-Algorithmus sich über kompositional definierten Spielgraphen adäquat anwenden lässt. Es wäre von Interesse, Strategien über hierarchisch oder kompositionell präsentierten Spielgraphen unter Berücksichtigung der hierarchischen bzw. kompositionellen Struktur zu konstruieren.

Schließlich sei erwähnt, dass eine Verallgemeinerung des dSV-Algorithmus auch auf gewisse unendliche Graphen möglich erscheint. Insbesondere ist hier an Paritätsspiele über Pushdown-Transitionsgraphen zu denken, wie sie von Seibert [Sei96a] und Walukiewicz [Wal96] betrachtet wurden.

Literaturverzeichnis

[AN90] Arnold, Andre und Damian Niwinski: *Fixed point characterization of Büchi automata on infinite trees*. Journal of Inf. Process. Cybern., 1990.

[BCDM86] Brown, Michael C., Edmund M. Clarke, David L. Dill und Bud Mishra: *Automatic Verification of Sequential Circuits Using Temporal Logic*. IEEE Transactions on Computers, 35(12):1035–1044, 1986.

[BCJ+97] Browne, A., E. M. Clarke, S. Jha, D. E. Long und W. Marrero: *An improved algorithm for the evaluation of fixpoint expressions*. Theoretical Computer Science, 178(1–2):237–255, Mai 1997.

[BL69] Büchi, J. Richard und Lawrence H. Landweber: *Solving Sequential Conditions by Finite-State Strategies*. Trans. Amer. Math. Soc., 138:295–311, 1969.

[BLV96] Buhrke, Nils, Helmut Lescow und Jens Vöge: *Strategy construction in infinite games with Streett and Rabin chain winning conditions*. In: Magaria, Tiziana und Bernhard Steffen (Herausgeber): *TACAS'96*, Band 1055 der Reihe *Lecture Notes in Computer Science*, Seiten 207–225. Springer-Verlag, 1996.

[Bra92] Bradfield, Julian C.: *Verifying Temporal Properties of Systems*. In: *Progress in Theoretical Computer Science*. Birkhäuser, Boston, Basel, Berlin, 1992.

[Bra96] Bradfield, Julian C.: *The modal μ-calculus alternation hierarchy is strict*. In: Montanari, U. und V. Sassone (Herausgeber): *Conference on Concurrency Theory*, Band 1119 der Reihe *Lecture Notes in Computer Science*, Seiten 233–246. Springer-Verlag, 1996.

[Bra98] Bradfield, Julian C.: *Simplifying the modal μ-calculus alternation hierarchy*. In: Morvan, M., C. Meinel und D. Krob (Herausgeber): *STACS'98*, Band 1376 der Reihe *Lecture Notes in Computer Science*, Seiten 39–49. Springer-Verlag, 1998.

[BS92] Bradfield, Julian C. und Colin Stirling: *Verifying temporal properties of processes*. Theoretical Computer Science, 96:157–174, 1992.

[Büc60] Büchi, J. Richard: *On a decision method in restricted second order arithmetic*. In: Nagel, E. (Herausgeber): *International Congress on Logic, Methodology and Philosophy of Science*, Seiten 1–12, Stanford, CA, 1960. Stanford University Press.

[Büc62] Büchi, J. Richard: *On a Decision Method in Restricted Second Order Arithmetic*. In: *Logic, Methodology and Philosophy of Science. Proceedings 1960 Intern. Congr.*, Seiten 1–11. Stanford Univ. Press, 1962.

[Büc83] Büchi, J. Richard: *State Strategies for Games in $F_{\sigma\delta} \cap G_{\delta\sigma}$*. In: Karpinski, M. (Herausgeber): *Fundamentals in Computation Theory*, Band 48 der Reihe *Lecture Notes in Computer Science*, Seiten 1171–1198, Berlin, Heidelberg, New York, 1983. Springer-Verlag.

[CE81] Clarke, Edmund M. und E. Allen Emerson: *Design and Synthesis of Synchronization Skeletons using Branching Time Temporal Logic*. In: Kozen, Dexter (Herausgeber): *Workshop on Logics of Programs*, Band 131 der Reihe *Lecture Notes in Computer Science*, Seiten 52–71, Yorktown Heights, New York, Mai 1981. Springer-Verlag.

