Konstruktion
von Isomorphieklassen
orientierter Matroide

Ralf Gugisch

ADRESSE DES AUTORS:
Ralf Gugisch
Leuschnerstr. $1\frac{1}{2}$
95447 Bayreuth
e-mail: ralf.gugisch@uni-bayreuth.de

Diese Arbeit wurde von der Fakultät für Mathematik und Physik der Universität Bayreuth als Dissertation zur Erlangung des Grades eines Doktors der Naturwissenschaften genehmigt.

D 703

1. Gutachter:	Prof. Dr. Adalbert Kerber
2. Gutachter:	Prof. Dr. Reinhard Laue
Tag der Einreichung:	14. 06. 2005
Tag des Kolloquiums:	21. 07. 2005

Vorwort

Orientierte Matroide haben neben ihrer Bedeutung innerhalb der Mathematik, z.B. im Rahmen der Polytop- und Zonotop-Theorie, auch Anwendung bei der vollständigen Konformationsanalyse chemischer Substanzen gefunden: Erstmals erkannten A. Dress und A. Dreiding die Möglichkeit, Konformationen eines Moleküls durch affine Punktkonfigurationen der Atome im $\mathbb{R}^3$ zu beschreiben, also durch realisierbare azyklische orientierte Matroide vom Rang 4 ([DW80, DDH82], siehe auch [TZ87, KTZ90, TZ96, CH88]). Die vorliegende Arbeit ist vor allem durch diese Anwendungsmöglichkeit motiviert. Ein Generator orientierter Matroide bildet so den Grundstein für einen Generator von Isomeren bzw. von Konformeren, einer sinnvollen und seit langem erwünschten Ergänzung des in Bayreuth entwickelten Molekül-Konfigurationsgenerators MOLGEN ([BGH$^+$95, BGK$^+$97, GKL$^+$00]).

Diese Anwendung stellt spezielle Anforderungen an den zu entwickelnden Generator: Er muss bereits während der Generierung zusätzliche Restriktionen wie verbotene Kreise berücksichtigen, damit man durch Ausnutzung chemischer Gesetzmäßigkeiten die Anzahl der zu generierenden Strukturen in Grenzen halten kann. Aus chemischer Sicht sind Konformationen beweglicher Moleküle zu Konformationsräumen, also zu Äquivalenzklassen zusammenfassbar, was ebenfalls zur Reduktion der zu generierenden Repräsentanten (partiell definierte Chirotope) beiträgt. Zudem sind wir nur an einer Transversale von Isomorphieklassen interessiert. Jedoch ist dabei die Struktur der vorgegebenen Konstitution zu berücksichtigen. Es operiert also von vornherein lediglich die Automorphismengruppe des molekularen Graphen anstatt der vollen symmetrischen Gruppe über der Menge der Atome.

Aufgrund dieser spezifischen Nebenbedingungen sind die bereits bestehenden Generatoren von J. Bokowski und A. Guedes de Oliveira ([BGdO00]) sowie von L. Finschi ([Fin01]) leider nicht direkt anwendbar. Wegen der mit n enorm schnell wachsenden Anzahl orientierter Matroide über n erscheint es auch wenig sinnvoll,

einen festen Katalog aller orientierten Matroide zu nutzen, um daraus bei Bedarf die zu einem speziellen Molekül relevanten Strukturen herauszusuchen. (L. Finschi stellt einen derartigen Katalog unter `http://www.om.math.ethz.ch` zur Verfügung: Für uniforme azyklische orientierte Matroide vom Rang 4 reicht der Katalog bis zu $n = 8$, und hierzu existieren bei Operation der vollen symmetrischen Gruppe bereits $160\,020$ Isomorphieklassen.) Stattdessen wurde im Rahmen dieser Arbeit ein neuer Generator entwickelt, welcher spezifisch zu vorgegebenen Restriktionen genau einen passenden Satz von orientierten Matroiden produziert. Dieser basiert auf dem Prinzip der ordnungstreuen Erzeugung [Far78b, Rea78]. Er ist so allgemein gehalten, dass er auch über die chemische Anwendung hinaus Verwendung finden wird. Bzgl. Effizienz ist er – bei höherer Flexibilität – in etwa vergleichbar mit dem Generator von L. Finschi (bei der Generierung uniformer Punktkonfigurationen sowie Reorientierungsklassen ohne weitere Nebenbedingungen, unter Operation der vollen symmetrischen Gruppe; es wurde mit den angegebenen Informationen aus [Fin01] verglichen). Dabei benötigen beide Generatoren den Hauptteil der Zeit zur Kanonisierung der gefundenen Kandidaten.

Auf der Suche nach Möglichkeiten der Effizienzsteigerung wurde die allgemeine Theorie der Konstruktion diskreter Strukturen vorangetrieben: So wurde der effiziente Algorithmus zur Kanonisierung von Graphen mittels iterierter Verfeinerung von B. D. McKay [McK81] unter dem Aspekt des Homomorphieprinzips [Lau93] nach R. Laue analysiert. Dadurch gelang es, eine entsprechende Strategie zum Kanonisieren allgemeiner Klassen diskreter Strukturen zu formulieren. Weiter wurde ein ebenfalls enger Zusammenhang zwischen McKay's Ansatz zur Konstruktion diskreter Strukturen [McK98] und dem Homomorphieprinzip herausgearbeitet. Die Verwendung dieses Ansatzes für orientierte Matroide erlaubt jetzt eine effiziente Kanonisierung mittels iterierter Verfeinerung, da im Gegensatz zur ordnungstreuen Erzeugung nicht mehr vorausgesetzt wird, dass die kanonische Form das Minimum seiner Bahn ist. Diese theoretischen Ergebnisse führen zu einer substantiellen Effizienzsteigerung für den Generator.

Das erste Kapitel dieser Arbeit führt in die Theorie orientierter Matroide ein. Der Schwerpunkt liegt dabei insbesondere auf den realisierten orientierten Matroiden als Vektoren bzw. affine Punkte im Raum. Beweise, falls angegeben, beschränken sich nur auf diese geometrisch realisierte Situation, ansonsten sei auf die Standardliteratur [BLVS$^+$93] verwiesen. In Abschnitt 1.5 werden *Hypergeradenlisten* nach J. Bokowski [Bok92] vorgestellt. Sie sind schon allein deswegen erwähnenswert, weil Bokowski sie als Grundlage für seinen Generator nutzte. Hier werden zu einigen Aussagen alternative Beweise angegeben.

Im zweiten Kapitel führen wir G-Mengen und Gruppenoperationen ein. Dabei heben wir speziell im Hinblick auf die konstruktive Theorie diskreter Strukturen eine bisher nur wenig beachtete Analogie zur linearen Algebra hervor: Offensichtlich sind Transversalen Basen im Sinne der Matroid-Theorie und spielen eine den Basen eines Vektorraums vergleichbare Rolle. Wir führen die Begriffe *G-Faktormenge* einer G-Menge (siehe Definition 2.3.11 on page 48) sowie *Kern* eines G-Homomorphismus (bzgl. einer Transversale des Bildbereichs; siehe Definition 2.4.3 on page 50) neu ein, sodass wir in der Lage sind, das bekannte Homomorphieprinzip ([Lau93]) analog zum Homomorphiesatz linearer Abbildungen zu formulieren (Satz 2.4.4 on page 50):

$$X/\ker_C(f) \cong f(X)\,.$$

Dieses Kapitel liefert an sich also keine grundlegend neuen Erkenntnisse. Vielmehr wird bereits Bekanntes in neuer Form dargestellt, um Analogien zu anderen Teilgebieten der Mathematik aufzuzeigen.

Einige Grundlagen aus der Theorie von Permutationsgruppen, insbesondere Stabilisatorketten und Labelled Branchings, werden in Kapitel 3 kurz vorgestellt. Des Weiteren untersuchen wir in Abschnitt 3.3 topologische Anordnungen zu einem Branching Γ. Ist Γ nämlich vollständiges Labelled Branching einer Gruppe $H \leq S_n$, so erhält man dadurch nach Jerrum [Jer86] eine Transversale von S_n/H. Wir entwickeln ein Kriterium, um im Fall $H \leq G \leq S_n$ aus einem Backtrack-Durchlauf durch G (entlang einer Untergruppenkette) Teilbäume derart abzuschneiden, dass die verbleibenden Blätter genau eine Transversale von G/H bilden. Das gefundene Kriterium ermöglicht es, die Kanonisierung von diskreten Strukturen mit nichttrivialer Automorphismengruppe stark zu beschleunigen.

In Kapitel 4 verallgemeinern wir, wie bereits erwähnt, die bekannten Algorithmen zur Kanonisierung und Konstruktion von Graphen auf allgemeinere Klassen diskreter Strukturen. Wir haben dabei diskrete Strukturen der Form $X \subseteq W^{(n^k)}$ im Auge, mit einer operierenden Gruppe $G \leq S_W \wr S_n \wr S_k$, W endliche Menge, $k, n \in \mathbb{N}$. Dabei ist die Verallgemeinerung derart zu verstehen, dass die Menge $\mathcal{G}(n)$ aller Graphen mit n Knoten mittels Adjazenzmatrix in die G-Menge $2^{(n^2)}$ der Form $W^{(n^k)}$ einbettbar ist. Orientierte Matroide vom Rang k über n sind dann durch Chirotope codierbar, d.h. als Elemente von $3^{(n^k)}$. Ebenfalls in diese Klasse diskreter Strukturen passt die Menge $2^n = 2^{(n^1)}$ aller Teilmengen von n. Es stellt sich heraus, dass der Algorithmus zum Kanonisieren von Graphen via iterierter Verfeinerung nach B.D. McKay [McK81] (bekannt durch das Computerprogramm `nauty` [McK90]) eine iterierte Anwendung des Homomorphieprinzips ist. In der

Tat werden dabei implizit H_i-Homomorphismen $f_{i,j} : X \to Y$, $j < j_i$, zu einer Kette von Untergruppen $H_i \leq G$ betrachtet. Während der Kanonisierung eines Graphen x erhält man so eine Kette von Untergruppen von G oberhalb des Stabilisators G_x,

$$G \geq G_{f_{0,0}(x)} \geq G_{(f_{0,0}(x),f_{0,1}(x))} \geq \cdots \geq G_x \,,$$

entlang der ein kanonisches Element x_c mittels Homomorphieprinzip bestimmt wird. Die H_i-Homomorphismen $f_{i,j}$ werden dabei aus einem einzigen G-Homomorphismus $\Phi : G \times \mathcal{L}(G) \to Y^X$, einem *Verfeinerer* (siehe Definition 4.1.9 on page 78), gewonnen. Ein Algorithmus zum Kanonisieren wird explizit in Pseudocode angegeben (siehe Seiten 90 – 91).

Weiter zeigen wir in Abschnitt 4.2 den ebenso engen Zusammenhang zwischen dem Homomorphieprinzip und McKay's Ansatz aus [McK98] zur Konstruktion diskreter Strukturen auf. Hier ist insbesondere eine Analogie zu dem in Bayreuth von B. Schmalz entwickelten Leiterspiel [Sch93] hervorzuheben. B. D. McKay vermutete diesen Zusammenhang bereits in [McK98]: „There are complicated relationships between these methods, but to a large extent they have not been explored." Für das bereits länger bekannte Leiterspiel ergibt sich aus den Überlegungen McKay's heraus die neue Möglichkeit, mehrstufige Konstruktionen diskreter Strukturen als Backtrack-Algorithmus in Tiefensuche zu formulieren.

Kapitel 5 wendet sich schließlich der Implementierung zu. Der Code ist in der Programmiersprache C++ geschrieben. Es wird u.a. eine C++-Basisklasse für Backtrack-Algorithmen vorgestellt, welche neben der Konstruktion diskreter Strukturen im Rahmen von Weiterentwicklungen zu MOLGEN bereits vielfältige weitere (bisher unveröffentlichte) Anwendungen gefunden hat, wie z.B. Tiefen- bzw. Breitensuche im Graphen oder das Durchlaufen aller Elemente einer Gruppe (gegeben als Transversalenkette).

Die orientierten Matroide werden als Chirotope generiert. (Die bisherige Einschränkung auf uniforme Chirotope ist dabei nicht von prinzipieller Natur. Es ist vorgesehen, dass alle Orientierungen zu einem vorgegebenen Matroid generiert werden können.) Die Konstruktion selbst ist dann im Wesentlichen ein Backtrack-Durchlauf durch alle Abbildungen von $\binom{n}{k} \to \{\pm 1\}$, wovon aufgrund folgender Kriterien Teilbäume jeweils zum frühestmöglichen Zeitpunkt abgeschnitten werden: Test der dreiwertigen Grassmann-Plücker-Relationen, eine benutzerdefinierte Liste verbotener Kreise (insbesondere auch Test auf azyklisch) sowie ein Test, ob die Werte auf den Bahnen einer vorgegebenen Gruppe von Automorphismen konsistent sind. Isomorphe Lösungen unter Negation, Umnummerierung und Reorientierung können nach dem Prinzip der ordnungstreuen Erzeugung unterdrückt werden. Bzgl. Umnummerierungen kann eine beliebige operierende Gruppe $G \leq S_n$

gewählt werden. Wie jedoch oben bereits erläutert wurde, bietet eine Umsetzung der Ergebnisse aus Kapitel 4 noch ein enormes Potential zur Effizienzsteigerung.

Kapitel 6 gibt eine kurze Zusammenfassung und liefert einen Überblick über die geplante Anwendung im Rahmen der Chemie.

An dieser Stelle möchte ich mich bei Herrn Prof. Dr. Adalbert Kerber und Herrn Prof. Dr. Reinhard Laue herzlich für die beständige und umfangreiche Unterstützung bedanken. Außerdem danke ich allen, die zur Entstehung dieser Arbeit beigetragen haben.

Bayreuth, im Juni 2005 *Ralf Gugisch*

Abstract

Following the approach of A. Dress and A. Dreiding [DW80, DDH82], oriented matroids are a helpful tool for presenting conformational information of chemical molecules, as we can describe the affine point configuration of the placed atoms in $\mathbb{R}^3$ as an acyclic oriented matroid of rank 4. In order to receive insight into the space of all possible conformations of a molecule, a generator of oriented matroids is more than useful. But this application requires special features of the generator: It should be able to handle restrictions like forbidden circuits already during the generation process, a prescribed group of known automorphisms should be respected, and we only want to get one representant out of each relabeling class. But in contrast to the common definition of relabeling classes, we should only consider the automorphism group of the molecular graph (the constitution of the molecule) as acting group, instead of the whole symmetric group over the set of atoms. Because of these specific requirements, the existing generator of J. Bokowski and A. Guedes de Oliveira ([BGdO00]) as well as the one of L. Finschi ([Fin01]) are not directly applicable.

This thesis is mainly motivated by the following application: A generator of oriented matroids fulfilling the described requirements is the first step towards a generator of stereoisomers or conformers, which is a reasonable and needed supplement to the molecule-(configuration)-generator MOLGEN ([BGH$^+$95, BGK$^+$97, GKL$^+$00]). Still, the developed generator is kept general and will find further applications beyond the chemical one, too.

Thus, a generator of isomorphism classes of oriented matroids (more exact: of chirotopes) respecting a given set of restrictions has been developed and implemented. Thereby, the isomorphism classes may be chosen with respect to negation, reorientation, or relabeling (with a given acting group $G \leq S_n$), as well as any combination thereof. The generation strategy is based upon the principle of orderly generation according to I. A. Faradzhev and R. C. Read [Far78b, Rea78]. Having a higher degree of flexibility, its efficiency is comparable to the generator of L. Finschi.

Looking for possibilities to improve efficiency, the general theory of constructing discrete structures has been improved: The algorithm of B. D. McKay [McK81] for canonizing graphs via iterated refinement (which is implemented in nauty) has been analyzed under the aspect of the homomorphism principle by R. Laue [Lau93]. It turned out, that during iterated refinement one implicitly considers H_i-homomorphisms $f_{i,j} : X \to Y$, for a chain of subgroups $H_i \leq G$. During canonization of an object x, these H_i-homomorphisms result in an even finer chain of subgroups above the stabilizer G_x:

$$G \geq G_{f_{0,0}(x)} \geq G_{(f_{0,0}(x),f_{0,1}(x))} \geq \cdots \geq G_x \,,$$

and the canonization of x is done with homomorphism principle along this chain. The H_i-homomorphisms are produced by a special G-homomorphism $\Phi : G \times \mathcal{L}(G) \to Y^X$ with $H \leq G_{\Phi(g,H)}$, which we introduce as a *refiner*. We are now able to formulate the principle of iterated refinement for any class of discrete structures, as long as there is a refiner available.

In addition, the connection between McKay's approach of constructing discrete structures [McK98] and the homomorphism principle, especially the ladder game by B. Schmalz [Sch93], has been worked out. B. D. McKay conjectured such a connection already in [McK98]: „There are complicated relationships between these methods, but to a large extent they have not been explored." As a consequence of this connection, it is now possible to formulate the ladder game as a backtrack algorithm, avoiding huge amounts of memory-usage, which were needed so far.

Using these results for the implemented generator of oriented matroids, the possibility to canonize in a much more efficient way than before arises, as we are not restricted to the minimum of each orbit as canonic transversal any more. The latter was necessary for the orderly generation, but the approach according to the homomorphism principle does not depend on this restriction anymore. Thus, the theoretical results of this work still provide a substantial potential to improve efficiency of the generator, which will be implemented in near future.

Inhaltsverzeichnis

1 Orientierte Matroide

Wir führen im Folgenden die wesentlichen Konzepte orientierter Matroide ein. Wir orientieren uns dabei an einer geometrischen Sichtweise von Vektoren im euklidischen Raum. Die Theorie der orientierten Matroide ist zwar allgemeiner als diese geometrische Interpretation – nicht jedes orientierte Matroid lässt sich durch Vektoren im $\mathbb{R}^n$ realisieren. Dennoch rechtfertigt die Anschaulichkeit der Geometrie die vereinfachte Einführung, insbesondere da in dieser Arbeit der Blick hauptsächlich auf realisierbare orientierte Matroide gerichtet ist.

Für eine ausführliche und allgemeine Behandlung orientierter Matroide sei auf das Standardwerk [BLVS$^+$93] verwiesen. Speziell im Zusammenhang mit der Chemie sind auch das Buch [CH88] sowie der Artikel [DDH82] zu nennen. Eine umfangreiche Literaturliste ist unter [Zie96] zu finden.

1.1 Punkt– und Vektorkonfigurationen

Ausgehend von der chemischen Anwendung sind wir an affinen Zusammenhängen zwischen einer gegebenen Menge von Punkten im $\mathbb{R}^3$ interessiert, nämlich den Positionen der einzelnen Atome. Unter Verwendung einer (willkürlichen) Anordnung der Atome untersuchen wir stattdessen eine Folge $p^{(0)}, \ldots, p^{(n-1)}$ von Punkten im $\mathbb{R}^3$, müssen jedoch zu gegebenem Zeitpunkt auch die Auswirkungen einer Umsortierung berücksichtigen.

Zur Analyse affiner Eigenschaften bietet die Geometrie als Standardmethode an, den $\mathbb{R}^3$ in den vierdimensionalen euklidischen Raum $\mathbb{R}^4$ einzubetten. Dabei liege der Ursprung des $\mathbb{R}^4$ nicht im Bild der Einbettung, etwa wie folgt:

$$\hat{} : \mathbb{R}^3 \to \mathbb{R}^4, p = \begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix} \mapsto \hat{p} = \begin{pmatrix} 1 \\ p_1 \\ p_2 \\ p_3 \end{pmatrix}.$$

Damit sind die affinen Eigenschaften der Punkte $p^{(i)}$ im $\mathbb{R}^3$ auf lineare Eigenschaften der Vektoren $\hat{p}^{(i)}$ im $\mathbb{R}^4$ zurückzuführen. Um die unterschiedlichen Sichtweisen auch sprachlich zu trennen, sprechen wir dabei im affinen Zusammenhang stets von *Punkten* und im linearen Zusammenhang von *Vektoren*.

Es ist sinnvoll, gleich die allgemeinere Situation linearer Zusammenhänge zwischen Vektoren zu untersuchen. Sei dazu $v^{(0)}, \ldots, v^{(n-1)}$ eine Folge von n Vektoren im euklidischen Raum $\mathbb{R}^k$. Wir setzen voraus, dass die Vektoren den ganzen Raum aufspannen: $\langle v^{(0)}, \ldots, v^{(n-1)} \rangle = \mathbb{R}^k$. Für affine Punkte $p^{(0)}, \ldots, p^{(n-1)}$ im $\mathbb{R}^d$, $d = k - 1$ erzeugen die Vektoren $\hat{p}^{(0)}, \ldots, \hat{p}^{(n-1)}$ genau dann den ganzen Raum $\mathbb{R}^k$, wenn die Punkte nicht bereits in einer Hyperebene des $\mathbb{R}^d$ enthalten sind. Insbesondere ist dann $n \geq k = d + 1$.

Die Koordinatentupel der Vektoren $v^{(0)}, \ldots, v^{(n-1)}$ schreiben wir als Spalten einer *Koordinatenmatrix* M:

$$M = \begin{pmatrix} v_0^{(0)} & & v_0^{(n-1)} \\ \vdots & \cdots & \vdots \\ v_{k-1}^{(0)} & & v_{k-1}^{(n-1)} \end{pmatrix}.$$

Im Fall, dass die Vektoren nach dem oben beschriebenen Verfahren aus affinen Punkten $p^{(0)}, \ldots, p^{(n-1)}$ des $\mathbb{R}^d$ hervorgehen, enthält die Matrix M in der ersten Zeile nur Einsen. Wir nennen eine Koordinatenmatrix M *affin*, wenn sie als Einbettung von affinen Punkten des $\mathbb{R}^{k-1}$ verstanden werden kann, und zwar auch dann, wenn dies nach einem evtl. durchgeführten Basiswechsel nicht mehr offensichtlich an der ersten Matrixzeile erkennbar ist:

1.1.1 Definition. Sei M eine $k\times n$-Matrix vom Rang k. Dann heißt M *affin*, falls es eine invertierbare $k \times k$-Matrix B gibt, sodass BM in der ersten Zeile nur Einsen stehen hat.

Bei der Analyse der Vektoren (bzw. der zugrunde liegenden affinen Punkte) streben wir danach, den Rückgriff auf die Koordinaten, also auf die Matrix M, zu vermeiden. Hierzu führen wir im Folgenden fünf Strukturen ein. Es stellt sich heraus, dass jede dieser Strukturen ausreicht, um die vier restlichen Strukturen daraus ohne nochmaligen Zugriff auf die Koordinaten in M herzuleiten. Jede dieser Strukturen eignet sich also gleichwertig, die linearen Zusammenhänge der Vektoren koordinatenfrei zu beschreiben.

Anschließend betrachten wir die Situation aus einer höheren Abstraktionsebene: Wir vermeiden alle quantitativen Aussagen wie Distanzen, Gewichtungen, etc. und

beschränken uns lediglich auf die qualitativen Aussagen positiv, negativ oder null. Dies lässt sich bei jeder der fünf Strukturen durchziehen, wir erhalten fünf abstrakte Begriffe, die wir jeweils durch Axiomensysteme definieren. Letztere sind dann die wichtigsten verschiedenen Erscheinungsformen orientierter Matroide: das Chirotop χ, die Menge der Vektoren, die Menge der Kreise sowie die Mengen der Kovektoren und der Kokreise.

Vorweg vereinbaren wir eine abkürzende Schreibweise für Indexmengen, indem wir rekursiv $0 := \emptyset$ und

$$n := \{0, \ldots, n-1\}$$

definieren. Aus dem Zusammenhang wird deutlich, ob n eine Menge oder eine Kardinalität bezeichnet. Weiter markieren wir zur besseren Lesbarkeit Tupel von Indizes stets durch einer Vektorpfeil, etwa

$$\vec{a} := (a_0, \ldots, a_{k-1}) \in n^k \,.$$

Dadurch bleiben Tupel von Indizes von einzelnen Indizes unterscheidbar. Auch identifizieren wir häufig die Indizes mit den Vektoren bzw. mit den Spalten der Koordinatenmatrix.

1.2 Analyse linearer Zusammenhänge

Die Volumenfunktion

Jedem k-Tupel von Vektoren im $\mathbb{R}^k$ ist das orientierte Volumen des dadurch beschriebenen Spanns zugeordnet, die Determinante der entsprechenden Koordinatenmatrix. (Im affinen Fall mit $k = d+1$ entspricht ferner das Volumen des Spanns der Vektoren $\hat{p}^{(i_0)}, \ldots, \hat{p}^{(i_d)}$ bis auf einen Vorfaktor $\frac{1}{d!}$ genau dem affinen Volumen des durch die Punkte $p^{(i_0)}, \ldots, p^{(i_d)}$ aufgespannten Simplexes im $\mathbb{R}^d$.)

Wir definieren also eine Volumenfunktion vol, die jeder k-Auswahl der Vektoren $v^{(0)}, \ldots, v^{(n-1)}$ die entsprechende Determinante zuordnet. Zur einfacheren Handhabung definieren wir dies sogar allgemeiner für alle k-Tupel aus $\{0, \ldots, n-1\}^k$.

1.2.1 Definition. Sei M eine $k \times n$-Matrix vom Rang k. Die *Volumenfunktion* zu M ist wie folgt definiert:

$$\mathrm{vol}_M : n^k \to \mathbb{R}, \mathrm{vol}_M(\vec{a}) := \det(M^{(\vec{a})}) \,,$$

wobei $M^{(\vec{a})}$ diejenige $k \times k$-Matrix bezeichne, welche als j-te Spalte die a_j-te Spalte von M enthält.

Diese Definition ist auch für nicht injektive k-Tupel $\vec{a}$ sinnvoll: Dann enthält die Matrix $M^{(\vec{a})}$ mindestens zwei identische Spalten und somit ist $\mathrm{vol}_M(\vec{a}) = 0$.

Da M vollen Rang hat, ist die Volumenfunktion für mindestens ein k-Tupel ungleich Null. Weiter ist vol_M alternierend, und sie erfüllt die für Determinantenfunktionen generell gültigen Grassmann-Plücker-Relationen:

1.2.2 Satz. *Sei M eine $k \times n$-Matrix vom Rang k. Dann gilt für die Volumenfunktion:*

- vol_M *ist nicht trivial:*

$$\exists\, \vec{a} \in n^k : \mathrm{vol}_M(\vec{a}) \neq 0\,.$$

- vol_M *ist alternierend:*
  Für jedes Tupel $\vec{a} \in n^k$ und jede Permutation $\pi \in S_k$ gilt:

$$\mathrm{vol}_M(a_{\pi^{-1}(0)}, \ldots, a_{\pi^{-1}(k-1)}) = \mathrm{sgn}(\pi) \cdot \mathrm{vol}_M(a_0, \ldots, a_{k-1})\,.$$

- vol_M *erfüllt die Grassmann-Plücker-Relationen:*
  Für je zwei Tupel $\vec{a}, \vec{b} \in n^k$ gilt:

$$\mathrm{vol}_M(\vec{a}) \cdot \mathrm{vol}_M(\vec{b}) = \sum_{i \in k} \mathrm{vol}_M(b_i, a_1, \ldots, a_{k-1}) \cdot \mathrm{vol}_M(b_0, \ldots, \underset{\substack{\uparrow \\ i\text{-te Stelle}}}{a_0}, \ldots, b_{k-1})$$

Bekanntlich lässt sich der euklidische Raum auf zwei Arten orientieren. Die Auswirkungen einer globalen Umorientierung, d.h. einer Vertauschung zweier Zeilen in der Koordinatenmatrix M, führt zur negativen Volumenfunktion $-\mathrm{vol}_M$. Um eine koordinatenfreie Volumenfunktion $\overline{\mathrm{vol}}_M$ zu erhalten, müssen wir also das ungeordnete Paar beider möglichen Volumenfunktionen betrachten:

$$\overline{\mathrm{vol}}_M := \{\mathrm{vol}_M, -\mathrm{vol}_M\}\,.$$

Der Raum der Abhängigkeiten

Die Vektoren $v^{(0)}, \ldots, v^{(n-1)}$ sind genau dann linear abhängig, wenn es nichttriviale Linearkombinationen des Nullvektors, d.h. nichttriviale Lösungen des folgenden linearen Gleichungssystems gibt:

$$M \cdot x = \vec{0}\,.$$

(Im affinen Fall ist letzteres auch gleichbedeutend mit der affinen Abhängigkeit der zugrunde liegenden Punkte $p^{(0)}, \ldots, p^{(n-1)}$.)

Jede Lösung des Gleichungssystems entspricht dabei den Koeffizienten einer Linearkombination des Nullvektors im $\mathbb{R}^k$. Wir bezeichnen jede Lösung als *Abhängigkeit* der Vektoren $v^{(0)}, \ldots, v^{(n-1)}$. Die Menge aller Abhängigkeiten, also die Menge aller Lösungen des linearen Gleichungssystems, bilden einen Unterraum des $\mathbb{R}^n$, den *Raum aller Abhängigkeiten*

$$\mathcal{A}(M) := \{x \in \mathbb{R}^n \mid M \cdot x = \vec{0}\}\,.$$

Es gilt bekanntlich:

1.2.3 Satz. *Sei M eine $k \times n$-Matrix vom Rang k. Der Raum der Abhängigkeiten $\mathcal{A}(M)$ ist ein linearer Unterraum des $\mathbb{R}^n$ von Dimension $n-k$.*

Der Raum der Hyperebenen

Als dritte natürliche Struktur können wir das Bild von $\mathbb{R}^k$ unter Linksmultiplikation mit M^t betrachten:

$$\mathcal{H}(M) := M^t \cdot \mathbb{R}^k = \{M^t \cdot v \mid v \in \mathbb{R}^k\}\,.$$

Für ein $x = M^t v \in \mathcal{H}(M)$ gilt dann: $x = (\langle v^{(0)}, v\rangle, \ldots, \langle v^{(n-1)}, v\rangle)^t$. Betrachtet man die zu v senkrecht stehende Hyperebene $H_v := \{v\}^\perp$, so ist der Abstand eines Vektors $v^{(i)}$ von der Hyperebene H_v durch $\text{dist}(v^{(i)}, H_v) = |\langle v^{(i)}, v\rangle| \cdot ||v||$ gegeben. Der Vektor $M^t v \in \mathcal{H}(M)$ ist also bis auf einen konstanten Faktor $||v||$ genau das Tupel der Abstände der Vektoren $v^{(i)}$ von der Hyperebene H_v des $\mathbb{R}^k$. Diese Abstände sind zusätzlich mit einem Vorzeichen versehen, je nachdem, ob v und $v^{(i)}$ auf der gleichen Seite von H_v liegen, oder nicht. In Analogie zu der Namensgebung bei orientierten Matroiden nennen wir $\mathcal{H}(M)$ den *Raum der Hyperebenen.*

Im affinen Fall entspricht mit $k = d + 1$ jede lineare Hyperebene des $\mathbb{R}^{d+1}$ genau einer affinen Hyperebene des $\mathbb{R}^d$, wobei die Hyperebene parallel zur Einbettung gerade der affinen Hyperebene „im Unendlichen“ entspricht. Die Abstände $\langle \hat{p}^{(i)}, v\rangle$, die im $\mathbb{R}^{d+1}$ gemessen wurden, lassen sich jedoch nicht ohne weiteres interpretieren. Hingegen gibt das Vorzeichen des Skalarprodukts die Seite an, auf der sich der Punkt $p^{(i)}$ von der affinen Hyperebene aus gesehen liegt.

Es gilt:

1.2.4 Satz. *Sei M eine $k \times n$-Matrix vom Rang k. Dann ist der Raum der Hyperebenen $\mathcal{H}(M)$ ein linearer Unterraum des $\mathbb{R}^n$ mit Dimension k, und die Zeilenvektoren von M bilden eine Basis von $\mathcal{H}(M)$.*

Zwischen $\mathcal{A}(M)$ und $\mathcal{H}(M)$ besteht folgender Zusammenhang:

1.2.5 Satz. *Sei M eine $k \times n$-Matrix vom Rang k. Dann sind $\mathcal{A}(M)$ und $\mathcal{H}(M)$ zueinander orthogonale Räume:*

$$\mathcal{A}(M) = \mathcal{H}(M)^{\perp} \quad \textit{und} \quad \mathcal{H}(M) = \mathcal{A}(M)^{\perp}\,.$$

Daraus ergibt sich eine wichtige Dualität:

1.2.6 Satz. *Seien $v^{(0)}, \ldots, v^{(n-1)}$ Vektoren im $\mathbb{R}^k$ mit Koordinatenmatrix M (vom Rang k). Sei weiter $b^{(0)}, \ldots, b^{(n-k)}$ eine Basis von $\mathcal{A}(M)$, sei M^d die Matrix mit den Zeilen $b^{(i)}$, und schließlich $v'^{(0)}, \ldots, v'^{(n-1)}$ die Spaltenvektoren von M^d, also Vektoren im $\mathbb{R}^{n-k}$ mit Koordinatenmatrix M^d.*
Dann existiert zwischen M und M^d folgende Dualität:

$$\mathcal{A}(M) = \mathcal{H}(M^d) \quad \textit{und} \quad \mathcal{H}(M) = \mathcal{A}(M^d)$$

Diese Dualität liefert noch wesentlich weiter reichende Ergebnisse, insbesondere auch für die Volumenfunktion. Da dies in dieser Arbeit jedoch nur von untergeordneter Bedeutung ist, ist hier auf eine Ausarbeitung verzichtet worden.

Minimale Trägermengen

Wir betrachten im Folgenden die Teilmenge $\mathrm{Min}(V)$ eines Vektorraums $V \subseteq \mathbb{R}^n$ aller Vektoren mit minimaler nichttrivialer Trägermenge, bei Koordinatendarstellung bzgl. der Standardbasis der $\mathbb{R}^n$. Wir werden sowohl $\mathrm{Min}(\mathcal{A}(M))$ als auch $\mathrm{Min}(\mathcal{H}(M))$ als weitere Strukturen zur Analyse der linearen Zusammenhänge in Betracht ziehen.

Wir führen die folgenden Überlegungen genauer aus, insbesondere, da hier ein enger Zusammenhang zur Codierungstheorie besteht. Beispielsweise kann die Minimaldistanz eines Codes C bereits aus der Menge $\mathrm{Min}(C)$ berechnet werden. Eine genauere Analyse dieser Zusammenhänge steht jedoch noch aus. Im Rahmen der orientierten Matroide werden letztendlich nur die minimalen *Vorzeichen*vektoren benutzt.

1.2.7 Definition. Für einen Vektor $v \in \mathbb{R}^n$ mit Koordinatendarstellung $v = (v_0, \ldots, v_{n-1})^t$ ist die *Trägermenge*, oder der *Träger* definiert als die Menge der Indizes, an denen v_i nicht verschwindet:

$$T(v) := \{i \in n \mid v_i \neq 0\}\,.$$

1.2.8 Definition. Sei $V \leq \mathbb{R}^n$ ein Untervektorraum. Ein Element $v \in V \setminus \{\vec{0}\}$ heißt *minimal* in V, falls es einen minimalen nichttrivialen Träger besitzt:

$$\forall w \in V : T(w) \subseteq T(v) \Rightarrow w = \vec{0} \text{ oder } T(w) = T(v)\,.$$

Die Menge aller minimalen Elemente von V bezeichnen wir mit

$$\mathrm{Min}(V) := \{v \in V \setminus \{\vec{0}\} \mid v \text{ ist minimal in } V\}\,.$$

Wir leiten nun eine Reihe von Eigenschaften von $\mathrm{Min}(V)$ her:

1.2.9 Lemma. *Sei $V \leq \mathbb{R}^n$. Für jedes $v \in V \setminus \{\vec{0}\}$ gibt es $w \in \mathrm{Min}(V)$ mit $T(w) \subseteq T(v)$.*

Beweis: Dies ist für endlichdimensionale Vektorräume klar. □

1.2.10 Satz. *Sei $V \leq \mathbb{R}^n$. Dann ist $\mathrm{Min}(V)$ ein Erzeugendensystem von V.*

Beweis: Wir zeigen durch Induktion über $k := |T(v)|$, dass jedes $v \in V$ aus $\mathrm{Min}(V)$ erzeugbar ist:
$k = 0$: Der Nullvektor liegt trivialerweise im Erzeugnis von $\mathrm{Min}(V)$.
$k-1 \to k$: Sei $v \in V$ mit $|T(V)| = k$. Falls $v \in \mathrm{Min}(V)$, so ist nichts zu zeigen. Ansonsten gibt es ein $w \in \mathrm{Min}(V)$ mit $T(w) \subsetneq T(v)$. (Es ist $T(w) \neq \emptyset$.) Sei $i \in T(w)$, und v_i, w_i seien die Koordinatenwerte von v bzw. von w an der entsprechenden Koordinate. Dann hat $v' := v - \frac{v_i}{w_i} w$ an der Koordinate i den Wert $v_i - \frac{v_i}{w_i} w_i = 0$, der Träger $T(v') \subsetneq T(v)$ hat also eine Mächtigkeit $|T(v')| < |T(v)| = k$, nach Induktionsannahme liegt v'also im Erzeugnis von $\mathrm{Min}(V)$. Damit liegt auch v im Erzeugnis. □

Wir zeigen im Folgenden, dass die Mächtigkeiten der Träger der $v \in \mathrm{Min}(V)$ nach oben beschränkt sind. Dies wird schließlich bei der Konstruktion von $\mathrm{Min}(V)$ von Nutzen sein. Doch zuvor zwei auch an sich interessante Lemmata:

1.2.11 Lemma. *Sei $V \leq \mathbb{R}^n$ und $w \in \mathrm{Min}(V)$. Dann haben w und ein $v \in V \setminus \{\vec{0}\}$ genau dann gleichen Träger, wenn w und v linear abhängig sind:*

$$T(w) = T(v) \iff \exists \lambda \in \mathbb{R} : v = \lambda w.$$

Beweis: „⇐“: Da $v \neq \vec{0}$ ist, ist auch $\lambda \neq 0$, und die Behauptung ist trivial.
„⇒“: Es sei $i \in T(w)$. Dann verschwindet für $v' := v - \frac{v_i}{w_i} w \in V$ der Wert an der Koordinate i, d.h. für den Träger: $T(v') \subsetneq T(v) \cup T(w) = T(w)$. Da $w \in \mathrm{Min}(V)$ ist, folgt $v' = \vec{0}$, d.h. $v = \frac{v_i}{w_i} w$. □

Die folgende Aussage verallgemeinert die vorherige für mehrere $v^{(i)} \in V$:

1.2.12 Lemma. *Sei $V \leq \mathbb{R}^n$ und $w \in \mathrm{Min}(V)$ mit $k := |T(w)|$, und $\pi_w : \mathbb{R}^n \to \mathbb{R}^{n-k}$ sei die natürliche Projektion auf die Nichtträger-Koordinaten von w. Weiter seien $v^{(1)}, \ldots, v^{(r)} \in V$. Dann sind $w, v^{(1)}, \ldots, v^{(r)}$ genau dann linear unabhängig, wenn $\pi_w(v^{(1)}), \ldots, \pi_w(v^{(r)})$ linear unabhängig sind.*

Beweis:
„⇒“: Angenommen, die projizierten Vektoren $\pi_w(v^{(1)}), \ldots, \pi_w(v^{(r)})$ seien linear abhängig, etwa $\sum_{i=1}^r \lambda_i \pi_w(v^{(i)}) = 0$. Dann gilt für den Träger von $v := \sum_{i=1}^r \lambda_i v^{(i)}$ die Gleichung $T(v) \subseteq T(w)$. Es muss also entweder $T(v) = 0$ sein, was gleichbedeutend mit der linearen Abhängigkeit der $v^{(1)}, \ldots, v^{(r)}$ wäre, oder es muss $T(v) = T(w)$ gelten. Dann wären aber nach Lemma 1.2.11 w und v linear abhängig. Beide Fälle stehen im Widerspruch zur Voraussetzung, dass $(w, v^{(1)}, \ldots, v^{(r)})$ linear unabhängig sind.
„⇐“: Sind $\pi_w(v^{(1)}), \ldots, \pi_w(v^{(r)})$ linear unabhängig, so sind auf jeden Fall auch die Urbilder $v^{(1)}, \ldots, v^{(r)}$ linear unabhängig. Es ist also lediglich zu zeigen, dass w sich nicht als Linearkombination $w = \sum_{i=1}^r \lambda_i v^{(i)}$ darstellen lässt. Letzteres würde aber bedeuten, dass $\pi_w(\sum_{i=1}^r \lambda_i v^{(i)}) = 0$ ist, also $\sum_{i=1}^r \lambda_i \pi_w(v^{(i)}) = 0$, ein Widerspruch zur linearen Unabhängigkeit der $\pi_w(v^{(i)})$. □

1.2.13 Satz. *Ist $V \leq \mathbb{R}^n$ ein Untervektorraum der Dimension k, so haben die $w \in \mathrm{Min}(V)$ höchstens Träger der Mächtigkeit $n - k + 1$.*

Beweis: Wir betrachten ein beliebiges $w \in \mathrm{Min}(V)$ mit $t := |T(w)|$. Das w lässt sich zu einer Basis $w, v^{(1)}, \ldots, v^{(k-1)}$ von V fortsetzen. Betrachten wir nun die Projektion π_w auf die Nichtträger-Koordinaten von w, so sind nach Lemma 1.2.12 auch die $\pi_w(v^{(1)}), \ldots, \pi_w(v^{(k-1)})$ linear unabhängig in $\mathbb{R}^{n-t}$. Also ist die Dimension $n - t \geq k - 1$, woraus $t \leq n - k + 1$ folgt. □

Die Menge $\mathrm{Min}(V)$ lässt sich aus einer Basis des Orthogonalraums $V^\perp$ wie folgt konstruieren: Ist $u^{(0)}, \ldots, u^{(n-k-1)}$ eine Basis von $V^\perp$, und ist $M \in \mathbb{R}^{(n-k)\times n}$ die Matrix, welche als i-te Zeile den Koordinatenvektor von $u^{(i)}$ besitzt, so ist V der Lösungsraum des homogenen Gleichungssystems

$$Mx = \vec{0}\,.$$

Wir geben zu jedem $(n-k+1)$-Tupel $\vec{t} = (t_0, \ldots, t_{n-k})$ von Koordinatenindizes einen Vektor $v^{(\vec{t})} \in \mathrm{Min}(V)$ an, eine Lösung des Gleichungssystems mit minimaler nichttrivialer Trägermenge $T(v^{(\vec{t})}) \subseteq \vec{t}$. Für $0 \leq i \leq n-k$ sei dazu

$M^{(\vec{t},i)}$ die $(n-k)\times(n-k)$-Teilmatrix von M, welche durch Auswahl der Spalten $t_0,\ldots,t_{i-1},t_{i+1},\ldots,t_{n-k}$ entsteht. Das $v^{(\vec{t})}$ ist dann gegeben durch:

$$v_j^{(\vec{t})} := \begin{cases} (-1)^i \det M^{(\vec{t},i)} & \text{falls } j = t_i \\ 0 & \text{sonst.} \end{cases}$$

1.2.14 Satz. *Sei M eine $(n–k)\times n$-Matrix, deren Zeilen eine Basis von $V^\perp$ bilden, und zu einem $(n-k+1)$-Tupel sei $v^{(\vec{t})}$ wie angegeben. Dann besteht $\mathrm{Min}(V)$ im Wesentlichen genau aus den nichttrivialen $v^{(\vec{t})}$ zu den geordneten $(n-k+1)$-Tupeln $\vec{t}$:*

$$\mathrm{Min}(V) = \{\lambda\cdot v^{(\vec{t})} \mid \lambda\in\mathbb{R}, \vec{t}\in\binom{n}{n-k+1}\}\setminus\{\vec{0}\}\,.$$

Beweis: Sei $\vec{t}$ ein $n-k+1$-Tupel. Der Beweis folgt in fünf Schritten:

1. Für den Träger von $v^{(\vec{t})}$ gilt: $T(v^{(\vec{t})})\subseteq\vec{t}$.
Bew.: Dies ist offensichtlich.

2. Es gilt: $v^{(\vec{t})}\in V$.
Bew.: Es sei $M^{(\vec{t})}$ die $(n-k)\times(n-k+1)$-Matrix, welche durch Auswahl aller durch $\vec{t}$ markierten Spalten von M entsteht. Dieser Matrix können wir auf $n-k$ verschiedene Weisen eine zusätzliche erste Zeile hinzufügen, indem wir die i-te Zeile kopieren. Wir erhalten also $n–k$ verschiedene $(n–k+1)\times(n–k+1)$-Matrizen $M^{(\vec{t},i)\prime}$, welche offensichtlich Determinante $\det M^{(\vec{t},i)\prime}=0$ haben. Berechnen wir andererseits die Determinante durch Entwicklung nach erster Zeile, so erhalten wir

$$0=\det M^{(\vec{t},i)\prime}=\sum_{j=0}^{n-k}(-1)^j a_{i,t_j}\det M^{(\vec{t},j)}=\sum_{j=0}^{n-k}a_{i,t_j}v_{t_j}^{(\vec{t})}=(Mv^{(\vec{t})})_i\,.$$

Jeder Koordinateneintrag von $Mv^{(\vec{t})}$ ist also gleich Null, d.h. $v^{(\vec{t})}$ ist eine Lösung des homogenen Gleichungssystems $Mx=\vec{0}$.