[CES86] Clarke, Edmund M., E. Allen Emerson und A. Prasad Sistla: *Automatic verification of finite-state concurrent systems using temporal logic specifications*. In: *ACM Transactions on Programming Languages and Systems*, Band 8, Seiten 244–263. Springer-Verlag, 1986.

[Chu63] Church, Alonzo: *Logic, arithmetic and automata*. In: Almqvist und Wiksells (Herausgeber): *Proc. Internat. Congress Math.*, Seiten 23–35, Uppsala, 1963.

[CKS81] Chandra, Ashok K., Dexter Kozen und Larry J. Stockmeyer: *Alternation*. Journal of ACM, 28:114–133, 1981.

[CKS92] Cleaveland, Rance, M. Klein und Bernhard Steffen: *Faster Model Checking in the Modal μ-Calculus*. In: G.v. Bochmann, D.K. Probst (Herausgeber): *Computer Aided Verification (CAV'92)*, Band 663 der Reihe *Lecture Notes in Computer Science*, Seiten 410–422, Heidelberg, Germany, 1992. Springer-Verlag.

[Cle89] Cleaveland, Rance: *Tableau-Based Model Checking in the Propositional μ-Calculus*. Acta Informatica, 27(8):725–747, 1989.

[Con92] Condon, Anne: *The Complexity of Stochastic Games*. Information and Computation, 96:203–224, 1992.

[Con93] Condon, Anne: *On Algorithms for Simple Stochastic Games*. In: Cai, Jin-Yi (Herausgeber): *Advances in Computational Complexity Theory*, Band 13 der Reihe *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, Seiten 51–73. American Mathematical Society, 1993.

[CPS93] Cleaveland, Rance, Joachim Parrow und Bernhard Steffen: *The Concurrency Workbench: A Semantics-Based Tool for the Verification of Concurrent Systems*. ACM Transactions on Programming Languages and Systems, 15(1):36–72, jan 1993.

[CS91] Cleaveland, Rance und Bernhard Steffen: *A Linear-Time Model-Checking Algorithm for the Alternation-Free Modal μ-Calculus*. In: *CAV'91, Aalborg (Denmark)*, Band 526 der Reihe *Lecture Notes in Computer Science*, Seiten 186–196, Heidelberg, Germany, July 1991. Springer-Verlag.

[CS93] Cleaveland, Rance und Bernhard Steffen: *A Linear-Time Model-Checking Algorithm for the Alternation-Free Modal μ-Calculus*. International Journal on Formal Methods in System Design, 1(1):121–147, 1993.

[CS96] Cleaveland, Rance und Steve Sims: *The NCSU Concurrency Workbench*. In: Alur, Rajeev und Thomas A. Henzinger (Herausgeber): *Computer Aided Verification*, Band 1102 der Reihe *Lecture Notes in Computer Science*, Seiten 394–397, New Brunswick, NJ, USA, Juli 1996. Springer-Verlag.

[EC82] Emerson, E. Allen und Edmund M. Clarke: *Using Branching Time Temporal Logic to Synthesize Synchronization Skeletons*. Science of Computer Programming, 3(2):241–266, 1982.

[EJ88] Emerson, E. Allen und Charanjit S. Jutla: *The Complexity of Tree Automata and Logics of Programs (Extended Abstract)*. In: *FOCS 1988*, Seiten 328–337, 1988.

[EJ89] Emerson, E. Allen und Charanjit S. Jutla: *On Simultaneously Determinizing and Complementing omega-Automata (Extended Abstract)*. In: *IEEE Symposion on Logic in Computer Science*, Seiten 333–342, 1989.

[EJ91] Emerson, E. Allen und Charanjit S. Jutla: *Tree Automata, μ-Calculus and Determinacy*. In: *Proc. 32nd IEEE Symposium on the Foundations of Computing*, Seiten 368–377, 1991.

[EJS93] Emerson, E. Allen, Charanjit S. Jutla und A. Prasad Sistla: *On Model-Checking for Fragments of μ-Calculus*. In: Courcoubetis, Costas (Herausgeber): *Computer Aided Verification*, Band 697 der Reihe *Lecture Notes in Computer Science*, Seiten 385–396, Elounda, Greece, 1993. Springer-Verlag.