3. Ist $v^{(\vec{t})}\neq\vec{0}$, so ist jedes weitere $v\in V$ mit $T(v)\subseteq\vec{t}$ ein Vielfaches von $v^{(\vec{t})}$.
Bew.: Da $v^{(\vec{t})}\neq\vec{0}$ ist, gibt es ein i mit $v_{t_i}^{(\vec{t})}\neq 0$, d.h. $\det M^{(\vec{t},i)}\neq 0$. Hat ein $v\in V$ mit $T(v)\subseteq\vec{t}$ an der Koordinate t_i den Eintrag $v_{t_i}=0$, so spielen in der Gleichung $Mv=\vec{0}$ nur die Koordinaten von $T(v)\subseteq\{t_0,\ldots,t_{i-1},t_{i+1},\ldots,t_{n-k}\}$ eine Rolle, es gilt also bereits $M^{(\vec{t},i)}\cdot(v_{t_0},\ldots,v_{t_{i-1}},v_{t_{i+1}},\ldots,v_{t_{n-k}})^T=\vec{0}$. Die $(n-k)\times(n-k)$-Matrix $M^{(\vec{t},i)}$ hat jedoch Determinante $\neq 0$, es folgt $v=\vec{0}$.

Ist hingegen $v_{t_i} \neq 0$, so ist auch $v' := v - \frac{v_{t_i}}{v^{(\vec{t})}_{t_i}} v^{(\vec{t})}$ eine Lösung des Gleichungssystems. Diese verschwindet an der Koordinate t_i, es ist also nach vorheriger Überlegung $v' = \vec{0}$ und damit ist v ein Vielfaches von $v^{(\vec{t})}$.

4. Es ist $v^{(\vec{t})} \in \mathrm{Min}(V)$.
Bew.: Folgt aus Punkt 3. nach Definition von $\mathrm{Min}(V)$.

5. Ist $w \in \mathrm{Min}(V)$, so ist w Vielfaches von einem $v^{(\vec{t})}$, $\vec{t} \in \binom{n}{n-k+1}$.
Bew.: Sei $\vec{t} = (t_0, \ldots, t_{r-1}) := T(w)$ der Träger von w. Nach Satz 1.2.13 ist $r \leq n-k+1$. Die t_0-te Spalte von M lässt sich dann als Linearkombination der restlichen durch $\vec{t}$ ausgewählten Spalten schreiben. Die Spalten $t_1, \ldots, t_{r-1}$ sind dabei linear unabhängig, sonst würde nämlich eine nichttriviale Linearkombination dieser Spalten eine Lösung des homogenen Gleichungssystems $Mx = \vec{0}$ liefern mit einem echt kleineren Träger als $T(w)$, was im Widerspruch zur Minimalität von w stünde. Nach dem Basisergänzungssatz lassen sich die Spalten $t_1, \ldots, t_{r-1}$ durch weitere Spalten $t_r, \ldots, t_{n-k}$ zu einer Basis des $\mathbb{R}^{n-k}$ fortsetzen, da die Matrix M vollen Rang $n-k$ hat. Das $(n-k+1)$-Tupel $\vec{t} := (t_0, \ldots, t_{n-k})$ liefert also ein $v^{(\vec{t})} \in V$ mit $v^{(\vec{t})}_{t_0} = \det M^{(\vec{t},0)} \neq 0$, also insbesondere $v^{(\vec{t})} \neq \vec{0}$. Aus $T(w) \subseteq \vec{t}$ folgt mit Punkt 3, dass w ein Vielfaches von $v^{(\vec{t})}$ ist.

Aus Punkt 4. folgt „$\supseteq$“, und aus Punkt 5. folgt „$\subseteq$“ in der Behauptung. □

1.2.15 Folgerung. Sei M wie in Satz 1.2.14. Sind $v, v' \in V$ linear unabhängig mit $|T(v) \cup T(v')| \leq n-k+1$, so gilt für jedes $n-k+1$-Tupel $\vec{t}$ mit $T(v) \cup T(v') \subseteq \vec{t}$:

$$v^{(\vec{t})} = \vec{0}\,.$$

Beweis: Wäre $v^{(\vec{t})} \neq \vec{0}$, so wären nach Punkt 3. des Beweises von Satz 1.2.14 sowohl v als auch v' Vielfache von $v^{(\vec{t})}$, also linear abhängig. □

1.2.16 Folgerung. Sei M wie in Satz 1.2.14. Hat ein $v^{(\vec{t})} \neq \vec{0}$ nicht volles Gewicht, d.h. ist $|T(v^{(\vec{t})})| < n-k+1$, so sind für alle weiteren $n-k+1$-Tupel $\vec{t}'$ mit $T(v^{(\vec{t})}) \subseteq \vec{t}'$ die Vektoren $v^{(\vec{t}')}$ Vielfache von $v^{(\vec{t})}$:

$$\vec{t}' \supseteq T(v^{(\vec{t})}) \Longrightarrow \exists \lambda \in \mathbb{R} : v^{(\vec{t}')} = \lambda v^{(\vec{t})}\,.$$

Das Zusammenspiel der Strukturen

Wir haben insgesamt fünf Strukturen vorgestellt, die sich in natürlicher Weise aus der Folge $v^{(0)}, \ldots, v^{(n-1)}$ von n Vektoren im $\mathbb{R}^k$, bzw. der entsprechenden Koordinatenmatrix M ergeben:

1. Die koordinatenfreie Volumenfunktion $\overline{\mathrm{vol}}_M$.
2. Der Raum der Abhängigkeiten $\mathcal{A}(M)$.
3. Die minimalen Abhängigkeiten $\mathrm{Min}(\mathcal{A}(M))$.
4. Der Raum der Hyperebenen $\mathcal{H}(M)$.
5. Die minimalen Hyperebenen $\mathrm{Min}(\mathcal{H}(M))$.

Es bestehen enge Zusammenhänge zwischen diesen Strukturen. So lassen sich Strukturen aus anderen explizit gegebenen Strukturen berechnen, ohne dazu auf die Matrix M zurückzugreifen. Wenngleich dies an dieser Stelle aufgrund des Umfangs der Datenstrukturen von eher theoretischem Nutzen sein kann, ist dies in der Verallgemeinerung der orientierten Matroide wichtig, da nicht jedes orientierte Matroid durch Vektoren in $\mathbb{R}^n$ samt Koordinatenmatrix darstellbar ist.

Beispielsweise ist klar, dass man aus $\mathcal{A}(M)$ direkt die minimalen Abhängigkeiten $\mathrm{Min}(\mathcal{A}(M))$ herausfiltern kann. Wegen Satz 1.2.13 muss man dabei nur die $v \in V$ mit $|T(V)| \leq n - k + 1$ auf Minimalität untersuchen. Mit Lemma 1.2.10 ist auch die umgekehrte Richtung beschrieben, $\mathrm{Min}(V)$ ist ein Erzeugendensystem von V. Entsprechendes gilt zwischen $\mathcal{H}(A)$ und $\mathrm{Min}(\mathcal{H}(A))$.

Der Zusammenhang zwischen $\mathcal{A}(M)$ und $\mathcal{H}(M)$ wurde bereits in Satz 1.2.5 dargelegt: Sie sind zueinander orthogonale Unterräume des $\mathbb{R}^n$.

Schließlich können die minimalen Abhängigkeiten $\mathrm{Min}(\mathcal{A}(M))$ aus der Volumenfunktion vol_M gemäß Satz 1.2.14 konstruiert werden, die Determinanten $\det M^{\vec{t},i}$ sind ja als Werte der Volumenfunktion abzulesen. Es ist jedoch unklar, ob sich die koordinatenfreie Volumenfunktion $\overline{\mathrm{vol}}_M = \{\mathrm{vol}_M, -\mathrm{vol}_M\}$ aus den minimalen Abhängigkeiten ablesen lässt. (Vgl. den entsprechenden Satz für orientierte Matroide, [BLVS$^+$93, Theorem 3.5.5 und Korollar 3.5.12].)

Aus den eingeführten Strukturen kann man die Affinität der Koordinatenmatrizen auf verschiedene Art und Weise ablesen:

1.2.17 Satz. *Sei M eine $k \times n$-Matrix vom Rang k. Dann sind folgende Aussagen äquivalent:*

1. *M ist affin.*
2. *Die Volumenfunktion erfüllt für jedes $k+1$-Tupel $\vec{a} \in n^{k+1}$ die Gleichung:*

$$\sum_{i \in k+1} (-1)^i \,\mathrm{vol}_M(a_0, \ldots, a_{i-1}, a_{i+1}, \ldots, a_k) = 0\,.$$

3. *Für jedes $v \in \mathrm{Min}(\mathcal{A}(M))$ gilt: $\sum_{i \in n} v_i = 0$.*
4. *Es ist $(1, \ldots, 1) \in \mathcal{H}(M)$.*

Beweis: Betrachte die Matrix M', welche aus M entsteht, indem als zusätzliche erste Spalte eine Zeile aus lauter Einsen hinzugefügt wird. Die Koordinatenmatrix M ist genau dann affin, wenn M' vom Rang k ist, ansonsten ist M' vom Rang $k+1$.

„1 $\Leftrightarrow$ 2“: Die Matrix M ist genau dann affin, wenn jede $(k+1)\times(k+1)$-Matrix $M'^{(\vec{a})}$ mit $\vec{a} \in n^{k+1}$ Determinante $\det M^{(\vec{a})\prime} = 0$ besitzt. Entwicklung nach der ersten Zeile liefert dann die Äquivalenz zu den angegebenen Gleichungen.

„1 $\Leftrightarrow$ 3“: Wir vergleichen die Lösungsräume der Gleichungssysteme $Mx = \vec{0}$ und $M'x = \vec{0}$: Offenbar ist die Lösungsmenge des zweiten Gleichungssystems in der Lösungsmenge des ersten enthalten, und die Lösungsmengen sind genau dann gleich, wenn M und M' gleichen Rang haben, also wenn M affin ist. Die Gleichheit der Lösungsmengen heißt aber genau, dass jedes $v \in \mathcal{A}(M)$ auch die zusätzliche Bedingung $(1,\ldots,1)\cdot v$ des zweiten Gleichungssystems erfüllt, also die angegebene Gleichung. Hierzu reicht es, diese Bedingung für ein Erzeugendensystem zu prüfen, also nach Lemma 1.2.10 für alle $v \in \mathrm{Min}(\mathcal{A}(M))$.

„1 $\Leftrightarrow$ 4“: Der Raum $M^t \cdot \mathbb{R}^k$ ist in $M'^t \cdot \mathbb{R}^{k+1}$enthalten, und die Räume sind genau dann gleich, wenn die M' vom Rang k ist. D.h. der zusätzliche Erzeuger $M'^t \cdot e_0 = (1,\ldots,1)^t$ ist genau dann in $\mathcal{H}(M)$ enthalten, wenn M affin ist. □

1.3 Orientierungen

Es folgt nun ein Abstraktionsschritt: Wir vernachlässigen sämtliche Informationen betreffend Koordinaten und Längen, und verwerten lediglich die Informationen betreffend dem Vorzeichen. Dies lässt sich bei jeder der fünf besprochenen Strukturen durchführen:

Volumenfunktion vol_M	$\rightarrow$	Chirotop χ_M
affine Abhängigkeiten $\mathcal{A}(M)$	$\rightarrow$	Vektoren $\mathcal{V}(M)$
$\mathrm{Min}(\mathcal{A}(M))$	$\rightarrow$	Kreise $\mathcal{C}(M)$
Hyperebenen $\mathcal{H}(M)$	$\rightarrow$	Kovektoren $\mathcal{V}^*(M)$
$\mathrm{Min}(\mathcal{H}(M))$	$\rightarrow$	Kokreise $\mathcal{C}^*(M)$

Das Chirotop

Ausgehend von der Volumenfunktion vol_M ordnen wir jedem k-Tupel $\vec{a}$ das Vorzeichen von $\mathrm{vol}_M(\vec{a})$ zu, wir erhalten die *Orientierungsfunktion* oder das *Chirotop*

zu $M \in \mathbb{R}^{k\times n}$:

$$\chi_M : n^k \to \{0, \pm 1\}, \vec{a} \mapsto \operatorname{sgn}(\operatorname{vol}_M(\vec{a}))\,.$$

Die Orientierungsfunktion gibt im zwei- und dreidimensionalen Fall die gängige Auffassung der Orientierung von Vektoren wieder:

- Linear abhängige Vektoren haben Orientierung 0.
- Kann man in der euklidischen Ebene von zwei linear unabhängigen Vektoren $v^{(1)}$ und $v^{(2)}$ den ersten Vektor durch eine Drehung von höchstens 180° gegen den Uhrzeigersinn in die gleiche Richtung bringen wie $v^{(2)}$, so sind die Vektoren positiv orientiert, ansonsten sind sie negativ orientiert.
- Im euklidischen Raum gilt die „Rechte-Hand-Regel“: Kann man den Daumen, Zeige- und Mittelfinger der rechten Hand derart ausrichten, dass sie in die gleichen Richtungen zeigen wie drei linear unabhängige Vektoren $v^{(1)}, v^{(2)}$ und $v^{(3)}$, so sind die Vektoren positiv orientiert, ansonsten sind sie negativ orientiert.

Auch im affinen Fall mit $k = d + 1$ ist eine anschauliche Erklärung der Orientierungsfunktion möglich:

- In der euklidischen Gerade sind zwei affin unabhängige Punkte $p^{(1)}$ und $p^{(2)}$ genau dann positiv orientiert, wenn $p^{(2)}$ rechts von $p^{(1)}$ liegt (d.h. $p^{(1)} < p^{(2)}$).
- In der euklidischen Ebene liegt jedes Tripel von Punkten entweder auf einer Geraden (Orientierung 0), oder sie sind gegen (positiv) bzw. im Uhrzeigersinn (negativ) orientiert.
- Liegen im $\mathbb{R}^3$ vier Punkte in einer Ebene, so ist deren Orientierung 0. Ansonsten betrachtet man die Ebene, welche die letzten drei Punkte enthält. Wählt man einen Standpunkt, sodass der erste Punkt hinter der Ebene liegt, so ist die Orientierung genau dann positiv, wenn von hier aus die letzten drei Punkte gegen den Uhrzeigersinn angeordnet sind.

Wir übertragen die Eigenschaften der Volumenfunktion (Satz 1.2.2) auf das Chirotop:

1.3.1 Satz. *Sei M eine $k\times n$-Matrix vom Rang k. Dann gilt für das Chirotop χ_M:*

- *χ_M ist nicht trivial:*
$$\exists\, \vec{a} \in n^k : \chi_M(\vec{a}) \neq 0\,.$$
- *χ_M ist alternierend:*
Für jedes Tupel $\vec{a} \in n^k$ und jede Permutation $\pi \in S_k$ gilt:
$$\chi_M(a_{\pi^{-1}(0)}, \ldots, a_{\pi^{-1}(k-1)}) = \operatorname{sgn}(\pi) \cdot \chi_M(a_0, \ldots, a_{k-1})\,.$$

- χ_M *erfüllt die binären Grassmann-Plücker-Relationen:*
  Für je zwei Tupel $\vec{a}, \vec{b} \in n^k$ *gilt:*

$$\chi_M(\vec{a}) \cdot \chi_M(\vec{b}) = 1 \Longrightarrow$$
$$\exists i \in n : \chi_M(b_i, a_1, \ldots, a_{k-1}) \cdot \chi_M(b_0, \underset{\substack{\uparrow \\ i\text{-te Stelle}}}{\ldots}, a_0, \ldots, b_{k-1}) = 1\,. \quad \textit{(GP)}$$

Beweis: Lediglich die binären Grassmann-Plücker-Relationen benötigen eine kurze Überlegung: Achten wir bei einer der Relationen aus Satz 1.2.2(3) nur auf die Vorzeichen, so muss bei positiver linker Seite auch die rechte Seite mindestens einen positiven Summanden enthalten. □

Die binären Grassmann-Plücker-Relationen (GP) kann man auch als orientierte Version des Basisaustauschsatzes für Matroide ansehen:

> Sind $v^{(a_0)}, \ldots, v^{(a_{k-1})}$ und $v^{(b_0)}, \ldots, v^{(b_{k-1})}$ zwei Basen im $\mathbb{R}^k$ mit gleicher Orientierung, so gibt es einen Vektor $v^{(b_i)}$ der zweiten Basis, welchen man mit dem ersten Vektor $v^{(a_0)}$ der ersten Basis vertauschen kann und dadurch wieder zwei gleichorientierte Basen des $\mathbb{R}^k$ erhält.

1.3.2. Bezeichnung. Ist $\chi_M(\vec{a}) \neq 0$ (d.h. die entsprechenden Spaltenvektoren von M bilden eine Basis des $\mathbb{R}^k$), so bezeichnen wir auch $\vec{a}$ als *Basis* des Chirotops χ_M.

Entsprechend dem Vorgehen in Abschnitt 1.2 führen wir als koordinatenfreie Orientierungsfunktion bzw. Chirotop das ungeordnete Paar beider möglichen Chirotope ein:

$$\overline{\chi}_M := \{\chi_M, -\chi_M\}\,.$$

Orientierte Mengen

Bei den weiteren Strukturen ordnen wir Vektoren des $\mathbb{R}^n$ stets einen Vorzeichenvektor aus $\{0, \pm 1\}^n$ zu. Um die dabei entstehenden Strukturen besser analysieren zu können, ist eine geeignete Notation nötig. Da wir die Vorzeichenvektoren auch als Verallgemeinerung der Trägermengen interpretieren, ist eine Sichtweise als orientierte Teilmengen der Menge n aller Koordinaten naheliegend. Orientierte Teilmenge heißt dabei, dass jedem Element der Teilmenge ein zusätzliches Vorzeichen, eine Orientierung zugeordnet wird.

1.3.3 Definition. Sei M eine Menge (das Universum). Eine *orientierte Teilmenge* von M ist ein geordnetes Paar $A = (A^+, A^-)$ disjunkter Teilmengen $A^+, A^- \subseteq M$. Ist A orientierte Teilmenge von M, so schreiben wir auch $A \sqsubseteq M$.

Wenn wir von A ohne direkten Bezug zum Universum M sprechen, nennen wir A auch einfach eine orientierte Menge. Die $a \in A^+$ heißen *positive Elemente* von A, im Gegensatz zu den *negativen Elementen* $a \in A^-$. A heißt *rein positiv* (*rein negativ*), wenn sie nur aus positiven (negativen) Elementen besteht, d. h. wenn A^- (A^+) leer ist. Durch Vertauschen aller Vorzeichen erhält man die *entgegengesetzt* orientierte Menge $-A := (A^-, A^+)$. Die Elemente einer orientierten Menge ohne Berücksichtigung der Vorzeichen bilden eine nicht orientierte Menge, wir bezeichnen sie als die *zugrunde liegende Menge* $\underline{A} := A^+ \cup A^-$.

Weiter sind die orientierten Mengen offensichtlich genau die Abbildungen von M in die Menge $\{0, \pm 1\}$ der Orientierungen:

$$A : M \to \{0, \pm 1\}, A(m) := \begin{cases} +1 & \text{falls } m \in A^+ \\ -1 & \text{falls } m \in A^- \\ 0 & \text{sonst.} \end{cases}$$

Insbesondere wenn M von der Form $M = n$ ist, bietet sich die bei Vektoren übliche Schreibweise $A_m := A(m)$ an. Wir werden parallel die Sichtweisen von orientierten Mengen als Paare von Mengen, als Abbildungen und als Vektoren aus $\{0, \pm 1\}^M$ verwenden.

Aufgrund der unterschiedlichen Interpretationen bieten sich auch mehrere Schreibweisen zur Auflistung der Elemente einer orientierten Menge an: Zunächst kann man A wie in der Definition als Paar von Mengen schreiben, z.B.:

$$A = (\{0, 3, 4\}, \{2, 7\}) .$$

Es ist aber oft einfacher, stattdessen die zugrunde liegende Menge aufzulisten und dabei die negativen Elemente durch einen Balken zu markieren. Im Beispiel etwa $A = \{0, \bar{2}, 3, 4, \bar{7}\}$, oder noch kompakter, einfach

$$A = \quad 0\bar{2}34\bar{7} .$$

Eine alternative gebräuchliche Schreibweise im Fall, dass das Universum M geordnet ist (wie etwa im Beispiel: $M = 8$), ist durch die Interpretation als Vorzeichenvektor gegeben: Wir listen für jedes $m \in M$ die Vorzeichen A_m auf:

$$A = \quad +0–++00–.$$

Ein $a \in M$ kann in zwei orientierten Teilmengen $A, B \sqsubseteq M$ unter Umständen mit unterschiedlichen Vorzeichen enthalten sein. Wir sagen a *separiere* die orientierten Mengen A und B. Wir fassen derartige Elemente in der *separierenden Menge* von A und B zusammen:

$$\begin{aligned} S(A,B) &:= (A^+ \cap B^-) \cup (A^- \cap B^+) \\ &\quad (\ = \{m \in M \mid A_m = -B_m \neq 0\}\ ) \end{aligned}$$

Besitzen A und B keine separierenden Elemente, so bezeichnen wir die zwei orientierten Mengen als *konform:*

$$A \sim B :\Longleftrightarrow S(A,B) = \emptyset.$$

Die *konforme Vereinigung* zweier Mengen ist die Vereinigung der konformen Teile:

$$A \uplus B := (A^+ \cup B^+ \setminus S(A,B), A^- \cup B^- \setminus S(A,B))\,.$$

Achtung: Die konforme Vereinigung ist zwar kommutativ, aber nicht assoziativ! Folgende *Komposition* orientierter Mengen spielt deshalb die wichtigere Rolle. Sie dient als nicht kommutativer Ersatz für die Vereinigung nicht konformer orientierter Mengen:

$$\begin{aligned} A \circ B &:= (A^+ \cup B^+ \setminus A^-\,,\, A^- \cup B^- \setminus A^+) \\ &\quad (\ = M \to \{0, \pm 1\}, m \mapsto \begin{cases} A_m & \text{falls } A_m \neq 0, \\ B_m & \text{sonst.} \end{cases}\ ) \end{aligned}$$

In der Komposition $A \circ B$ bevorzugen wir also das Vorzeichen aus A vor denjenigen in B. Sind $A, B \sqsubseteq M$ konform, so ist diese Operation kommutativ.