[EL86] Emerson, E. Allen und C.L. Lei: *Efficient Model Checking in Fragments of the Propositional μ-Calculus*. In: *Logic in Computer Science (LICS)*, Seiten 267–278, Los Alamitos, CA, 1986. IEEE.

[EM79] Ehrenfeucht, Andrzej und Jan Mycielski: *Positional Strategies for Mean Payoff Games*. Int. Journal of Game Theory, 8(2):109–113, 1979.

[Eme96] Emerson, E. Allen: *Model Checking and the μ-calculus*. In: Immerman, Neil und Phokion G. Kolaitis (Herausgeber): *Descriptive Complexity and Finite Models*, Band 31 der Reihe *Discrete Mathematics and Computer Science*, Seiten 185–214. Princeton University, American Mathematical Society, Januar 1996.

[GS53] Gale, D. und F. M. Stewart: *Infinite games with perfect information*. Annals of Mathematical Studies, 28:245–266, 1953.

[HK66] Hoffman, A. J. und Richard M. Karp: *On Non-terminating Stochastic Games*. Management Science, 12:359–370, 1966.

[How60] Howard, Ronald A.: *Dynamic Programming and Markov Processes*. MIT Press, 1960.

[Jur98] Jurdziński, Marcin: *Deciding the winner in parity games is in UP $\cap$ co-UP*. Information Processing Letters, 68:119–124, 1998.

[Jur00a] Jurdziński, Marcin: *Games for Verification: Algorithms and Decidability Issues*. Doktorarbeit, BRICS International PhD School, Department of Computer Science, University of Aarhus, 2000.

[Jur00b] Jurdziński, Marcin: *Small Progress Measures for Solving Parity Games*. In: Reichel, Horst und Sophie Tison (Herausgeber): *STACS 2000, 17th Annual Symposium on Theoretical Aspects of Computer Science, Proceedings*, Band 1770 der Reihe *Lecture Notes in Computer Science*, Seiten 290–301, Lille, France, February 2000. Springer-Verlag.

[Kec94] Kechris, Alexander S.: *Classical Descriptive Set Theory.* Springer-Verlag, 1994.

[KK91] Klarlund, Nils und Dexter Kozen: *Rabin measures and their applications to fairness and automata theory.* In: *Proceedings, Sixth Annual IEEE Symposion on Logic in Computer Science*, Seiten 256–265, Amsterdam, The Netherlands, Juli 1991. IEEE Computer Society Press.

[Koz83] Kozen, Dexter: *Results on the propositional μ-calculus.* Theoretical Computer Science, 27:333–354, 1983.

[KV86] Kumar, P. R. und P. Varaiya: *Stochastic Systems: Estimation, Identification and Adaptive control.* Prentice Hall, 1986.

[LT98] Leucker, Martin und Stephan Tobies: *Truth—A Platform for Verification of Distributed Systems.* Technischer Bericht 98-05, RW-TH Aachen, Mai 1998.

[Mad97] Mader, Angelika: *Verification of Modal Properties Using Boolean Equation Systems.* Doktorarbeit, Fakultät für Informatik, Technische Universität München, 1997.

[Mar75] Martin, Donald A.: *Borel Determinacy.* Annals of Mathematics, 102:363–371, 1975.

[MC94] Melekopoglou, Mary und Anne Condon: *On the Complexity of the Policy Improvement Algorithm for Stochastic Games.* ORSA (Op. Res. Soc. of America) Journal of Computing, 6(2):188–192, 1994.

[McN66] McNaughton, R.: *Testing and generating infinite sequences by a finite automaton.* Information and Control, 9:521–530, 1966.

[McN93] McNaughton, Robert: *Infinite Games Played on finite Graphs.* Ann. Pure Appl Logic, 65:149–184, 1993.

[Meg83] Megiddo, Nimrod: *Towards a genuinely polynomial algorithm for linear programming.* SIAM Journal of Computing, 12(2):347–353, Mai 1983.

[Mil89] Milner, R.: *Communication and Concurrency.* International Series in Computer Science. Prentice Hall, 1989.

[Mos80] Moschovakis, Yiannis N.: *Descriptive Set Theory.* North Holland, Amsterdam, 1980.