Schließlich führen wir eine Verbandsstruktur auf der Menge aller orientierten Teilmengen von M ein, die *orientierte Potenzmenge* . Dazu benötigen wir ein künstliches Element $\hat{M} := (M, M)$, um für alle orientierten Teilmengen eine Vereinigung angeben zu können:

$$\mathcal{OP}(M) := \{A \sqsubseteq M\} \dot\cup \{\hat{M}\}$$

Die orientierte Potenzmenge $\mathcal{OP}(M)$ bildet einen vollständigen Verband mittels

$$A_1 \subseteq A_2 :\Longleftrightarrow A_1^+ \subseteq A_2^+ \text{ und } A_1^- \subseteq A_2^-$$

Die Supremum- und Infimumoperatoren des Verbandes sind dabei:

$$A_1 \cap A_2 = (A_1^+ \cap A_2^+, A_1^- \cap A_2^-)$$
$$A_1 \cup A_2 = \begin{cases} A_1 \circ A_2 & \text{falls } A_1 \sim A_2 \\ \hat{M} & \text{sonst.} \end{cases}$$

Orientierte Trägermengen

Einem Vektor $v \in \mathbb{R}^n$ kann man anstelle des Trägers $T(v)$ auch mittels der Signum-Funktion eine orientierte Teilmenge von n zuordnen, die *orientierte Trägermenge* bzw. den *orientierten Träger* von v:

$$OT(v) := \operatorname{sgn} \circ v \in \mathcal{OP}(n) .$$

Die Schreibweise $\operatorname{sgn} \circ v$ rechtfertigt sich dabei dadurch, dass man einen Vektor $v \in \mathbb{R}^n$ auch als Abbildung $v : n \to \mathbb{R}$ auffassen kann.

Als erstes liefern wir ein notwendiges Kriterium für die orientierten Träger zweier orthogonaler Vektoren:

1.3.4 Definition. Zwei orientierte Mengen $A, B \sqsubseteq M$ heißen *orthogonal* zueinander, falls entweder der Schnitt der zugrunde liegenden Mengen leer ist, oder falls es sowohl Elemente gibt, die in A und B gleiches Vorzeichen haben, als auch solche, die in A und B unterschiedliches Vorzeichen haben (d.h. separierende Elemente):

$$A \perp B :\Longleftrightarrow (\, A \sim B \Leftrightarrow A \sim -B \,) .$$

Dieser Begriff der Orthogonalität orientierter Mengen ist mit der euklidischen Orthogonalität zweier Vektoren im $\mathbb{R}^n$ verträglich: Wir schreiben $v \perp w$, falls zwei Vektoren im euklidischen Sinne senkrecht aufeinander stehen, und somit gilt:

1.3.5 Satz. *Seien $v, w \in \mathbb{R}^n$, so gilt*

$$v \perp w \Longrightarrow OT(v) \perp OT(w) .$$

Beweis: Ist v orthogonal zu w, so gilt $\langle v, w \rangle = \sum_{i \in n} v_i w_i = 0$. Der i-te Summand dieser Summe ist genau dann echt negativ, wenn v_i und w_i beide ungleich Null sind, jedoch unterschiedliches Vorzeichen besitzen. Also genau dann, wenn $i \in S(OT(v), OT(w))$. Entsprechend ist der i-te Summand genau dann echt positiv, wenn der Koordinatenindex $i \in S(OT(v), -OT(w))$. Die Summe kann nur dann Null ergeben, wenn sie entweder trivial ist, oder sowohl positive als auch negative Summanden enthält. Dies ist nach vorangehender Überlegung äquivalent zur Orthogonalität der orientierten Mengen $OT(v)$ und $OT(w)$. □

Weiter ordnen wir jedem Untervektorraum $V \subseteq \mathbb{R}^n$ die Menge der orientierten Trägermengen zu:

$$OT(V) := \{OT(v) \mid v \in V\}\,.$$

Diese Menge erfüllt folgende charakteristischen Eigenschaften:

1.3.6 Satz. *Ist $V \leq \mathbb{R}^n$ ein linearer Unterraum, so gilt für $OT(V)$:*

1. $\emptyset \in OT(V)$
2. $A \in OT(V) \Longrightarrow -A \in OT(V)$ *(Symmetrie)*
3. $A, B \in OT(V) \Longrightarrow A \circ B \in OT(V)$ *(Komposition)*
4. *Sind $A, B \in OT(V)$ und ist $i_0 \in S(A, B)$, so gibt es ein $C \in OT(V)$ mit:* *(Elimination)*
$$C_{i_0} = 0 \text{ und } A \uplus B \subseteq C\,.$$

Beweis:

1. Es ist $\vec{0} \in V$, also ist $\operatorname{sgn}(\vec{0}) = \emptyset \in \operatorname{sgn}(V)$.

2. Mit $v \in V$ ist auch $-1 \cdot v \in V$, und damit: $-\operatorname{sgn}(v) = \operatorname{sgn}(-1 \cdot v) \in \operatorname{sgn}(V)$.

3. Seien $v, w \in V$, so ist auch $v' := v + \varepsilon w \in V$. Ist ε hinreichend klein, so ändern sich dabei die Vorzeichen an den Koordinaten $i \in n$ mit $v_i \neq 0$ nicht. Die i mit $v_i = 0$ und $w_i \neq 0$ erhalten in v' durch die Addition von εw dasselbe Vorzeichen wie in w. Der neue Vektor $v' \in V$ hat als Vorzeichenvektor also genau die Komposition der orientierten Mengen $OT(v)$ und $OT(w)$, d.h. $OT(v') = OT(v) \circ OT(w)$.

4. Seien $v, w \in V$ mit $A = OT(v)$ und $B = OT(w)$. Für i_0 gilt nach Definition von $S(A, B)$ die Gleichung $\operatorname{sgn}(v_{i_0}) = -\operatorname{sgn}(w_{i_0}) \neq 0$. Es gibt also ein $\kappa := -\frac{v_{i_0}}{w_{i_0}} > 0$, sodass $v' := v + \kappa w \in V$ an der Stelle i_0 verschwindet. Weiter ändern sich für nicht separierende Elemente i die Vorzeichen durch Addition eines positiven Vielfachen κ von w nicht, also ist $A \uplus B \subseteq OT(v')$. Für $C := OT(v)$ gilt also die Behauptung. □

Insbesondere besagt 1.3.6(3), dass die Menge $OT(V)$ abgeschlossen bezüglich Komposition, also ein $\vee$-Unterhalbverband der orientierten Potenzmenge $\mathcal{OP}(n)$ ist. Andererseits ist $OT(V)$ i.A. *kein* ($\mathbb{F}_3$-)Untervektorraum von $\{0, \pm 1\}^n$, da $OT(V)$ nicht abgeschlossen bzgl. Addition ist.

Zu einer Koordinatenmatrix $M \in \mathbb{R}^{k \times n}$ definieren wir die Menge der *Vektoren* $\mathcal{V}(M)$ und die Menge der *Kovektoren* $\mathcal{V}^*(M)$:

$$\begin{aligned} \mathcal{V}(M) &:= OT(\mathcal{A}(M))\,, \\ \mathcal{V}^*(M) &:= OT(\mathcal{H}(M))\,. \end{aligned}$$

Die Aussage von Satz 1.3.6 ist direkt auf diese beiden Mengen anwendbar. Aus Satz 1.3.5 folgt (vgl. Satz 1.2.5):

1.3.7 Satz. *Sei M eine $k \times n$-Matrix vom Rang k. Dann besteht zwischen der Menge der Vektoren und der Menge der Kovektoren folgende Orthogonalitätsbeziehung:*

$$\forall A \in \mathcal{V}(M), B \in \mathcal{V}^*(M) : A \perp B\,.$$

Minimale orientierte Trägermengen

Die minimalen orientierten Träger eines Vektorraums $V \leq \mathbb{R}^n$ bezeichnen wir als die *Kreise* von $V \leq \mathbb{R}^n$:

$$\mathcal{C}(V) := OT(\mathrm{Min}(V))\,.$$

Zur Koordinatenmatrix $M \in \mathbb{R}^{k\times n}$ erhalten wir so die Menge der *Kreise* $\mathcal{C}(M)$ von M und die Menge der *Kokreise* $\mathcal{C}^*(M)$:

$$\begin{aligned}\mathcal{C}(M) &:= \mathcal{C}(\mathcal{A}(M))\,,\\ \mathcal{C}^*(M) &:= \mathcal{C}(\mathcal{H}(M))\,.\end{aligned}$$

Wir leiten im Folgenden den Satz 1.3.10 her, eine Charakterisierung von $\mathcal{C}(V)$ für einen Vektorraum V. Dieser ist dann sowohl für $\mathcal{C}(M)$ als auch für $\mathcal{C}^*(M)$ anwendbar.

1.3.8 Lemma. *Sei $V \leq \mathbb{R}^n$. Ein $v \in V \setminus \{\vec{0}\}$ hat bereits dann minimalen Träger, wenn der orientierte Träger $OT(v)$ minimal ist:*

$$\Big(\forall w \in V : OT(w) \subseteq OT(v) \Longrightarrow w = \vec{0} \text{ oder } OT(w) = OT(v)\Big) \Longrightarrow v \in \mathrm{Min}(V)$$

Beweis: Sei $v \neq \vec{0}$ mit minimalem orientierten Träger. Nach 1.2.9 gibt es ein $w \in \mathrm{Min}(V)$ mit $T(w) \subseteq T(v)$. Ohne Einschränkung gebe es separierende Elemente zwischen $OT(w)$ und $OT(v)$. (Sonst wähle $-w \in \mathrm{Min}(V)$.) Sei $i \in S(OT(w), OT(v))$ ein separierender Koordinatenindex, d.h. die i-ten Komponenten v_i und w_i haben unterschiedliches Vorzeichen. Insbesondere ist $\frac{v_i}{w_i} < 0$. Dabei sei i derart gewählt, dass $|\frac{v_i}{w_i}|$ minimal ist unter allen separierenden Koordinatenindizes, d.h. für alle $j \in S(OT(w), OT(v))$ gilt: $|\frac{v_i}{w_i}| \leq |\frac{v_j}{w_j}|$.

Wir betrachten $v' := v - \frac{v_i}{w_i} w \in V$: Der Faktor ist genau so gewählt, dass die i-te Komponente von v' verschwindet, d.h. $T(v') \subsetneq T(v)$. Wir zeigen, dass sogar $OT(v') \subsetneq OT(v)$ gilt. Aufgrund der Minimalität von $OT(v)$ folgt dann $v' = \vec{0}$, d.h. v ist ein Vielfaches von w und damit in $\mathrm{Min}(V)$ enthalten.

Sei dazu $j \neq i$ ein weiterer separierender Koordinatenindex zwischen $OT(w)$ und $OT(v)$. Ist $v'_j \neq 0$, so gilt wegen der Wahl von i: $|\frac{v_i}{w_i}| < |\frac{v_j}{w_j}|$. Weiter: $\mathrm{sgn}(v'_j) = \mathrm{sgn}(v_j - \frac{v_i}{w_i} w_j) = \mathrm{sgn}(\frac{v_j}{w_j} - \frac{v_i}{w_i}) \cdot \mathrm{sgn}(w_j) = -\mathrm{sgn}(w_j) = \mathrm{sgn}(v_j)$. Für die restlichen Indizes $j \in T(w) \setminus S(OT(w), OT(v))$ haben v_j und w_j gleiches Vorzeichen, also gilt: $\mathrm{sgn}(v'_j) = \mathrm{sgn}(v_j - \frac{v_i}{w_i} w_j) = \mathrm{sgn}(v_j)$, da $\frac{v_i}{w_i} < 0$. Insgesamt ist also tatsächlich $OT(v') \subsetneq OT(v)$, und die Behauptung ist gezeigt. □

1.3.9 Folgerung. Sei $V \leq \mathbb{R}^n$. Zu jedem $v \in V$ gibt es ein $w \in \mathrm{Min}(V)$ mit $OT(w) \subseteq OT(v)$.

1.3.10 Satz. *Sei $V \leq \mathbb{R}^n$. Für die Menge der Kreise $\mathcal{C}(V)$ gilt:*

1. $\emptyset \notin \mathcal{C}(V)$.
2. $A \in \mathcal{C}(V) \Longrightarrow -A \in \mathcal{C}(V)$ (Symmetrie)
3. $A, B \in \mathcal{C}(V)$ mit $\underline{A} \subseteq \underline{B} \Longrightarrow A = \pm B$. (Unvergleichbarkeit)
4. Für alle $A, B \in \mathcal{C}(V)$ mit $A \neq -B$ und $i_0 \in S(A, B)$ gibt es ein $C \in \mathcal{C}(V)$ mit:
$$C_{i_0} = 0 \text{ und } \forall i \in \underline{C} : C_i \in \{A_i, B_i\}.$$
(Schwache Kreis Elimination)

Beweis:

1. Klar.

2. Mit $v \in \mathrm{Min}(V)$ ist auch $-1 \cdot v \in \mathrm{Min}(V)$.

3. Es seien $v, w \in \mathrm{Min}(V)$ mit $A = OT(v)$ und $B = OT(w)$. Ist $\underline{A} \subset \underline{B}$, d.h. $T(v) \subseteq T(w)$, so folgt wegen der Definition von $\mathrm{Min}(V)$: $T(v) = T(w)$, und wegen Lemma 1.2.11 folgt, dass v und w Vielfache voneinander sind. Daraus folgt die Behauptung.

4. Nach Satz 1.3.6(4) findet man auf jeden Fall ein $C' := OT(v')$ in $OT(\mathcal{V})$ mit den gewünschten Eigenschaften. Also gibt es nach Lemma 1.3.9 auch einen minimalen Vektor w' mit $OT(w') \leq OT(v')$. Damit ist $C := OT(w') \in \mathcal{C}(V)$ der gesuchte Kreis. □

Die Affinität einer $k \times n$-Matrix kann man bereits an den orientierten Trägermengen erkennen (vgl Satz 1.2.17):

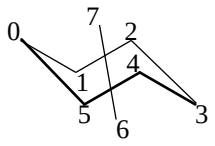


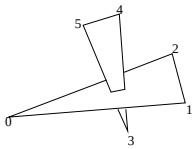
Abbildung 1.1: Eine ungewöhnliche chemische Konformation, wird durch die Abhängigkeit $124\bar{5}\bar{6}\bar{7}$ erkannt.

1.3.11 Satz. *Sei M eine $k \times n$-Matrix vom Rang k. Dann sind folgende Aussagen äquivalent:*

1. *M ist affin.*
2. *Es gibt keine rein positiven Kreise in $\mathcal{C}(M)$.*
3. *Es ist $(+, \ldots, +) \in \mathcal{V}^*(M)$.*

Beweis: Die Richtungen „1 $\Rightarrow$ 2“ und „1 $\Rightarrow$ 3“ folgen direkt aus Satz 1.2.17. Die Rückrichtungen sind in der Standardliteratur zu orientierten Matroiden (z.B. [BLVS$^+$93]) allgemeiner bewiesen. □

Im affinen Fall von n Punkten $p^{(0)}, \ldots, p^{(n-1)}$ im $\mathbb{R}^d$, $k = d+1$ und Koordinatenmatrix $M \in \mathbb{R}^{k \times n}$ haben die Vorzeichenvektoren $\mathcal{V}(M)$ eine schöne geometrische Interpretation: Sie bilden Radon--Partitionen: Eine *Radon-Partition* (X, Y) ist ein Paar von disjunkten Punktemengen, deren konvexe Hüllen sich schneiden, also $X, Y \subseteq \mathbb{R}^d$, $X \cap Y = \emptyset$ und $\overline{X} \cap \overline{Y} \neq \emptyset$, (wobei $\overline{X}$ die konvexe Hülle bezeichne, die von den Punkten $p \in X$ aufgespannt wird).



Eine Radon-Partition $012\bar{3}\bar{4}\bar{5}$.

Die Kreise $\mathcal{C}(M)$ sind dann genau die minimalen Radon--Partitionen, also diejenigen, deren Hüllen sich nach Entnahme eines einzigen Punktes bereits nicht mehr schneiden. Diese Interpretation spielt eine wichtige Rolle in der Chemie, da wir chemische Restriktionen oft in Form von „unmöglichen Kreisen“ bzw „unmöglichen Abhängigkeiten“ formulieren können (siehe z.B. Abbildung 1.1).

1.4 Orientierte Matroide

Die Charakteristika aus Satz 1.3.1 eignen sich, um ein abstraktes Chirotop zu definieren:

1.4.1 Definition. Sei $k \leq n$. Dann heißt eine Abbildung $\chi : n^k \to \{0, \pm 1\}$ *Chirotop* der Ordnung n vom Rang k, falls gilt:

- χ ist nicht trivial:
$$\exists\, \vec{a} \in n^k : \chi(\vec{a}) \neq 0\,.$$
- χ ist alternierend:
Für jedes Tupel $\vec{a} \in n^k$ und jede Permutation $\pi \in S_k$ gilt:
$$\chi(a_{\pi^{-1}(0)}, \ldots, a_{\pi^{-1}(k-1)}) = \operatorname{sgn}(\pi) \cdot \chi(a_0, \ldots, a_{k-1})\,.$$
- χ erfüllt die binären Grassmann-Plücker-Relationen:
Für alle $\vec{a}, \vec{b} \in n^k$ gilt:
$$\begin{aligned}&\chi(\vec{a}) \cdot \chi(\vec{b}) = 1 \Longrightarrow \\ &\quad \exists i \in n : \chi(b_i, a_1, \ldots, a_{k-1}) \cdot \chi(b_0, \underset{\substack{\uparrow \\ i\text{-te Stelle}}}{\ldots}, a_0, \ldots, b_{k-1}) = 1\,. \qquad \text{(GP)}\end{aligned}$$

Für ein Chirotop χ setzen wir

$$\overline{\chi} := \{\chi, -\chi\}.$$

Natürlich ist jedes χ_M zu einer regulären $k \times n$-Matrix auch ein Chirotop im Sinne von Definition 1.4.1. Umgekehrt stellt sich die Frage, wann einer Funktion χ, welche die Kriterien von Definition 1.4.1 erfüllt, auch Vektoren $v^{(0)}, \ldots, v^{(n-1)}$ im $\mathbb{R}^k$ samt Koordinatenmatrix M zugeordnet werden können, sodass $\chi = \chi_M$ gilt. Falls dies möglich ist, heißt χ *realisierbar*. Ist die zugeordnete Koordinatenmatrix affin, sodass χ als Orientierungsfunktion von n Punkten im $\mathbb{R}^d$, $d = k - 1$ verstanden werden kann, so heißt χ auch *affin realisierbar*.

Es ist leicht erkennbar, wann ein realisierbares Chirotop auch affin realisierbar ist (vgl. Satz 1.3.11). Allerdings ist bisher kein polynomialer Algorithmus bekannt, um auf Realisierbarkeit zu testen. Ebenso wenig ist bisher die Realisierung eines realisierbaren Chirotops in polynomialer Zeit durchführbar, wenngleich es brauchbare Algorithmen gibt, siehe [RG96].

1.4.2. Bemerkung und Definition. Die Basen (im Sinne von Bezeichnung 1.3.2) eines Chirotops $\chi : n^k \to \{0, \pm 1\}^n$ bilden die Menge der Basen eines nichtorientierten Matroids $\underline{\mathcal{M}}_\chi$. Wir bezeichnen das entsprechende nichtorientierte Matroid als das dem Chirotop χ *zugrunde liegende Matroid* $\underline{\mathcal{M}}_\chi$.

Beweis: Die Menge der Basen von χ ist nichtleer, sie formen eine Antikette, und aus den Grassmann-Plücker-Relationen folgt direkt der Basisaustauschsatz. □

Entsprechend Satz 1.2.14 können wir dem Chirotop eine Menge von *Kreisen* zuordnen:

1.4.3 Definition. Sei $\chi : n^k \to \{0, \pm 1\}$ ein Chirotop. Dann sei für ein $(k+1)$-Tupel $\vec{t} \in \binom{n}{k+1}$ der Vorzeichenvektor $C^{\vec{t}} \in \{0, \pm 1\}^n$ definiert durch:

$$C_i^{\vec{t}} := \begin{cases} (-1)^j \chi(t_0, \ldots, t_{j-1}, t_{j+1}, \ldots, t_k) & \text{falls } i = t_j \\ 0 & \text{sonst.} \end{cases}$$

Dann ist die Menge der Kreise zum Chirotop:

$$\mathcal{C}_\chi := \{\varepsilon C^{\vec{t}} \mid \epsilon = \pm 1, t \in \binom{n}{k+1}\}\,.$$

Die Menge der Vektoren sei weiter der Abschluss von $\mathcal{C}_\chi$ unter Komposition:

$$\mathcal{V}_\chi := \{C^{(0)} \circ \cdots \circ C^{(n-1)} \mid C^{(i)} \in \mathcal{C}_\chi\}\,,$$

die Menge der Kovektoren sei der Orthogonalraum zu $\mathcal{V}_\chi$:

$$\mathcal{V}_\chi^\star = \mathcal{V}_\chi^\perp = \{V^* \in \{0, \pm 1\}^n \mid \forall V \in \mathcal{V}_\chi : V^* \perp V\}\,,$$

und schließlich sei die Menge der Kokreise genau die minimalen nichttrivialen Vorzeichenvektoren in $\mathcal{V}_\chi^*$:

$$\mathcal{C}_\chi^* = \{C^* \in \mathcal{V}_\chi^* \mid C^* \neq \emptyset \text{ und } C^* \text{ ist minimal in } \mathcal{V}_\chi^* \setminus \{\emptyset\}\}\,.$$

Für die eingeführten Strukturen $\mathcal{V}_\chi$, $\mathcal{C}_\chi$, $\mathcal{V}_\chi^*$ und $\mathcal{C}_\chi^*$ gelten damit die Aussagen der Sätze 1.3.6 bzw. 1.3.10.

Wir nennen χ *azyklisch*, falls $\mathcal{C}_\chi$ keine rein positiven Kreise enthält. Nach Satz 1.3.11 ist ein realisierbares Chirotop also genau dann affin realisierbar, wenn χ azyklisch ist.

1.4.4 Definition. Das Quintupel $(\overline{\chi}, \mathcal{C}_\chi, \mathcal{V}_\chi, \mathcal{C}^*_\chi, \mathcal{V}^*_\chi)$ heißt *orientiertes Matroid*.

Man kann orientierte Matroide auch über Axiomensysteme für $\mathcal{V}$ (Satz 1.3.6) oder $\mathcal{C}$ (Satz 1.3.10), bzw. entsprechend auch für $\mathcal{V}^*$ oder $\mathcal{C}^*$ einführen. Es gibt ein Verfahren, um aus den Kreisen auf das Chirotop zu schließen, sodass diese Axiomensysteme in der Tat äquivalent sind (siehe [BLVS$^+$93, Theorem 3.5.5 und Korollar 3.5.12]).

Mit folgender bereits für nichtorientierte Matroide gebräuchlichen Definition können wir einige Spezialfälle ausschließen:

1.4.5 Definition. Sei $\chi : n^k \to \{0, \pm 1\}$ ein Chirotop. Ein Element $i \in n$ heißt *Schleife* des Chirotops, falls $\{i\}$ ein Kreis ist. Zwei Elemente i und j heißen *parallel* (*antiparallel*) bzgl. χ, falls $\{e, f\}$ (bzw. $\{e, \bar{f}\}$) ein Kreis ist. Das Chirotop heißt *schlicht*, falls es bzgl. χ weder Schleifen noch (anti-)parallele Elemente gibt.

Dreiwertige Grassmann-Plücker-Relationen

Im Wesentlichen werden wir Chirotope generieren, indem wir alle alternierenden Abbildungen $\chi : n^k \to \{0, \pm 1\}$ durchlaufen, und für jeden Kandidaten die Grassmann-Plücker-Relationen überprüfen. Es ist aber zum Glück nicht nötig, alle n^{2k} derartigen Relationen zu berücksichtigen.

Erfüllt eine Abbildung χ nämlich die ersten beiden Kriterien der Definition 1.4.1, und bilden weiter die $\vec{a} \in n^k$ mit $\chi(\vec{a}) \neq 0$ eine Menge von Basen eines nichtorientierten Matroids $\underline{\mathcal{M}}$, so kann man sich beim Test, ob χ ein Chirotop ist, auf die dreiwertigen Grassmann-Plücker-Relationen beschränken, wie folgendes wichtige (und keineswegs triviale) Ergebnis der Theorie besagt:

1.4.6 Satz. *Sei $\chi : n^k \to \{0, \pm 1\}$ eine alternierende, nicht triviale Abbildung, deren Träger $T(\chi) := \{\vec{a} \in n^k \mid \chi(\vec{a}) \neq 0\}$ die Menge der Basen eines (nichtorientierten) Matroids bildet.*
Dann ist χ genau dann ein Chirotop, wenn χ die dreiwertigen Grassmann-Plücker-Relationen (GP3) erfüllt, d.h. alle GP-Relationen für Vektoren $\vec{a}, \vec{b} \in n^k$, die sich an genau 2 Stellen unterscheiden:

- Für $\vec{x} \in n^{k-2}$ und $a_1, a_2, b_1, b_2 \in n$ gelten mit[1]

[1] Ist $\vec{a} \in n^r$ ein r-Tupel, und $i \in n$, so schreiben wir f¨ur die Fortsetzung von $\vec{a}$ zu einem $r+1$-Tupel kurz $(\vec{a}, i)$. Entsprechend ist $(\vec{a}, i, j)$, usw. zu verstehen.

$$\vec{a} := (\vec{x}, a_1, a_2) \qquad \vec{a}' := (\vec{x}, b_1, a_2) \qquad \vec{a}'' := (\vec{x}, b_2, a_2)$$
$$\vec{b} := (\vec{x}, b_1, b_2) \qquad \vec{b}' := (\vec{x}, a_1, b_2) \qquad \vec{b}'' := (\vec{x}, b_1, a_1)$$

und mit

$$s_1 := \chi(\vec{a}) \cdot \chi(\vec{b}) \qquad s_2 := \chi(\vec{a}') \cdot \chi(\vec{b}') \qquad s_3 := \chi(\vec{a}'') \cdot \chi(\vec{b}'')$$

folgende Implikationen:

$$\begin{aligned} s_1 = \ \ 1 &\implies s_2 = \ \ 1 \ \vee \ s_3 = \ \ 1 \\ s_1 = -1 &\implies s_2 = -1 \ \vee \ s_3 = -1 \end{aligned} \qquad \text{(GP3)}$$

Beweis: Es ist klar, dass für ein Chirotop die Bedingung (GP3) gilt, es treten ja nur Spezialfälle der allgemeinen Grassmann-Plücker-Relationen aus (GP) auf.
Für einen Beweis der hinreichenden Eigenschaft siehe [BLVS$^+$93, Theorem 3.6.2]. □

Sind die Voraussetzungen des Satzes erfüllt, so reduzieren sich die n^{2k} zu testenden Relationen aus GP auf nur n^{k+2} Relationen. Es gilt sogar:

1.4.7. Bemerkung: Sei χ wie in Satz 1.4.6. Zur Verifikation, ob χ ein Chirotop ist, sind lediglich die $\binom{n}{4}\binom{n-4}{k-2}$ $(= \binom{n}{k+2}\binom{k+2}{4})$ folgenden Bedingungen mit streng monoton steigenden disjunkten Folgen $(a_1, a_2, b_1, b_2) \in n^4$ und $\vec{x} \in n^{k-2}$ zu überprüfen (bei gleichen Bezeichnungen wie in Satz 1.4.6):

$$(s_1 = 0 \wedge (s_2 = -s_3)) \vee (s_1 \neq 0 \wedge (s_1 = s_2 \vee s_1 = s_3)) \qquad \text{(GP3')}$$

Beweis: Benutzen wir die Tatsache, dass χ alternierend ist, so sind die GP3--Relationen trivial, falls das $(k+2)$-Tupel $\vec{t} = (\vec{x}, a_1, a_2, b_1, b_2)$ nicht aus paarweise verschiedenen Elementen besteht. Ist nämlich $a_i = b_i$, $i \in \{1, 2\}$, so gilt $s_1 = s_2$, ist $a_i = b_j$, $\{i, j\} = \{1, 2\}$, so gilt $s_1 = s_3$, und in allen anderen Fällen gilt $s_1 = 0$. Für jede nichttriviale (GP3)-Relation bilden die Parameter also eine $(k+2)$-Menge $\{\vec{x}, a_1, a_2, b_1, b_2\}$.

Permutationen der Folgenglieder von $\vec{x}$ verändern höchstens sämtliche Vorzeichen in den beiden Relationen von (GP3), führen also zu äquivalenten Bedingungen. Permutationen auf (a_1, a_2, b_1, b_2) führen bis auf evtl. eine globale Vorzeichenvertauschung zu einer GP3-Bedingung mit folgenden Parametern:

(a) $\vec{x}$ und $(a_1, a_2;\ b_1, b_2)$,
(b) $\vec{x}$ und $(b_1, a_2;\ a_1, b_2)$,
(c) $\vec{x}$ und $(b_2, a_2;\ b_1, a_1)$.

Es genügt also, zu jedem Paar einer $(k-2)$-Menge $\{\vec{x}\}$ und einer dazu disjunkten 4-Menge $\{a_1, a_2, b_1, b_2\}$ die drei GP3-Relationen mit den genannten Parametersätzen zu überprüfen. Die sechs auftretenden Aussagen

$$\begin{aligned}
s_1 = \;\; 1 &\Longrightarrow \;\; s_2 = \;\; 1 \;\vee\;\; s_3 = \;\; 1\\
s_1 = -1 &\Longrightarrow \;\; s_2 = -1 \;\vee\;\; s_3 = -1\\
s_2 = \;\; 1 &\Longrightarrow \;\; s_1 = \;\; 1 \;\vee\; -s_3 = \;\; 1\\
s_2 = -1 &\Longrightarrow \;\; s_1 = -1 \;\vee\; -s_3 = -1\\
s_3 = \;\; 1 &\Longrightarrow -s_2 = \;\; 1 \;\vee\;\; s_1 = \;\; 1\\
s_3 = -1 &\Longrightarrow -s_2 = -1 \;\vee\;\; s_1 = -1
\end{aligned}$$

kann man äquivalent umformen zu der Aussage (GP3'), da nach Definition von χ die $s_1, s_2, s_3 \in \{0, \pm 1\}$ sind. □

1.5 Hypergeradenlisten

Zur kompakteren Darstellung von Chirotopen (zumindest im Fall $k \leq \frac{n}{2}$) benutzen wir die von J. Bokowski zuerst 1978 eingeführte Struktur der Hypergeradenlisten (siehe [Bok92]). Wir führen diese wieder anhand der geometrischen Anschauung realisierter Chirotope ein. Seien dazu $v^{(0)}, \ldots, v^{(n-1)} \in \mathbb{R}^k$ Vektoren in allgemeiner Lage mit Koordinatenmatrix M und Chirotop $\chi = \chi_M$.

Eine *Hypergerade* bezeichne einen Unterraum des $\mathbb{R}^k$ der Dimension $k-2$ (vgl. Bezeichnung Hyperebene = Unterraum der Dimension $k-1$). Die Hypergeradenliste speichert geeignete Informationen, die aus den Projektionen entlang verschiedener Hypergeraden abzulesen sind. Zur Veranschaulichung ist in Abbildung 1.2 eine derartige Projektion für ein affines Beispiel von 7 Punkten im $\mathbb{R}^3$ (d.h $k = 4$) skizziert.

Mit $\vec{a} \in n^{k-2}$ seien $k-2$ linear unabhängige Vektoren $v^{(a_0)}, \ldots, v^{(a_{k-3})}$ ausgewählt, und $\pi_{\vec{a}}$ sei die entsprechende Projektion entlang der davon aufgespannten Hypergerade in deren Orthogonalraum, also in eine Ebene isomorph zum $\mathbb{R}^2$. Nach dem Basisergänzungssatz gibt es zwei weitere Vektoren $v^{(b)}, v^{(b')}$, sodass $(v^{(a_0)}, \ldots, v^{(a_{k-3})}, v^{(b)}, v^{(b')})$ eine Basis des $\mathbb{R}^k$ bilden. Die Indizes $b, b' \in n$ seien dabei minimal gewählt.

Gemäß den projizierten Vektoren können wir die $v^{(i)}$, $i \in n$, zu Äquivalenzklassen zusammenfassen: Diejenigen Vektoren, welche auf den Nullvektor projiziert werden, bilden eine Äquivalenzklasse $K_0^{(\vec{a})} = \{v^{(i)} \mid v^{(i)} \in \pi_{\vec{a}}^{-1}(\vec{0})\}$, die übrigen

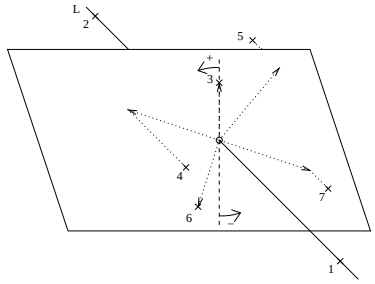


Abbildung 1.2: Projektion entlang einer Hypergeraden L:
Die orientierte Anordnung ist: [1,2]<[+3]<[+4,–7]<[–5]<[+6]

Klassen werden durch eindimensionale Unterräume (Geraden) G der Projektionsebene bestimmt, sind also von der Form $\{v^{(i)} \mid v^{(i)} \in \pi_{\vec{a}}^{-1}(G \setminus \vec{0})\}$. Die dadurch definierte Äquivalenzrelation auf den Indizes $i \in n$ bezeichnen wir mit $\sim_{\vec{a}}$. In jeder Äquivalenzklasse von Indizes sei ein kanonischer Repräsentant durch den minimalen Index i_0 bestimmt. Den kanonischen Repräsentanten zu einem Index $i \in n$ bezeichnen wir mit $\mathrm{hlq}(\vec{a}, i)$ (hlq steht für englisch ***hyperline equivalency***):

$$\mathrm{hlq}(\vec{a}, i) := \min\{j \mid i \sim_{\vec{a}} j\}\,.$$

Man kann hlq aus dem Chirotop ablesen. Der folgende Satz gibt diesen für orientierte Matroide bekannten Sachverhalt auch für allgemeine (nichtorientierte) Matroide wieder. Wir erinnern uns, dass $\mathcal{B}$ nach Definition genau dann die Menge der Basen eines nichtorientierten Matroids $\underline{\mathcal{M}}$ vom Rang k über n ist, wenn $\mathcal{B}$ eine nichtleere Menge von k-Teilmengen von n ist, die das Basisaustauschaxiom erfüllt.

1.5.1 Satz. *Sei $\mathcal{B}$ die Menge der Basen eines (nichtorientierten) Matroids $\underline{\mathcal{M}}$ über n vom Rang k. Zu $\vec{a} \in n^{k-2}$ und $i \in n$ sei:*

$$K_0^{(\vec{a})} := \begin{cases} \{0, \ldots, n-1\} & \textit{falls } \vec{a} \textit{ abhängig ist} \\ \{c \in n \mid (\vec{a}, b, c), (\vec{a}, b', c) \notin \mathcal{B}\} & \textit{sonst,} \end{cases}$$

wobei im unabhängigen Fall $b, b' \in n$ minimal seien mit $(\vec{a}, b, b') \in \mathcal{B}$. Weiter sei

$$\mathrm{hlq}(\vec{a}, i) := \begin{cases} \min K_0^{(\vec{a})} & \textit{falls } i \in K_0^{(\vec{a})} \\ \min\{c \in n \setminus K_0^{(\vec{a})} \mid (\vec{a}, i, c) \notin \mathcal{B}\} & \textit{sonst.} \end{cases}$$

Dann gilt für die Abbildung $\mathrm{hlq} : n^{k-1} \to \mathbb{Z}$*:*

$(\vec{a}, i, j) \in \mathcal{B} \iff \mathrm{hlq}(\vec{a}, i), \mathrm{hlq}(\vec{a}, j)$ *und* $\mathrm{hlq}(\vec{a}, a_0)$ *sind paarweise verschieden.*

Beweis:
Wir zeigen zunächst: Für festes $\vec{a} \in n^{k-2}$ bildet die Menge $K_0^{(\vec{a})}$ gemeinsam mit den $K_i'^{(\vec{a})} := \{c \in n \setminus K_0^{(\vec{a})} \mid (\vec{a}, i, c) \notin \mathcal{B}\}$, $i \notin K_0^{(\vec{a})}$ eine Mengenpartition von n:
Ist $\vec{a}$ abhängig, so ist diese Aussage trivial. Sei also $\vec{a}$ unabhängig. Zunächst ist für $i \notin K_0^{(\vec{a})}$ trivialerweise $i \in K_i'^{(\vec{a})}$, also ergibt die Vereinigung der Klassen $K_i'^{(\vec{a})}$ samt $K_0^{(\vec{a})}$ ganz n. Weiter gilt $i \notin K_0^{(\vec{a})}$ genau dann, wenn sich $(\vec{a}, i)$ zu einer Basis $(\vec{a}, i, i')$ vervollständigen lässt, mit $i' \in \{b, b'\}$. Ist nun für $i, j \notin K_0^{(\vec{a})}$: $i \in K_j'^{(\vec{a})}$ (d.h. $(\vec{a}, i, j) \notin \mathcal{B}$), so gilt $K_i'^{(\vec{a})} = K_j'^{(\vec{a})}$: Wäre nämlich ein $c \in K_i'^{(\vec{a})} \setminus K_j'^{(\vec{a})}$, also $(\vec{a}, i, c) \notin \mathcal{B}$, aber $(\vec{a}, j, c) \in \mathcal{B}$, so folgt aus dem Basisaustauschsatz zu den Basen $(\vec{a}, i, i')$ und $(\vec{a}, j, c)$ ein Widerspruch zu $(\vec{a}, i, j), (\vec{a}, i, c) \notin \mathcal{B}$. Wir haben $K_i'^{(\vec{a})} \subseteq K_j'^{(\vec{a})}$ gezeigt. Da $i \in K_j'^{(\vec{a})}$ äquivalent zu $j \in K_i'^{(\vec{a})}$ ist, gilt auch die umgekehrte Inklusion. Insgesamt folgt daraus die paarweise Disjunktheit der Mengen $K_i'^{(\vec{a})}$, $i \notin K_0'^{(\vec{a})}$.

Damit zeigen wir die eigentliche Behauptung.
„⇒“: Ist $(\vec{a}, i, j) \in \mathcal{B}$,so ist insbesondere $\vec{a}$ nicht abhängig. Aufgrund des Basisaustauschsatzes zu den Basen $(\vec{a}, i, j)$ und $(\vec{a}, b, b')$ ist entweder $(\vec{a}, i, b)$ oder $(\vec{a}, i, b')$ eine Basis, also ist $i \notin K_0^{(\vec{a})}$. Damit ist auch $\mathrm{hlq}(\vec{a}, i) \neq \mathrm{hlq}(\vec{a}, a_0)$, da $a_0 \in K_0^{(\vec{a})}$ ist. Analog folgt $\mathrm{hlq}(\vec{a}, j) \neq \mathrm{hlq}(\vec{a}, a_0)$. Weiter ist $j \notin K_i'^{(\vec{a})}$, also folgt $K_i'^{(\vec{a})} \cap K_j'^{(\vec{a})} = \emptyset$ und damit $\mathrm{hlq}(\vec{a}, i) \neq \mathrm{hlq}(\vec{a}, j)$.

Zu „⇐“ zeigen wir die Kontraposition:
Sei $(\vec{a}, i, j) \notin \mathcal{B}$. Ist eines der beiden Tupel $(\vec{a}, i)$ bzw. $(\vec{a}, j)$ abhängig, so ist $\mathrm{hlq}(\vec{a}, i) = \mathrm{hlq}(\vec{a}, a_0)$ bzw. $\mathrm{hlq}(\vec{a}, j) = \mathrm{hlq}(\vec{a}, a_0)$. Sind hingegen $(\vec{a}, i)$ und $(\vec{a}, j)$ jeweils unabhängig, so gilt $i, j \notin I_0$. Es ist aber $i \in K_j'^{(\vec{a})}$, also $K_i'^{(\vec{a})} = K_j'^{(\vec{a})}$, und damit auch $\mathrm{hlq}(\vec{a}, i) = \mathrm{hlq}(\vec{a}, j)$. □

Man überzeuge sich, dass die im Satz abstrakt definierte Funktion hlq mit der zuvor anschaulich motivierten Definition im realisierten Fall übereinstimmt.

Wir kehren zur realisierten Anschauung zurück (vgl. auch Abbildung 1.2): Die Geraden zu den Äquivalenzklassen $\neq K_0$ sind zyklisch um den Ursprung angeordnet. Orientieren wir die Projektionsebene geeignet, so können wir dies benutzen, um Orientierungen von k-Tupeln von Vektoren abzulesen. Dazu betrachten wir die

Projektionsebene von derjenigen Seite, welche die projizierten Vektoren $\pi_{\vec{a}}(v^{(b)})$ und $\pi_{\vec{a}}(v^{(b')})$ in richtiger Orientierung erscheinen lässt: Sie sollen genau dann gegen den Uhrzeigersinn (positiv) orientiert erscheinen, wenn $\chi_M(\vec{a}, b, b') > 0$ ist. Von diesem Blickpunkt aus gilt auch für die anderen Paare $v^{(i)}$, $v^{(j)}$ von Vektoren $\chi_M(\vec{a}, i, j) > 0$ genau dann, wenn die projizierten Vektoren linear unabhängig sind, und $\pi_{\vec{a}}(v^{(i)})$ durch eine Drehung um weniger als 180° gegen den Uhrzeigersinn in gleiche Richtung mit $\pi_{\vec{a}}(v^{(j)})$ zu bringen ist.

Wir wählen weiter die Gerade $G_1 := \langle \pi_{\vec{a}}(v^{(b)}) \rangle$ als Referenz aus und bezeichnen die zugehörige Äquivalenzklasse mit $K_1^{(\vec{a})}$. Damit können wir die restlichen Äquivalenzklassen $K_2^{(\vec{a})}, \ldots, K_r^{(\vec{a})}$ derart nummerieren, dass die entsprechenden Geraden $G_1, \ldots, G_r$ gegen den Uhrzeigersinn angeordnet sind.

Diese Anordnung lässt sich wie folgt aus dem Chirotop bestimmen.

1.5.2 Lemma. *Sei $\chi : n^k \to \{0, \pm 1\}$ ein orientiertes Matroid, und sei $\vec{a} \in n^{k-2}$ sowie $(\vec{a}, b, b')$ die lexikographisch minimale Fortsetzung zu einer Basis von χ. Sei $I := \{\mathrm{hlq}(\vec{a}, i) \mid i \in n\} \setminus \{\mathrm{hlq}(\vec{a}, a_0), b\}$ die Indexmenge der kanonischen Repräsentanten der Äquivalenzklassen $K_2, \ldots, K_r$. Dann ist auf I eine Ordnung definiert durch*

$$i \leq_{\vec{a}} j :\Longleftrightarrow \chi(\vec{a}, b, i) \cdot \chi(\vec{a}, b, j) \cdot \chi(\vec{a}, i, j) \geq 0\,.$$

Wir führen einen alternativen Beweis für diese bekannte Aussage an, welcher eine dreiwertige Grassmann-Plücker-Relation geschickt ausnutzt:

Beweis: Die Menge I ist so gewählt, dass für $i, j \in I$ mit $i \neq j$ keine der Faktoren Null wird.