[Mos84] Mostowski, Andrzej Wlodzimierz: *Regular expressions for infinite trees and a standard form of automata*. In: Skowron, A. (Herausgeber): *Computation Theory*, Band 208 der Reihe *Lecture Notes in Computer Science*, Seiten 157–168, Berlin, 1984. Springer-Verlag.

[MP92] Manna, Zohar und Amir Pnueli: *The Temporal Logic of Reactive and Concurrent Systems*. Springer-Verlag, Berlin, Heidelberg, New York, 1992.

[Mul63] Muller, D. E.: *Infinite sequences on finite machines*. In: *Proc. 4th IEEE Symposium on Switching Circuit Theory and Logical Design*, Seiten 3–16, 1963.

[MW84] Manna, Zohar und Pierre Wolper: *Synthesis of Communicating Processes from Temporal Logic Specifications*. ACM Transactions on Programming Languages and Systems, 6:68–93, 1984.

[Niw86] Niwinski, Damian: *On Fixed-Point Clones*. In: Kott, L. (Herausgeber): *ICALP*, Band 226 der Reihe *Lecture Notes in Computer Science*, Seiten 464–473, Berlin, 1986. Springer-Verlag.

[Niw88] Niwinski, Damian: *Fixed Points vs. Infinite Generation*. In: *Proc. 3rd Ann. IEEE Symposium on Logic in Computer Science*, Seiten 402–409, Washington, DC, 1988. IEEE Computer Society Press.

[Pur95] Puri, Anuj: *Theory of hybrid systems and discrete event systems*. Ph.D. Thesis, Electronics Research Laboratory, College of Engineering, University of California, Berkeley, CA 94720, Dezember 1995. Memorandum No. UCB/ERL M95/113.

[Rab69] Rabin, Michael O.: *Decidability of Second-Order Theories and Automata on Infinite Trees*. Transactions of the American Mathematical Society, 141:1–35, 1969.

[Rab72] Rabin, Michael O.: *Automata on Infinite Objects and Church's Problem*. In: *Regional Conference Series in Mathematics*, Band 13, Providence, Rhode Island, 1972. American Mathematical Society.

[Ros83] Ross, S. M.: *Introduction to Stochastic Dynamic Programming*. Academic Press, 1983.

[SE89] Streett, Robert S. und E. Allen Emerson: *An Automata Theoretic Decision Procedure for the Propositional μ-Calculus*. Information and Computation, 81(3):249–264, 1989.

[Sei96a] Seibert, S.: *Effektive Strategiekonstruktion für Gale-Stewart-Spiele auf Transitionsgraphen*. Doktorarbeit, Christian-Albrechts-Universität zu Kiel, 1996.

[Sei96b] Seidl, Helmut: *Fast and Simple Nested Fixpoints*. Information Processing Letters, 59(6):303–308, September 1996.

[Sha53] Shapley, Lloyd Stowell: *Stochastic Games*. In: *Proceedings of the National Academy of Science*, Band 39, Seiten 1095–1100, 1953.

[SS98] Stevens, Perdita und Colin Stirling: *Practical Model-Checking Using Games*. In: Steffen, Bernhard (Herausgeber): *Tools and Algorithms for Construction and Analysis of Systems*, Band 1384 der Reihe *Lecture Notes in Computer Science*, Seiten 85–101, Lisabon, Portugal, March 1998. Springer-Verlag.

[Sti95] Stirling, Colin: *Local Model Checking Games (Extended Abstract)*. In: Lee, Insup und Scott A. Smolka (Herausgeber): *CONCUR'95: Concurrency Theory, 6th International Conference*, Band 962 der Reihe *Lecture Notes in Computer Science*, Seiten 1–11. Springer-Verlag, 1995.

[Str82] Streett, R.S.: *Propositional dynamic logic of looping and converse*. Information and Control, 54:121–141, 1982.

[SV00a] Schmitz, Dominik und Jens Vöge: *Implementation of a strategy improvement algorithm for parity games*. In: *CIAA*, Lecture Notes in Computer Science. Springer-Verlag, 2000.

[SV00b] Schmitz, Dominik und Jens Vöge: *The Package ω, Reference-Manual*. RWTH Aachen, 2000.