- Die Relation ist reflexiv, da $\chi(\vec{a}, i, i) = 0$ für alle $i \in n$ gilt.
- Ist $i \leq_{\vec{a}} j$ und $j \leq_{\vec{a}} i$, so folgt $\chi(\vec{a}, i, j) = \chi(\vec{a}, j, i)$, also $\chi(\vec{a}, i, j) = 0$, da χ alternierend ist. Aus der Vorbemerkung folgt $i = j$.
- Zur Transitivität betrachten wir den nichttrivialen Fall, mit i, j, l paarweise verschieden. Sei $i <_{\vec{a}} j$ und $j <_{\vec{a}} l$, d.h. es gelte:

$$\begin{aligned} &\text{(I)} && \chi(\vec{a}, b, i)\chi(\vec{a}, b, j)\chi(\vec{a}, i, j) > 0, \\ &\text{(II)} && \chi(\vec{a}, b, j)\chi(\vec{a}, b, l)\chi(\vec{a}, j, l) > 0\,. \end{aligned}$$

Um $i <_{\vec{a}} l$ zu zeigen, muss folgende Aussage gelten

$$\overset{!}{\Longrightarrow} \qquad \chi(\vec{a}, b, i)\chi(\vec{a}, b, l)\chi(\vec{a}, i, l) > 0\,.$$

Wir benutzen die dreiwertige Grassmann-Plücker-Relation zu (l, b, i, j) und $\vec{a}$; nach Voraussetzung sind alle vorkommenden Werte $\neq 0$, deswegen reduziert sich bei Bezeichnung wie in 1.4.6 die logische Aussage (GP3) zu

$$s_1 = s_2 \vee s_1 = s_3 \,.$$

Dies ergibt:

$$\begin{array}{ll} \text{(a)} & \chi(\vec{a}, b, l)\chi(\vec{a}, i, j) \;=\; \chi(\vec{a}, b, i)\chi(\vec{a}, l, j) \quad \text{oder} \\ \text{(b)} & \chi(\vec{a}, b, l)\chi(\vec{a}, i, j) \;=\; \chi(\vec{a}, b, j)\chi(\vec{a}, i, l)\,. \end{array}$$

Durch Multiplikation mit $\chi(\vec{a}, b, l)$ erhalten wir

$$\begin{array}{ll} \text{(a)} & \chi(\vec{a}, i, j) \;=\; \chi(\vec{a}, b, l)\chi(\vec{a}, b, i)\chi(\vec{a}, l, j) \quad \text{oder} \\ \text{(b)} & \chi(\vec{a}, i, j) \;=\; \chi(\vec{a}, b, l)\chi(\vec{a}, b, j)\chi(\vec{a}, i, l)\,. \end{array}$$

Ersetzt man im Fall (a) $\chi(\vec{a}, i, j)$ in Gleichung (I), so erhält man einen Widerspruch zu Gleichung (II). Also muss Fall (b) gelten. In (I) eingesetzt ergibt, dies die Behauptung. □

Wir können also die Nummerierung der Äquivalenzklassen $K_2, \ldots, K_r$ durch Sortieren der Repräsentanten gewinnen. Auf diese Weise erhalten wir eine Anordnung aller Äquivalenzklassen $K_0^{(\vec{a})}, \ldots, K_r^{(\vec{a})}$, und wir ordnen jedem der Vektoren v_i den Ordnungsindex $j = \mathrm{hlo}(\vec{a}, i)$ der entsprechenden Äquivalenzklasse zu (hlo steht für ***hyperline order***):

$$\mathrm{hlo}(\vec{a}, i) = j :\Longleftrightarrow v^{(i)} \in K_j^{(\vec{a})}\,.$$

Wir führen unsere Überlegungen entlang der geometrischen Anschauung fort:

Um die Referenzgerade G_1 mit einer weiteren Geraden in Deckung zu bringen, ist höchstens eine Drehung bis zu 180° gegen den Uhrzeigersinn nötig. Wir ordnen jedem Vektor ein Vorzeichen zu, je nachdem, mit welcher Hälfte der Geraden er bei entsprechender Drehung der Referenzgeraden in Deckung kommt. Der Vektor v_i erhält genau dann positives Vorzeichen $\mathrm{hls}(\vec{a}, i)$ (***hyperline sign***), wenn $\pi_{\vec{a}}(v^{(i)})$ nach Drehung der Referenzgeraden G_1 auf derselben Hälfte liegt, wie ursprünglich $\pi_{\vec{a}}(v^{(b)})$.

Die Berechnung von $\mathrm{hls}(\vec{a}, i)$ aus dem Chirotop geschieht durch:

$$\mathrm{hls}(\vec{a}, i) := \begin{cases} \chi(\vec{a}, b, i) & \text{falls } \chi(\vec{a}, b, i) \neq 0 \\ \chi(\vec{a}, b', b) \cdot \chi(\vec{a}, b', i) & \text{sonst}\,. \end{cases}$$

Damit gilt:

1.5.3 $$\chi(\vec{a},i,j) = \mathrm{hls}(\vec{a},i)\cdot\mathrm{hls}(\vec{a},j)\cdot\mathrm{sgn}(\mathrm{hlo}(\vec{a},j)-\mathrm{hlo}(\vec{a},i))\,.$$

Dies kann man einerseits leicht an den Definitionen nachrechnen (benötigt jedoch einige Fallunterscheidungen), oder sich im realisierten Fall klar machen: Zwei nichttriviale projizierte Vektoren $\pi_{\vec{a}}(v^{(i)})$ und $\pi_{\vec{a}}(v^{(j)})$ sind nämlich genau dann gegen den Uhrzeigersinn angeordnet, wenn sie gleiche Vorzeichen $\mathrm{hls}(\vec{a},i) = \mathrm{hls}(\vec{a},j)$ haben und $\mathrm{hlo}(\vec{a},i) < \mathrm{hlo}(\vec{a},j)$ ist, oder wenn sie unterschiedliche Vorzeichen $\mathrm{hls}(\vec{a},i) \neq \mathrm{hls}(\vec{a},j)$ besitzen und dafür $\mathrm{hlo}(\vec{a},i) > \mathrm{hlo}(\vec{a},j)$ gilt.

Der Vollständigkeit wegen setzen wir noch für abhängige Tupel $\vec{a} \in n^{k-2}$ von Vektoren und für $i \in n$: $\mathrm{hlo}(\vec{a},i) := 0$, $\mathrm{hls}(\vec{a},i) := 0$. Wir haben damit Abbildungen von $n^{k-1} \to \mathbb{Z}$ definiert. Wir fassen die Informationen in einer Abbildung zusammen, der Hypergeradenliste hll_χ (englisch: *hyperlinelist*) zu einem gegebenen Chirotop χ:

$$\mathrm{hll}_\chi : n^{k-1} \to \mathbb{Z}, (\vec{a},i) := \mathrm{hls}(\vec{a},i)\cdot\mathrm{hlo}(\vec{a},i)\,.$$

Mit Gleichung 1.5.3 ist klar, dass sich daraus das Chirotop χ rekonstruieren lässt:

1.5.4 Definition. Sei $\mathrm{hll} : n^{k-1} \to \mathbb{Z}$ eine Abbildung. Wir definieren zu $\vec{a} \in n^{k-2}$ und $i \in n$:

$$\begin{aligned}\mathrm{hls}(\vec{a},i) &:= \mathrm{sgn}(\mathrm{hll}(\vec{a},i))\,,\\ \mathrm{hlo}(\vec{a},i) &:= \mathrm{abs}(\mathrm{hll}(\vec{a},i))\,.\end{aligned}$$

Und weiter:

$$\begin{aligned}\chi_{\mathrm{hll}} : \quad n^k &\to \{0,\pm1\},\\ (\vec{a},i,j) &\mapsto \mathrm{hls}(\vec{a},i)\cdot\mathrm{hls}(\vec{a},j)\cdot\mathrm{sgn}(\mathrm{hlo}(\vec{a},j)-\mathrm{hlo}(\vec{a},i))\,.\end{aligned}$$

1.5.5 Satz. *Sei $\chi : n^k \to \{0,\pm1\}$ ein Chirotop und sei $\mathrm{hll} := \mathrm{hll}_\chi$. Dann gilt:*

$$\chi = \chi_{\mathrm{hll}}\,.$$

Die Hypergeradenliste ist also eine zum Chirotop äquivalente Datenstruktur. Für die Kodierung eines orientierten Matroids als Hypergeradenliste benötigt man n^{k-1} Integer-Zahlen, um sie im Computer zu speichern (also genau genommen $n^{k-1}\cdot\log(n)$ Bytes), im Vergleich zu $O(n^k)$ Bytes für das Chirotop. Für $k < \frac{n}{2}$ und große n ist das ein merklicher Gewinn. Zum Beispiel benötigt man für ein orientiertes Matroid vom Rang 4 über 50 Elementen 125KB Speicher, gegenüber

782KB bei Speicherung des Chirotops, wenn man 2 Bit für jedes Vorzeichen vorsieht. Bei einem orientierten Matroid über 100 Punkten wird der Unterschied noch klarer: 1MB reicht für die Hypergeradenliste aus, während das Chirotop bereits 25MB benötigt.

Da jedoch die Reichweite der Konstruktion orientierter Matroide bereits für wesentlich kleinere n an ihre Grenzen stößt, ist der Speicherplatz ein sekundäres Problem. J. Bokowski nutzt die Hypergeradenlisten, um einen effizienten Generator von uniformen orientierten Matroiden zu formulieren. Es gilt nämlich

1.5.6 Satz. *Sei* $\mathrm{hll} : n^{k-1} \to \mathbb{Z}$ *und* $\chi := \chi_{\mathrm{hll}}$ *definiert wie in 1.5.4. Ist* χ *alternierend, und bilden die Basen von* χ *die Basen eines Matroids, so erfüllt* χ *auch die dreiwertigen binären Grassmann-Plücker-Relationen, d.h.* χ *ist ein Chirotop.*

Beweis: Seien $\vec{a} \in n^{k-2}$ und $i, j, k, l \in n$. Wir zeigen (GP3):

$$\chi(\vec{a}, i, j) \cdot \chi(\vec{a}, k, l) = \pm 1 \Longrightarrow \\ \chi(\vec{a}, k, j) \cdot \chi(\vec{a}, i, l) = \pm 1 \quad \text{oder} \quad \chi(\vec{a}, l, j) \cdot \chi(\vec{a}, k, i) = \pm 1\,.$$

Nach Einsetzen der Definition für χ_{hll} enthalten alle Terme den Faktor

$$\varepsilon := \mathrm{hls}(\vec{a}, i)\,\mathrm{hls}(\vec{a}, j)\,\mathrm{hls}(\vec{a}, k)\,\mathrm{hls}(\vec{a}, l)\,.$$

Setzen wir weiter für $x \in \{i, j, k, l\}$:

$$o_x := \mathrm{hlo}(\vec{a}, x)$$

so vereinfacht sich die GP3-Relation nach Kürzen von ε (die folgende abkürzende Schreibweise steht für zwei simultane Aussagen betreffend Ungleichungen mit „$>$“ oder „$<$“):

$$(o_j - o_i)(o_l - o_k) \gtrless 0 \quad \Longrightarrow \quad (o_j - o_k)(o_l - o_i) \gtrless 0 \;\vee\; (o_j - o_l)(o_i - o_k) \gtrless 0\,.$$

Aufgrund der Symmetrie der GP3-Relationen können wir o.E. annehmen, dass $o_i = \mathrm{hlo}(\vec{a}, i)$ minimal unter den betrachteten Werten ist. Es folgt:

$$(o_l - o_k) \gtrless 0 \quad \Longrightarrow \quad (o_j - o_k) \gtrless 0 \;\vee\; (o_l - o_j) \gtrless 0\,.$$

Dies ist äquivalent zu:

$$o_l \gtrless o_k \quad \Longrightarrow o_j \gtrless o_k \quad \vee \quad o_l \gtrless o_j\,,$$

eine für Zahlen o_j, o_k und o_l allgemein gültige Aussage. □

Man kann also analog zu J. Bokowskis Ansatz zu einem gegebenem nichtorientierten Matroid $\mathcal{M}$ alle Orientierungen konstruieren, indem wir die wesentlich verschiedenen alternierenden Abbildungen $\mathrm{hll} : n^{k-1} \to \mathbb{Z}$ generieren, sodass die Funktion χ_{hll} als Basen genau die Basen von $\mathcal{M}$ hat. (Bekanntlich ist nicht jedes Matroid orientierbar, aber dies erkennt der Algorithmus korrekt und gibt dann 0 Orientierungen zurück.)

Dies wurde in einer frühen Phase der Dissertation auch durchgeführt. Es hat sich aber herausgestellt, dass unter Verwendung der ordnungstreuen Erzeugung (siehe Kapitel 5) die direkte Generierung aller alternierenden Funktionen von $n^k \to \{\pm 1\}$ samt Test der dreiwertigen Grassmann-Plücker-Relationen effizienter ist. Dies ist damit erklärbar, dass die erzielten Lerneffekte im Backtrackbaum (Überspringen von Teilen) wesentlich genauer umgesetzt werden. Bei der Erzeugung alternierender Funktionen $\chi \in 2^{(n^k)}$ hat jeder Knoten des Baums genau 2 Söhne, während bei der Erzeugung der Hypergeradenliste eine höhere Anzahl von Söhnen pro Knoten vorkommt.

1.6 Isomorphieklassen von Chirotopen

Es werden drei Arten von Isomorphie für Chirotope betrachtet: Seien $\chi, \chi' : n^k \to \{0, \pm\}$ zwei Chirotope.

1. Ein χ' heißt *Umnummerierung* von χ, falls es durch eine Permutation $\pi \in S_n$ der Elemente $i \in n$ aus χ hervorgeht: $\chi' = \chi \circ \pi^{-1}$.
2. Wir haben bereits gesehen, dass χ und $-\chi$ das gleiche orientierte Matroid bestimmen. Die Äquivalenzklassen $\overline{\chi} = \{\chi, -\chi\}$ definieren also eine weitere wichtige Isomorphie für Chirotope. Das Chirotop $-\chi$ heißt *Negation* von χ. (Wir können also nur bei Berücksichtigung dieser Art von Isomorphie von einer Konstruktion von orientierten Matroiden sprechen.)
3. Im Rahmen der Zonotop-Theorie ([BLVS$^+$93, Abschnitt 2.2]) spielen *Reorientierungen* von Chirotopen eine wichtige Rolle: χ' heißt Reorientierung von χ, falls es eine Teilmenge $I \subseteq n$ gibt mit

   $$\chi'(\vec{a}) = -1^{|I \cap \vec{a}|} \chi(\vec{a})\,.$$

   In der Realisierung durch Vektoren erkennt man eine Reorientierung dadurch, dass einige der Vektoren v_i durch $-v_i$ ersetzt werden.

Um die Isomorphieklassen von Chirotopen zu analysieren, betrachten wir induzierte Gruppenoperationen auf der Menge $Y^X = \{0, \pm 1)^{(n^k)}$ (vgl. [Ker99]): Die symmetrische Gruppe $G := S_n$ operiert in natürlicher Weise auf der Menge $X = n^k$ der k-Tupel über n, und $H := S_2$ operiert auf $Y = \{0, \pm 1\}$ durch Negation. Es ergibt sich:

1. Die Symmetrieklassen von Umnummerierungen werden durch die Bahnen von ${}_G(Y^X)$ charakterisiert.
2. Die Symmetrieklassen unter Umnummerierung und Negation sind genau die Bahnen unter der Gruppenoperation ${}_{G\times H}(Y^X)$.
3. Symmetrieklassen unter Reorientierung und Umnummerierung erhält man als die Bahnen des Kranzprodukts: ${}_{H\wr_n G}(Y^X)$. Jedoch ist dies nicht die Standardsituation, da $X \neq n$ ist. Es gilt:

   $$(\varphi, g)f(\vec{a}) := \varphi(g^{-1}a_0) \cdots \varphi(g^{-1}a_{k-1}) f(g^{-1}a_0, \ldots, g^{-1}a_{k-1}) \,.$$

   Dies ist eine Gruppenoperation, da die Gruppe H kommutativ ist.
4. Falls k gerade ist, ist durch Reorientierung und Umnummerierung noch nicht die Negation berücksichtigt. Es gibt also weiter die Symmetrieklassen unter Reorientierung, Umnummerierung und Negation: ${}_{H\times(H\wr_n G)}(Y^X)$.

Jede dieser Symmetriebegriffe kann bei der Konstruktion von Chirotopen berücksichtigt werden. Auf Einzelheiten wird in Kapitel 5 eingegangen.

2 G-Mengen

In diesem Kapitel behandeln wir Gruppenoperationen bzw. G-Mengen insbesondere im Hinblick auf deren Anwendung als Hilfsmittel bei der Konstruktion diskreter Strukturen. Als Literatur sei hier auf [Ker99] und [Lau93] verwiesen.

Im Gegensatz zu der spätestens seit Wielandt ([Wie64]) vorherrschenden Nutzung von Gruppenoperationen als Werkzeug der Gruppentheorie sind im Rahmen der konstruktiven Theorie diskreter Strukturen andere Schwerpunkte zu setzen. Insgesamt ist hier das Augenmerk deutlich in Richtung auf die Menge X verschoben. Die Bahnen einer Gruppenoperation, die zwar starke Aussagen über die operierende Gruppe G zulassen (siehe z.B. Fundamentallemma), verlieren etwas an Bedeutung. Zudem ist es bei unserer Problemstellung nicht sinnvoll, sich von vornherein auf transitive Gruppenoperationen zurückzuziehen. Wir sind ja gerade an nicht-transitiven G-Mengen von diskreten Strukturen interessiert. Dabei ist unser zentrales Ziel, eine Transversale zu konstruieren.

Erstaunlich weitreichend ist die an sich triviale Beobachtung, dass Transversalen Basen im Sinne der Matroid-Theorie sind. Sie übernehmen somit eine den Basen in der linearen Algebra vergleichbare Rolle. Dies führt zu einer bisher nur wenig beachteten Analogie zwischen Gruppenoperationen ${}_GX$ und Vektorräumen ${}_{\mathbb{K}}V$, welche an dieser Stelle ausgearbeitet wird. Dabei wird die Nomenklatur entsprechend angepasst. So bevorzugen wir konsequenterweise die ebenfalls von A. Dress (z.B. in [DK70]) favorisierte Bezeichnung *G-Menge*, da so bereits in der Namensgebung die Menge X in den Vordergrund gestellt ist, ähnlich wie bei der Bezeichnung eines K-Vektorraums V nicht der Körper, sondern der Raum V betont wird.

Ferner führen wir die Begriffe *G-Faktormenge* einer G-Menge (siehe Definition 2.3.11) sowie *Kern* eines G-Homomorphismus (siehe Definition 2.4.3) neu ein, sodass wir in der Lage sind, das bekannte Homomorphieprinzip ([Lau93]) analog zum Homomorphiesatz linearer Abbildungen zu formulieren (siehe Satz 2.4.4):

$$X/\ker_C(f) \cong f(X)\,.$$

Das Kapitel liefert also keine grundlegend neuen Erkenntnisse. Vielmehr wird bereits Bekanntes in einer neuen Form dargestellt, um Analogien zu anderen Teilgebieten der Mathematik aufzuzeigen.

2.1 Transversale und Dimension

2.1.1 Definition. Sei G Gruppe. Eine Menge X heißt *G-(Links-)Menge*, falls es eine „Multiplikation“

$$\cdot : G \times X \to X, (g, x) \mapsto g \cdot x$$

gibt, welche mit der vorhandenen Struktur auf G verträglich ist:

$$g \cdot (g' \cdot x) = (gg') \cdot x \qquad \text{und} \qquad 1 \cdot x = x\,.$$

Dabei schreiben wir häufig kurz gx statt $g \cdot x$. Wir verdeutlichen den Sachverhalt, dass X eine G-Menge ist, kurz durch die Schreibweise ${}_GX$, und sagen alternativ auch, G *operiere* auf X, bzw. ${}_GX$ sei eine *Gruppenoperation*.

2.1.2 Definition. Eine Teilmenge $U \subseteq X$ heißt *G-Untermenge*, kurz $U \leq X$, falls U unter der Multiplikation mit Gruppenelementen abgeschlossen ist, also

$$\forall g \in G, u \in U : gu \in U\,.$$

Mit der Einschränkung der Multiplikation auf $G \times U$ ist dann auch U eine G-Menge.

2.1.3 Definition. Das *Erzeugnis* einer Teilmenge $A \subseteq X$ ist definiert als

$$\langle A \rangle := GA := \{gx \mid g \in G, x \in A\}\,.$$

A heißt *Erzeugendensystem* von X, falls $X = \langle A \rangle$ ist.

Das Erzeugnis von $A \subseteq X$ ist stets eine G-Untermenge von X. Wir führen Transversalen analog zu Basen in Vektorräumen ein:

2.1.4 Definition. Zwei Elemente $x, y \in X$ heißen *abhängig*, falls es ein $g \in G$ gibt mit $gx = y$, ansonsten heißen sie *unabhängig*. Eine Teilmenge $A \subseteq X$ heißt *unabhängig*, falls sie nur triviale Abhängigkeiten enthält, d.h. falls für zwei $x, y \in A$ gilt:

$$x = gy \Longrightarrow x = y\,.$$

Ein unabhängiges Erzeugendensystem B von X heißt *Transversale*.

Transversalen von G-Mengen spielen eine ähnlich wichtige Rolle wie Basen von Vektorräumen.

2.1.5. Basisaustauschsatz. *Sind B und B' Transversalen von ${}_GX$, und ist $b \in B$, so gibt es genau ein $b' \in B'$, sodass $B \setminus \{b\} \cup \{b'\}$ wieder eine Transversale ist.*

2.1.6. Basisergänzungssatz. *Ist $A \subseteq X$ unabhängig, so lässt es sich zu einer Transversale fortsetzen.*

2.1.7. Dimensionssatz. *Gibt es eine endliche Transversale, so ist jede Transversale von ${}_GX$ endlich, und alle Transversalen besitzen dieselbe Mächtigkeit.*

Dies motiviert die Einführung der Dimension einer G-Menge.

2.1.8 Definition. Die *Dimension* von ${}_GX$ ist definiert als die Mächtigkeit einer (und damit jeder) Transversalen B von X (bzw. ∞, falls die Transversalen nicht endlich sind):

$$\dim(X) := {}_G\dim(X) := |B| .$$

Eine eindimensionale G-Untermenge $U \leq X$ heißt auch *G-Bahn*. Die Menge aller Bahnen einer G-Menge wird mit $G\backslash\backslash X$ bezeichnet. Ist X selbst eindimensional, so heißt X *transitiv*.

Das Cauchy-Frobenius Lemma liefert mit dieser Begriffsbildung eine Formel zur Bestimmung der Dimension von X:

2.1.9. Cauchy-Frobenius Lemma. *Sind G und X endlich, so ist die Dimension von ${}_GX$ gleich der mittleren Fixpunktzahl:*

$$\dim(X) = \frac{1}{|G|} \sum_{g \in G} |X_g| ,$$

wobei $X_g := \{x \in X \mid gx = x\}$ die Menge der Fixpunkte von g bezeichnet.

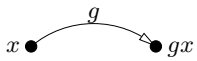
Beweis: Siehe [Ker99, Lemma 2.1.1]. □

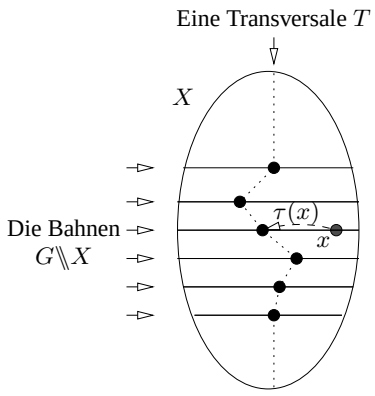
2.1.10 Beispiel. Jeder $\mathbb{K}$-Vektorraum V bildet bzgl. der multiplikativen Gruppe $\mathbb{K}^*$ eine $\mathbb{K}^*$-Menge. Die Dimension von V als $\mathbb{K}^*$-Menge entspricht aber nicht der Dimension des Vektorraums. Vielmehr ist sie gleich der Anzahl der eindimensionalen Unterräume + 1. (Der Nullvektor bildet eine eigene Bahn!) Insbesondere ist bereits der $\mathbb{R}^2$ eine ∞-dimensionale $\mathbb{R}^*$-Menge.

Während die Dimensionen eines Vektorraums anschaulich „senkrecht“ aufeinander stehen, und ein Element des Vektorraums sich gleichzeitig in mehrere Dimensionen erstrecken kann, liegen die Dimensionen einer G-Menge sozusagen „nebeneinander“. Bei endlichen Vektorräumen vervielfacht sich mit jeder neuen Dimension die Anzahl der Elemente. Bei endlichen G-Mengen hingegen kommen höchstens $|G|$ neue Elemente hinzu. Die G-Dimension ist sozusagen „additiv“, im Gegensatz zur „multiplikativen“ Dimension des Vektorraums.

Es ist noch eine andere gebräuchliche Sichtweise von G-Mengen hilfreich:

Man kann sich Gruppenoperationen auch als einen gerichteten Graphen (mit Schleifen und Mehrfachkanten) vorstellen: Der *Cayley-Action-Graph (CAG)* hat X als Menge von Knoten und für jedes $g \in G$, $x \in X$ führt eine Kante von x nach gx. Die Kanten werden mit den entsprechenden g beschriftet. Dabei kann man sich auch auf einen Teilgraphen *CAG*(E) beschränken, welcher nur Kanten enthält, die aus den $g \in E$ eines Erzeugendensystems E von G hervorgehen.





Zwei Elemente x, x' sind genau dann abhängig, wenn ein Pfad von x nach x' führt. Man erkennt die eindimensionalen Unterräume, die Bahnen, als die Zusammenhangskomponenten wieder. Die Dimension ist also die Anzahl der Zusammenhangskomponenten.

Die Bahnen werden häufig schematisch als waagrechte Striche innerhalb von X dargestellt, eine Transversale ist dann ein Querschnitt durch alle Bahnen, d.h. ein Pfad „von oben nach unten“ quer durch die Menge.

Für G-Mengen von zentraler Bedeutung sind die Stabilisatoren:

2.1.11 Definition. Sei X eine G-Menge. Der *Stabilisator* von $x \in X$ ist die Untergruppe (!) aller $g \in G$, welche x fix lassen:

$$G_x := \{g \in G \mid gx = x\}$$

In der linearen Algebra spielt diese Definition keine Rolle, da abgesehen vom Nullvektor alle Vektoren trivialen Stabilisator haben.

2.1.12 Bemerkung. Den Stabilisator eines von x abhängigen Elements $x' := gx$ erhält man aus dem Stabilisator von x durch Konjugation:

$$G_{gx} = gG_xg^{-1}\,.$$

Mit Hilfe der Stabilisatoren können wir eine „Koordinatendarstellung“ der Elemente in X einführen.

2.1.13. Eindeutige Darstellung. *Ist B eine Transversale von ${}_GX$, so ist jedes $x \in X$ bis auf Linksmultiplikation mit Elementen aus dem Stabilisator G_x eindeutig darstellbar in der Form*

$$x = g_x \cdot b_x\,.$$

Das bedeutet, es gibt genau ein zu x abhängiges $b_x \in B$, und g_x ist aus einer eindeutigen Rechtsnebenklasse des Stabilisators G_x.

Beweis: Es ist $X = \langle B\rangle = \{gb \mid g \in G, b \in B\}$, also gibt es Darstellungen von obiger Form.

Da B unabhängig ist, ist b_x dabei eindeutig: Ist nämlich $x = g_x \cdot b_x = g'_x \cdot b'_x$, so folgt daraus $b_x = g_x^{-1}g'_xb'_x$, also sind b_x und b'_x abhängig. Es gibt aber in B nur triviale Abhängigkeiten, also ist $b_x = b'_x$.

Ist nun $x = g_xb_x = g'_xb_x$, so folgt $b_x = g_x^{-1}x$ und $b_x = g_x'^{-1}x$, durch Gleichsetzen ist also $x = g'_xg_x^{-1}x$, d.h. $g_0 := g'_xg_x^{-1}$ liegt im Stabilisator G_x. Damit geht $g'_x = g_0g_x$ aus g_x durch Linksmultiplikation mit einem Stabilisatorelement $g_0 \in G_x$ hervor. □

2.1.14 Folgerung. Sei X eine G-Menge mit Transversale B. Für $g, g' \in G$ und $b, b' \in B$ gilt:

$$gb = g'b' \Longrightarrow b = b' \text{ und } g^{-1}g' \in G_b\,.$$

Beweis: Aus der Eindeutigkeit der Darstellung von gb folgt $b = b'$. Weil $1b = g^{-1}g'1b$ ist, liegt $g^{-1}g'$ im Stabilisator von $1b$. □

Eine Abbildung $\tau : X \to G$, die jedem $x \in X$ ein $\tau(x) := g_x^{-1}$ mit $x = g_xb_x$ zuordnet, wird als *kanonisierende* (oder *fusionierende) Abbildung* bezeichnet. In obiger Abbildung ist ein $\tau(x)$ eingezeichnet, um zu verdeutlichen, dass $\tau(x)$ jedes x in ein Transversalenelement b_x überführt.

2.2 Homomorphismen

Wir betrachten Abbildungen, die mit der Gruppenoperation verträglich sind:

2.2.1 Definition. Es seien X, Y G-Mengen. Eine Abbildung $f : X \to Y$ heißt G-Homomorphismus, falls für alle $g \in G$, $x \in X$ gilt:

$$f(g \cdot x) = g \cdot f(x) .$$

Surjektive G-Homomorphismen heißen auch *G-Epimorphismen*, injektive *G-Monomorphismen* und bijektive *G-Isomorphismen*.

Zunächst stellen wir ein wichtiges notwendiges Kriterium für G-Homomorphie vor:

2.2.2 Bemerkung. Ist $f : X \to Y$ ein G-Homomorphismus, so gilt für die Stabilisatoren aller $x \in X$:

$$G_x \leq G_{f(x)} .$$

Beweis: Ist $gx = x$, so folgt $gf(x) = f(gx) = f(x)$. □

In der linearen Algebra genügt es, das Bild der Basiselemente einer linearen Abbildung zu kennen. Für G-Homomorphismen gilt analog:

2.2.3. Eindeutige Fortsetzung *Seien X, Y G-Mengen, B eine Transversale von X, und $f : B \to Y$ eine Abbildung mit $G_b \leq G_{f(b)}$ für alle $b \in B$.*
Dann ist f eindeutig zu einem G-Homomorphismus $\bar{f}$ fortsetzbar,

$$\bar{f} : X \to Y, x \mapsto g_x f(b_x),$$

wobei $x = g_x b_x$ eine Darstellung von x bzgl. der Transversale B sei.

Beweis:
1. Die Abbildung $\bar{f}$ ist wohldefiniert: Da die Darstellung $x = g_x b_x$ eindeutig ist bis auf Linksmultiplikation mit Stabilisatorelementen, müssen wir nur zeigen, dass für eine weitere Koordinatendarstellung $x = g_0 g_x b_x$ mit $g_0 \in G_x$ sich nichts am Wert der Definition für $\bar{f}(x)$ ändert. Es ist also zu zeigen, dass g_0 im Stabilisator von $g_x f(b_x)$ liegt.
Nach Folgerung 2.1.14 liegt $g_x^{-1} g_0 g_x$ im Stabilisator von b_x, nach Voraussetzung also auch im Stabilisator von $f(b_x)$. Es gilt demnach $g_x^{-1} g_0 g_x \cdot f(b_x) = f(b_x)$. Linksmultiplikation mit g_x liefert die gewünschte Aussage.

2. Sei $g \in G$ und $x \in X$ beliebig. Aus einer Darstellung $x = g_x b_x$ erhalten wir eine Darstellung von gx durch $gx = gg_x\, b_x$. Damit ist $\bar{f}(gx)$ nach Definition gleich $gg_x f(b_x)$. Hier können wir die Definition von $\bar{f}(x)$ wieder einsetzen und erhalten $\bar{f}(gx) = g\bar{f}(x)$, also ist $\bar{f}$ ein G-Homomorphismus.

3. Die Eindeutigkeit sehen wir wie folgt: Ist $\psi : X \to Y$ eine weitere G-homomorphe Fortsetzung von f, so gilt für jedes $x = g_x b_x \in X$:

$$\psi(x) = \psi(g_x b_x) = g_x \psi(b_x) = g_x \bar{f}(b_x) = \bar{f}(g_x b_x) = \bar{f}(x)\,.$$

□

Das Bild eines G-Homomorphismus ist eine G-Untermenge:

2.2.4 Satz. *Seien X, Y G-Mengen und $f : X \to Y$ ein G-Homomorphismus. Dann ist das Bild $f(X)$ eine G-Untermenge von Y.*

Beweis: Ist $y \in f(X)$, so gibt es ein $x \in X$ mit $f(x) = y$. Damit ist aber auch für jedes $g \in G$: $gy = gf(x) = f(gx) \in f(X)$. Also ist $f(X)$ abgeschlossen unter Multiplikation mit Gruppenelementen. □

Im folgenden Abschnitt werden die Voraussetzungen geschaffen, um ein Analogon zum Kern von G-Homomorphismen zu definieren. Damit können wir in Abschnitt 2.4 schließlich einen entsprechenden Homomorphiesatz formulieren.

Wir möchten zuvor an dieser Stelle noch erwähnen, dass die Begriffe Monomorphismus und Epimorphismus durchaus mit den üblichen Definitionen auf Kategorien übereinstimmen:

2.2.5 Satz. *Sei $f : X \to Y$ ein G-Homomorphismus. Es gilt:*

1. *f surjektiv $\Longleftrightarrow$ f rechtskürzbar, d.h.* $\forall$ *G-Homom.* $f_1', f_2' : Y \to Z$:
$$f_1' \circ f = f_2' \circ f \Rightarrow f_1' = f_2'$$

2. *f injektiv $\Longleftrightarrow$ f linkskürzbar, d.h.* $\forall$ *G-Homom.* $f_1', f_2' : Z \to X$:
$$f \circ f_1' = f \circ f_2' \Rightarrow f_1' = f_2'$$

Beweis: Während die Beweisrichtungen „⇒“ direkt aus dem entsprechenden Satz für die Kategorie der Mengen folgt, sind die Rückrichtungen keinesfalls trivial. Da sich hier Kürzbarkeit ausschließlich auf G-Homomorphismen bezieht, sind die rechtsseitigen Eigenschaften zunächst schwächer als Rechts-/Linkskürzbarkeit für

Abbildungen zwischen Mengen. Dass die obigen Eigenschaften dennoch hinreichend für die Surjektivität bzw. Injektivität sind, muss also explizit gezeigt werden:
1. „$\Leftarrow$“: Das Bild von f ist eine G-Untermenge von Y, eine Transversale B von $f(X)$ lässt sich also zu einer Transversalen $\overline{B}$ von Y fortsetzen. Betrachten wir die G-Menge

$$Z := \{z_0, z_1\}, \text{ mit } gz_0 = z_0 \text{ und } gz_1 = z_1 \text{ für alle } g \in G\,,$$

und die folgenden beiden G-Homomorphismen von Y nach Z, welche durch die Bilder der Transversalenelemente $b \in \overline{B}$ definiert sind:

$$f_1' : b \mapsto z_0$$
$$f_2' : b \mapsto \begin{cases} z_0 & \text{falls } b \in f(X) \\ z_1 & \text{sonst}\,. \end{cases}$$

Da die Stabilisatoren der Bilder $G_{z_0} = G_{z_1} = G$ maximal sind, sind die Voraussetzungen von Satz 2.2.3 für beide Abbildungen erfüllt. Durch die Vorschrift sind also tatsächlich G-Homomorphismen definiert.
Auf dem Bild $f(X)$ von f sind die beiden Abbildungen identisch, es gilt also $f_1' \circ f = f_2' \circ f$. Nach Voraussetzung ist f rechtskürzbar, es folgt $f_1' = f_2'$. Also muss $Y \setminus f(X) = \emptyset$ sein, d.h. f ist surjektiv.

2. „$\Leftarrow$“: Sei $x_1, x_2 \in X$ mit $f(x_1) = f(x_2)$. Wir zeigen, dass $x_1 = x_2$ folgt. Betrachte dazu die Abbildungen $f_1', f_2' : G \to X$, welche auf der Transversale $B := (1)$ der G-Menge $Z := G$ definiert sind:

$$f_1' : G \to X, 1 \mapsto x_1$$
$$f_2' : G \to X, 1 \mapsto x_2$$

g_1 und g_2 sind nach Satz 2.2.3 G-Homomorphismen, denn der Stabilisator von $1 \in G$ ist trivial: $G_1 = \{1\}$. Die Voraussetzung des Satzes ist demnach für beide Abbildungen erfüllt. Es gilt also $f \circ f_1'(g) = f \circ f_2'(g)$ für alle $g \in G$. Aufgrund der Linkskürzbarkeit folgt $f_1' = f_2'$, also $x_1 = x_2$. □

2.3 Blöcke, Blocktransversalen und Faktormengen

Ist V ein $\mathbb{K}$-Vektorraum, und $\mathbb{U} \leq \mathbb{K}$ ein Unterkörper, so wird aus V in natürlicher Weise auch ein $\mathbb{U}$-Vektorraum höherer Dimension. Ähnliches gilt auch bei G--

Mengen: Ist $G' \leq G$ eine Untergruppe, so ist jede G-Menge auch eine G'-Menge. Dabei zerfallen die G-Bahnen im Allgemeinen in mehrere G'-Bahnen.

Andererseits ist aber nicht jede G'-Untermenge von X eine G-Untermenge. Zum Beispiel ist für $G' = \{1\}$ trivialerweise jede Teilmenge $A \subseteq G$ eine $\{1\}$-Menge. Die größte Untergruppe $G' \leq G$, sodass $A \subseteq X$ eine G'-Menge ist, ist genau der Mengenstabilisator von A:

2.3.1 Definition. Sei X G-Menge, und $A \subseteq X$ Teilmenge. Dann heißt folgende Untergruppe (!) von G der *Mengenstabilisator* (oder *mengenweise Stabilisator*) von X:

$$G_A := \{g \in G \mid A = gA\}\,.$$

G_A kann als Verallgemeinerung des weiter oben eingeführten Stabilisators für ein Element gesehen werden. Es gilt nämlich $G_x = G_{\{x\}}$.

2.3.2 Lemma. *Sei X eine G-Menge und $A \subseteq X$. Dann ist G_A die größte Untergruppe von G, sodass die Operation eingeschränkt auf G_A abgeschlossen in A ist. Insbesondere ist A eine G_A-Menge und es gilt:*

$$A \text{ ist } G'\text{-Menge} \iff G' \leq G_A\,.$$

Beweis: Die Menge aller g, für welche die Multiplikation auf A abgeschlossen ist, also

$$M := \{g \in G \mid gA \subseteq A\}\,,$$

kann für unendliche Gruppen unter Umständen größer sein als G_A. Ist allerdings für ein $g \in G$ das Bild $gA \subsetneq A$, so gibt es ein $x \in A$, sodass $g^{-1}x \notin A$. Also ist g^{-1} dann nicht in obiger Menge M enthalten und M damit keine Gruppe. Entfernt man nun aus M all diese problematischen g, so bleibt G_A übrig, eine Gruppe. □

Innerhalb jeder Teilmenge $A \subseteq X$ können wir gleichzeitig zwei Gruppenoperationen berücksichtigen: Einerseits ist A eine G_A-Menge, andererseits gehören die Elemente auch zu X, einer G-Menge. Hervorzuheben sind also Teilmengen A, für welche die Abhängigkeitsbegriffe sowie die Stabilisatoren bzgl. der beiden Gruppenoperationen zusammenfallen:

2.3.3 Definition. Sei X eine G-Menge. Eine Teilmenge $A \subseteq X$ heißt *Block* oder *Imprimitivitätsbereich*, falls gilt:

1. G-abhängige Elemente aus A sind auch G_A-abhängig, d.h. für $x, x' \in A$ gilt:

$$\exists g \in G : x' = gx \Longrightarrow \exists \tilde{g} \in G_A : x' = \tilde{g}x\,.$$

2. Die Stabilisatoren G_x und $(G_A)_x$ sind für $x \in A$ gleich, d.h. es gilt:

$$\forall x \in A : G_x \leq G_A \,.$$

Beispielsweise ist jede G-Untermenge $A \leq X$ ein Block. Andererseits ist auch die leere Menge $\emptyset$ sowie jedes $\{x\}$, $x \in X$ ein Block. Die genannten Beispiele werden auch als die *trivialen* Blöcke bezeichnet.

2.3.4 Satz. *Sei X eine G-Menge und $A \subseteq X$ eine Teilmenge. Es sind äquivalent:*

1. A ist Block.
2. Für alle $x, x' \in A$, $g \in G$ gilt:
$$x' = gx \Longrightarrow g \in G_A$$
3. Für jedes $g \in G$ gilt:
$$A \cap gA = \emptyset \text{ oder } A = gA \,.$$

Beweis:
„1⇒2“. Für $x' = gx$ gibt es zunächst ein $\tilde{g} \in G_A$ mit $x' = \tilde{g}x$. Die Menge $\{x\}$ lässt sich in G aber zu einer Transversale B von X fortsetzen, wegen der Eindeutigkeit der Darstellung von x' bzgl. B ist dann aber $\tilde{g}^{-1}g$ im Stabilisator G_x, und wegen $G_x \leq G_A$ folgt auch $g \in G_A$.

„2⇒3“. Sei $A \cap gA$ nicht leer, etwa $x \in A \cap gA$. Dann gibt es ein $x' \in A$ mit $x = gx'$. Nach Voraussetzung ist also $g \in G_A$, d.h. $A = gA$.

„3⇒1“. 1. Sei $x' = gx$ für $x, x' \in A$ und $g \in G$. Dann ist $x' \in A \cap gA$, der Schnitt $A \cap gA$ ist also nicht leer. Also folgt $A = gA$, das heißt $\tilde{g} := g \in G_A$.
2. Sei $x \in A$. Für $g \in G_x$ gilt $x = gx$, d.h. $x \in A \cap gA$, also folgt wieder $A = gA$ und damit $g \in G_A$. □

2.3.5 Folgerung. Sei X eine G-Menge und $A \subseteq X$ Block. Dann gilt:

Sei $A' \subseteq A$.

1. A' ist G_A-Block von $A \iff A'$ ist G-Block von X
2. A' ist G_A-Erz.system von $A \iff A'$ ist G-Erz.system von GA

Sei $x, x' \in A$.

3. x, x' sind G_A-abhängig $\iff x, x'$ sind G-abhängig

Sei $B \subseteq A$.

4. B ist G_A-unabhängig $\iff B$ ist G-unabhängig
5. B ist G_A-Transversale von $A \iff B$ ist G-Transversale von GA

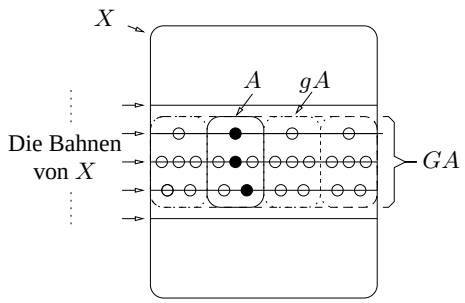


Abbildung 2.1: Ein Block A und die Partitionierung des G-Erzeugnisses GA.

Sei $B \subseteq A$ Transversale und $x \in A$.