[SW89] Stirling, Colin und David Walker: *Local Model Checking in the Modal μ-Calculus*. In: Díaz, Josep und Fernando Orejas (Herausgeber): *International Joint Conference on Theory and Practice of Software Development*, Band 351 der Reihe *Lecture Notes in Computer Science*, Seiten 369–383, Barcelona, Spain, März 1989. Springer-Verlag.

[SW91] Stirling, Colin und D. Walker: *Local model checking in the modal μ-calculus*. Theoretical Computer Science, 89:161–177, 1991.

[Tar55] Tarski, Alfred: *A Lattice-Theoretical Fixpoint Theorem and Its Applications*. Pacific Journal of Mathematics, 1955.

[TB73] Trakhtenbrot, Boris A. und Ya. M. Barzdin: *Finite Automata: Behavior and Synthesis*. North Holland, Amsterdam, 1973.

[Tho90] Thomas, Wolfgang: *Automata on infinite objects.* In: Leeuwen, Jan van (Herausgeber): *Handbook of Theoretical Computer Science*, Band B, Seiten 135–191. Elsevier, Amsterdam, 1990.

[Tho95] Thomas, Wolfgang: *On the synthesis of strategies in infinite games.* In: *STACS 95*, Band 900 der Reihe *Lecture Notes in Computer Science*, Seiten 1–13. Springer-Verlag, 1995.

[Tho97] Thomas, Wolfgang: *Languages, Automata, and Logic.* In: Rozenberg, Grzegorz und Arto Salomaa (Herausgeber): *Handbook of Formal Language Theory*, Band III, Seiten 389–455. Springer-Verlag, New York, 1997.

[Tho00] Thomas, Wolfgang: *Automaten und Reaktive Systeme.* RWTH Aachen, 2000. Vorlesungsskript.

[Tob98] Tobies, Stephan: *Design und Implementierung einer Plattform zur Verifikation verteilter Systeme.* Diplomarbeit, Mathematisch-Naturwissenschaftliche Fakultät, RWTH Aachen, Februar 1998.

[VJ00] Vöge, Jens und Marcin Jurdziński: *A Discrete Strategy Improvement Algorithm for Solving Parity Games.* In: Emerson, E. Allen und A. Prasad Sistla (Herausgeber): *Computer Aided Verification (CAV)*, Lecture Notes in Computer Science. Springer-Verlag, 2000.

[Vor00] Vorobyov, Sergei: *Randomized Strategy Improvement Algorithm for Parity Games.* In: *IEEE Symposium on Logic in Computer Science*, Santa Barbara, California, Juni 2000. IEEE Computer Society Press.

[Wal96] Walukiewicz, Igor: *Pushdown Processes: Games and Model Checking.* In: *Computer Aided Verification*, Band 1102 der Reihe *Lecture Notes in Computer Science*, Seiten 62–74. Springer-Verlag, Juli 1996.

[Zie98] Zielonka, Wiesław: *Infinite games on finitely coloured graphs with applications to automata on infinite trees.* Theoretical Computer Science, 200:135–183, 1998.

[ZP96] Zwick, Uri und Michael S. Paterson: *The complexity of mean payoff games on graphs.* Theoretical Computer Science, 158:343–359, 1996.

Bildungsgang

Name:	Jens Vöge
Geboren am:	15. Juni 1967
in:	Schleswig

08/1974 – 07/1978	Grundschule (Flensburg/Kappeln)
08/1978 – 07/1980	Gymnasium (Klaus-Harms-Schule Kappeln)
08/1980 – 05/1987	Gymnasium (Domschule Schleswig)
05/1987	Abitur
10/1988 – 04/1995	Studium der Informatik an der Christian–Albrechts–Universität zu Kiel
04/1995	Diplom in Informatik (Nebenfach: Mathematik)
05/1995 – 08/1998	Wissenschaftlicher Mitarbeiter am Lehrstuhl für Theoretische Informatik der Christian-Albrechts-Universität zu Kiel
09/1998 – 02/2001	Wissenschaftlicher Mitarbeiter am Lehrstuhl für Informatik VII (Logik und Theorie diskreter Systeme) der Rheinisch-Westfälischen Technischen Hochschule Aachen