6. $x = g_x b_x$ ist G_A-Darstellung in $A \iff x = g_x b_x$ ist G-Darstellung in GA

7. Ist $f : A \to Y$ ein G_A-Homomorphismus, so gibt es eine eindeutige G-homomorphe Fortsetzung
$$\bar{f} : GA \to Y\,.$$
Ist $f : X \to Y$ ein G-Homomorphismus, so ist die Einschränkung ein G_A-Homomorphismus:
$$f \downarrow_A : A \to Y\,.$$

Weiter ergibt sich aus der dritten äquivalenten Charakterisierung von Blöcken:

8. Ist A ein Block, so ist für jedes $g \in G$ auch gA ein Block
9. Die Menge $\{gA \mid g \in G\}$ ist eine Mengenpartition von GA. Wir erhalten also eine Äquivalenzrelation auf GA:
$$x \sim x' :\iff \exists g \in G : x, x' \in gA$$

Zur Visualisierung der Punkte 8. und 9. siehe auch Abbildung 2.1. Wir sprechen im Folgenden schlicht von Transversalen eines Blockes bzw. von der Dimension eines Blockes:
$$\dim(A) := {}_{G_A}\dim(A) = {}_G\dim(GA)\,.$$

2.3.6 Satz. *Sei X eine G-Menge. Der Schnitt zweier Blöcke $A, B \subseteq X$ ist wieder ein Block. Ist weiter $A \cap B \neq \emptyset$, so gilt für den mengenweisen Stabilisator*

$$G_{A\cap B} = G_A \cap G_B \,.$$

Beweis: Sei $C := A \cap B$. Wir zeigen den ersten Teil der Behauptung mit Kriterium 2.3.4(3): Für $g \in G$ gilt: $C \cap gC = (A\cap B)\cap(gA\cap gB) = (A\cap gA)\cap(B\cap gB)$. Letzteres ist wegen Kriterium 2.3.4(3), angewandt auf die Blöcke A und B, entweder leer oder gleich $A \cap B = C$.
Zur Aussage bzgl. des mengenweisen Stabilisators: Für $g \in G_A \cap G_B$ gilt $gC = gA \cap gB = A \cap B = C$, also ist $g \in G_C$, d.h. $G_A \cap G_B \leq G_C$. Ist $C \neq \emptyset$, und $x \in C$, so folgt aus $g \in G_C$ weiter: $gx \in C \Longrightarrow gx \in A$ und $gx \in B$ $\underset{\text{2.3.4(2)}}{\Longrightarrow}$ $g \in G_A$ und $g \in G_B \Longrightarrow g \in G_A \cap G_B$, d.h. $G_C \leq G_A \cap G_B$. □

Wir definieren auch für eine Menge von Blöcken Abhängigkeit und Erzeugnis, was uns zu den Begriff der Blocktransversale führt. Blocktransversalen werden wir als Analogon (bzw. als Verallgemeinerung) zum Kern einer linearen Abbildung wiedererkennen.

2.3.7 Definition. Zwei Blöcke A und A' heißen *abhängig*, falls sie abhängige Elemente $x \in A$, $x' \in A'$ enthalten. Eine Menge $\mathcal{B}$ von Blöcken heißt *unabhängig*, falls sie nur triviale Abhängigkeiten enthält, d.h. falls für $x_1 \in A_1 \in \mathcal{B}$, $x_2 \in A_2 \in \mathcal{B}$ und $g \in G$ gilt:

$$x_1 = gx_2 \Longrightarrow A_1 = A_2$$

Eine Menge $\mathcal{B}$ von Blöcken heißt *erzeugende Blockmenge* von X, falls die Vereinigung der Blöcke ganz X erzeugt.

$$X = G(\bigcup_{A\in\mathcal{B}} A)$$

Schließlich heißt eine Menge $\mathcal{B}$ von Blöcken *Blocktransversale*, falls sie eine unabhängige erzeugende Blockmenge ist.

2.3.8 Satz. *Sei X G-Menge und $\mathcal{B}$ eine Blocktransversale. Dann erhält man eine Transversale B von X als disjunkte Vereinigung von G_A-Transversalen B_A der Blöcke $A \in \mathcal{B}$:*

$$B = \dot{\bigcup_{A\in\mathcal{B}}} B_A \,.$$

Insbesondere gilt im endlichdimensionalen Fall:

$$\dim(X) = \sum_{A\in\mathcal{B}} \dim(A) \,.$$

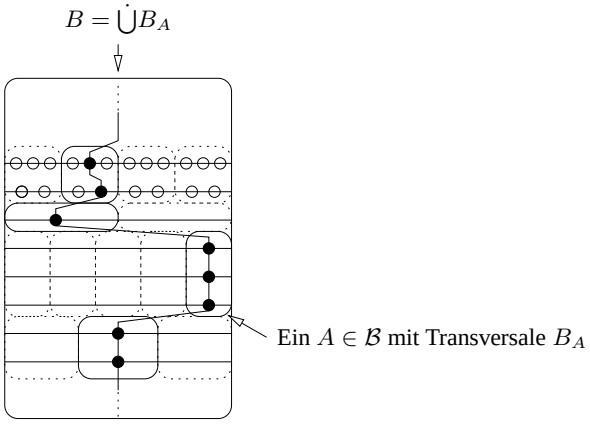


Abbildung 2.2: Eine Blocktransversale $\mathcal{B}$.

Beweis: Da die Vereinigung der Blöcke von $\mathcal{B}$ ein G-Erzeugendensystem bildet, ist auch B ein G-Erzeugendensystem. Sind $x \in B_A$ und $x' \in B_{A'}$ abhängig, so sind die Blöcke A und A' abhängig, also $A = A'$. Sind aber $x, x' \in B_A$ abhängig, so sind sie auch G_A-abhängig (A ist Block!), es folgt $x = x'$. B ist also unabhängiges Erzeugendensystem. □

Eine Blocktransversale ist in Abbildung 2.2 schematisiert. Die Mengenpartitionen der Erzeugnisse der Blöcke vereinigen sich in natürlicher Weise zu einer Mengenpartition auf ganz X. Wir erhalten also zu jeder Blocktransversale eine Äquivalenzrelation auf X:

2.3.9 Lemma. *Sei X eine G-Menge und $\mathcal{B}$ eine Blocktransversale. Dann induziert $\mathcal{B}$ wie folgt eine Äquivalenzrelation auf X:*

$$x \sim_{\mathcal{B}} x' :\Longleftrightarrow \exists A \in \mathcal{B}, g \in G : x, x' \in gA\,.$$

Die Blöcke aus $\mathcal{B}$ tauchen dabei wieder als Äquivalenzklassen auf, d.h. $\mathcal{B} \subseteq X/\sim_{\mathcal{B}}$.

Beweis: Da $\mathcal{B}$ unabhängig ist, sind die Erzeugnisse zweier Blöcke $A, A' \in \mathcal{B}$ disjunkt: $\langle A\rangle \cap \langle A'\rangle = \emptyset$. Da $\mathcal{B}$ ein erzeugendes Blocksystem ist, ist die Vereinigung der Erzeugnisse $\dot{\bigcup}_{A\in\mathcal{B}}\langle A\rangle = X$. Mit 2.3.5(9) folgt die Behauptung. □

2.3.10 Satz. *Sei X eine G-Menge, und $\sim \subseteq X \times X$ eine Äquivalenzrelation auf X. Dann sind äquivalent:*

1. *Es gibt eine Blocktransversale $\mathcal{B}$, sodass $\sim$ der induzierten Äquivalenzrelation $\sim_{\mathcal{B}}$ zu $\mathcal{B}$ entspricht.*
2. *Die Äquivalenzrelation ist mit der Multiplikation verträglich:*

$$x \sim x' \Leftrightarrow gx \sim gx' .$$

3. *Die Menge $X/\sim$ der Äquivalenzklassen von X ist eine G-Menge vermöge*

$$g \cdot [x]_\sim := [gx]_\sim .$$

Beweis:

„2⇔3“. Die in (3) angegebene Multiplikation ist genau dann wohldefiniert, wenn (2) gilt.

„1⇒2,3“. Ist $x \sim x'$, so gibt es ein $\tilde{g} \in G$, $A \in \mathcal{B}$ mit $x, x' \in \tilde{g}A$. Damit liegen $gx, gx' \in g\tilde{g}A$, also $gx \sim gx'$. Ist $gx \sim gx'$, so sind mit dem gleichen Argument auch $x = g^{-1}gx \sim g^{-1}gx' = x'$.

„2,3⇒1“. Ist $A := [x]_\sim$ eine Äquivalenzklasse, so ist nach Voraussetzung auch gA eine. Also ist der Schnitt von A mit gA entweder leer oder ganz A (Schnitt zweier Äquivalenzklassen). Damit ist A Block.
Weiter sind die zu A abhängigen Äquivalenzklassen (im Sinne von Definition 2.3.7) genau diejenigen der Form gA. Wählen wir also eine maximale unabhängige Teilmenge von Blöcken aus $X/\sim$ aus, so erhalten wir eine Blocktransversale. □

2.3.11 Definition. Sei X eine G-Menge. Eine Äquivalenzrelation $\sim$ auf X heißt *G-Äquivalenzrelation*, falls sie eine der äquivalenten Bedingungen von Satz 2.3.10 erfüllt. Ist $\mathcal{B}$ eine Blocktransversale mit $\sim_{\mathcal{B}} = \sim$, so heißt die G-Menge der Äquivalenzklassen auch *G-Faktormenge* von X durch $\mathcal{B}$:

$$X/\mathcal{B} := X/\sim_{\mathcal{B}}$$

2.3.12 Satz. *Sei X eine G-Menge und $\mathcal{B}$ eine Blocktransversale. Dann ist $\mathcal{B}$ eine Transversale der Faktormenge $X/\mathcal{B}$.*

Beweis: Jedes $A \in \mathcal{B}$ ist eine eigene Äquivalenzklasse aus $X/\mathcal{B}$, d.h. $\mathcal{B} \subseteq X/\mathcal{B}$. Weiter ist $\mathcal{B}$ unabhängiges Erzeugendensystem, weil es unabhängige erzeugende Blockmenge ist. □

Die Dimensionsformel 2.3.8 wird dadurch zum Pendant der Tatsache, dass in der linearen Algebra $\dim(V) = \dim(U) \cdot \dim(V/U)$ ist.

Die Situation ist insgesamt zwar nicht ganz so schön wie in der linearen Algebra, hat aber immer noch bemerkenswerte Eigenschaften:

Allgemein sind Faktorstrukturen die Äquivalenzklassen einer mit der Struktur verträglichen Äquivalenzrelation.

In der Kategorie der Mengen ist jede Äquivalenzrelation mit der Struktur verträglich. (Die Mengen haben keine zusätzliche Struktur.) Demzufolge kann man die Faktorstruktur nur durch Angabe aller Äquivalenzklassen beschreiben.

Im Fall von Vektorräumen haben die mit der Struktur verträglichen Äquivalenzrelationen hingegen besondere Äquivalenzklassen. Insbesondere ist eine der Klassen ein Unterraum. Die Bedeutung der Faktorräume in der linearen Algebra ist u.a. dadurch begründet, dass die Kenntnis einer einzigen Äquivalenzklasse ausreicht, um die gesamte Struktur zu beschreiben.

Die Situation bei G-Mengen liegt nun irgendwo dazwischen: Mit der Struktur verträgliche Äquivalenzrelationen haben Blöcke als Äquivalenzklassen. Es reicht aber nicht aus, eine einzige Äquivalenzklasse (einen Block) zu kennen. Dennoch ist ein deutlicher Gewinn gegenüber strukturlosen Mengen zu erkennen, da eine Blocktransversale die gesamte Struktur der Faktormenge beschreibt.

2.4 Der Homomorphiesatz

Wir leiten im Folgenden den Kern eines G-Homomorphismus f (bzgl. einer Transversalen C von Y) her. Dieser ist eine Blocktransversale.

2.4.1 Lemma. *Sei $f : X \to Y$ ein G-Homomorphismus. Dann bilden die Urbilder $f^{-1}(y)$, $y \in Y$ Blöcke in X. Liegt y im Bild $f(X)$, so ist der Stabilisator G_y gleich dem mengenweisen Stabilisator des Urbilds von y:*

$$G_y = G_{f^{-1}(y)} .$$

Beweis: Liegt y nicht im Bild $f(X)$, so ist das Urbild leer, also trivialerweise ein Block von X. Sei also $y \in f(X)$ und $A := f^{-1}(y)$.

Wir zeigen zunächst den zweiten Teil: Sei $g \in G_y$. Für jedes $x \in A$ ist $f(gx) = gf(x) = gy = y$, also $gA \subseteq A$. Demnach ist für die Gruppe G_y die Multiplikation

auf A abgeschlossen, Nach 2.3.2 gilt also $G_y \subseteq G_A$.
Sei andererseits $g \in G_A$. Da $y \in f(X)$ ist, gibt es ein $x \in A$. Damit ist auch $gx \in A$. Also ist $y = f(gx) = gf(x) = gy$, d.h. $g \in G_y$. Wir haben somit $G_y = G_A$ gezeigt.

Dass A ein Block ist, sieht man mit dem zweiten Kriterium aus 2.3.4: Seien $x, x' \in A$, $g \in G$ mit $x' = gx$. Es gilt $gy = gf(x) = f(gx) = f(x') = y$, also ist g im Stabilisator G_y und $G_y = G_A$.

□

2.4.2 Lemma. *Sei $f : X \to Y$ ein G-Homomorphismus sowie C eine Transversale von Y. Dann bilden die Urbilder der $c \in C$ eine Blocktransversale in X.*

Beweis: Die Blöcke $f^{-1}(c)$, $c \in C$ sind unabhängig: Gilt für $c, c' \in C$ und für $x \in f^{-1}(c)$ und $x' \in f^{-1}(c')$ die Abhängigkeit $x = gx'$, so folgt $c = f(x) = f(gx') = gf(x') = gc'$. Da die Transversale C aber nur triviale Abhängigkeiten enthält, folgt $c = c'$.

Die Vereinigung der Blöcke $f^{-1}(c)$, $c \in C$ erzeugt ganz X: Ist $x \in X$, so lässt sich $y := f(x)$ in der Form $y = g_y c_y$ darstellen. Damit ist $f(g_y^{-1}x) = g_y^{-1}f(x) = g_y^{-1}y = c_y$, also ist $g_y^{-1}x$ im Block $A := f^{-1}(c_y)$, und x liegt im Erzeugnis GA.

□

2.4.3 Definition. Sei $f : X \to Y$ ein G-Homomorphismus sowie C eine Transversale von Y. Dann ist der *Kern* von f bzgl. C die Blocktransversale der Urbilder der Transversalenelemente $c \in C$:

$$\ker_C(f) := \{f^{-1}(c) \mid c \in C\}\,.$$

2.4.4. Homomorphiesatz. *([Lau93]) Seien X und Y G-Mengen, C eine Transversale von Y sowie $f : X \to Y$ ein G-Homomorphismus. Dann gilt:*

$$X/\ker_C(f) \cong f(X)\,.$$

Sei weiter für jedes $c \in C$ eine (G_c-) Transversale B_c des Urbildes $f^{-1}(c)$ gegeben. Dann erhält man eine Transversale B von X als disjunkte Vereinigung der B_c:

$$B = \dot{\bigcup_{c \in C}} B_c\,.$$

Im endlichdimensionalen Fall gilt insbesondere:

$$\dim(X) = \sum_{c \in C} \dim(f^{-1}(c))$$

Beweis: Nach Lemma 2.4.2 ist $\ker_C(f)$ eine Blocktransversale, gemäß Satz 2.3.10 ist also die Faktormenge $X/\ker_C(f)$ eine G-Menge. Das Bild $f(X)$ ist hingegen nach Satz 2.2.4 eine G-Menge. Betrachte nun die Abbildung

$$\bar{f} : X/\ker_C(f) \to f(X), [x]_\sim \mapsto f(x) :$$

$\bar{f}$ ist ein wohldefinierter G-Homomorphismus, da für $x \sim x'$ nach Definition ein $A \in \ker_C(f)$, $g \in G$ existiert mit $gx, gx' \in A$. Also gilt $f(gx) = f(gx')$. Da f G-Homomorphismus ist, folgt daraus $f(x) = f(x')$.

Die Surjektivität ist klar, da wir als Bildbereich $f(X) = \bar{f}(X/\ker(f))$ betrachten. Zur Injektivität: Ist $y := \bar{f}([x]_\sim) = \bar{f}([x']_\sim)$, und ist $y = g_y c_y$, so liegen $g_y^{-1}x$ und $g_y^{-1}x'$ im gleichen Urbild $A := f^{-1}(c_y)$, also ist $x \sim x'$. Die weiteren Aussagen folgen direkt aus 2.3.8. □

2.4.5 Folgerung. *Seien X, Y, f wie in 2.4.4.* Zwei Elemente $x, x' \in X$ sind genau dann G-abhängig, wenn gilt:

1. $f(x)$ und $f(x')$ sind G-abhängig.
2. Für ein entsprechendes (beliebiges) $g \in G$ mit $f(x) = gf(x')$ gilt: x und gx' sind $G_{f(x)}$-abhängig.

Die Bedeutung dieser Aussage ist hauptsächlich in folgender Situation zu sehen: Ist bereits bekannt, dass zwei Elemente x, x' unter f abhängige Bilder haben, so kann man die G-Abhängigkeit von x und x' auf eine äquivalente $G_{f(x)}$--Abhängigkeit zurückführen.

Beweis: „⇒“: 1. Nach Voraussetzung gibt es ein g_0 mit $x = g_0x'$, also gilt $f(x) = f(g_0x') = g_0f(x')$.
2. Sei $g \in G$ ein (weiteres) Gruppenelement mit $f(x) = gf(x')$ bzw. $f(x) = f(gx')$, d.h. $gx' \in f^{-1}(x)$. Nach 2.4.1 liegen x und gx' dann in einem Block mit mengenweisem Stabilisator $G_{f(x)}$. Nach 2.3.5(3) sind x und gx' genau dann $G_{f(x)}$-abhängig, wenn sie G-abhängig sind. Letzteres ist aber nach Voraussetzung der Fall.

„⇐“ ist klar, da es ja nach (2) ein $g_2 \in G_{f(x)}$ gibt, sodass $x = g_2gx'$ ist. □

Der Homomorphiesatz ist fundamental für die Entwicklung effizienter Kanonisierungs- und Konstruktionsalgorithmen. Wir gehen darauf in Kapitel 4 ein.

3 Permutationsgruppen

Zur Kanonisierung und Konstruktion diskreter Strukturen (Kapitel 4) benötigen wir die Konzepte Stabilisatorkette [Sim71] und Labelled Branching [Jer86, CF91], um die operierende Gruppe G einer Gruppenoperation ${}_GX$ zu verwalten. Diese Datenstrukturen werden hier kurz eingeführt, und es wird ein Überblick über die in diesem Zusammenhang wichtigsten Algorithmen gegeben. Ansonsten sei auf die Literatur verwiesen, z.B. [Lau93, HEO05]. Schließlich diskutieren wir einen Algorithmus zum lexikographischen Durchlaufen einer Transversale von G/H, $H \leq G \leq S_n$ bei gegebenem vollständigen Labelled Branching zu H. Donald E. Knuth [Knu79] formuliert allgemeiner einen Algorithmus zum Durchlaufen aller topologischen Anordnungen einer partiellen Ordnung, der sich direkt auf die beschriebene Problemstellung anwenden lässt. Er stellt jedoch eine starke Bedingung an die gewählte Basis $\vec{b}$. Hier werden Überlegungen angestellt, die erlauben, diese Bedingung zu lockern.

3.1 Stabilisatorketten

Es sei X eine (relativ kleine) G-Menge, mit $n := |X|$. Mit X ist auch die Menge X^k_{inj} aller injektiven k-Tupel (für $k \leq n$) eine G-Menge:

$$X^k_{\text{inj}} := \{(x_0, \ldots, x_{k-1}) \in X^k \mid \forall\, i \neq j \in k : x_i \neq x_j\}\,,$$

vermöge

$$g \cdot (x_0, \ldots, x_{k-1}) := (gx_0, \ldots, gx_{k-1})\,.$$

Der *punktweise Stabilisator* von k Elementen $x_0, \ldots, x_{k-1} \in X$ ist definiert als der Stabilisator der Folge $(x_0, \ldots, x_{k-1}) \in X^k_{\text{inj}}$. (Dabei ist die Reihenfolge der Elemente beliebig.) Es gilt:

$$G_{(x_0,\ldots,x_{k-1})} = G_{x_0} \cap \ldots \cap G_{x_{k-1}}.$$

Ein Tupel $(x_0, \ldots, x_{k-1}) \in X^k_{\text{inj}}$, $k \in n$, mit trivialem Stabilisator $G_{(x_0,\ldots,x_{k-1})} = \{\mathrm{id}\}$ heißt *Basis*[1] von X von der Länge k. Eine Gruppenoperation ${}_GX$ heißt *treu*, falls es in X Basen gibt. Wir beschränken uns im Folgenden auf treue Gruppenoperationen. Da der punktweise Stabilisator $G_{(x_0,\ldots)}$ aller Elemente aus X ein Normalteiler in G ist, kann man jede Gruppenoperation ${}_GX$ durch eine treue Gruppenoperation ersetzen, indem man als operierende Gruppe $G/G_{(x_0,\ldots)}$ wählt.

3.1.1. Bezeichnung. Sei X eine treue G-Menge mit Basis $\vec{b} := (x_0, \ldots, x_{k-1})$. Dann bezeichne $G^{(i)}$, $0 \leq i \leq k$, den punktweisen Stabilisator der ersten i Elemente der Basis $\vec{b}$.

$$G^{(i)} := G_{(x_0,\ldots,x_{i-1})}\,.$$

Dabei ist $G^{(0)} = G$, $G^{(k)} = \{\mathrm{id}\}$, und es gilt $G^{(i)} \geq G^{(i+1)}$, für $i \in k$. Die Kette der Untergruppen $G = G^{(0)} \geq G^{(1)} \geq \cdots \geq G^{(k-1)} \geq G^{(k)} = \{\mathrm{id}\}$ heißt *Stabilisatorkette* von G zur Basis $\vec{b}$. Ein Satz von Transversalen der Linksnebenklassen

$$T^{(i)} \in \mathcal{T}(G^{(i)}/G^{(i+1)}),\ i \in k$$

bildet ein *starkes Erzeugendensystem*, das heißt die einzelnen $G^{(i)}$ lassen sich daraus rekonstruieren:

$$G^{(i)} = \langle \bigcup_{i \leq j < k} T^{(j)} \rangle.$$

Wir vereinbaren, dass die Transversalen stets mit $\mathrm{id} \in T^{(i)}$ gewählt seien. Es gilt:

$$G = T^{(0)} \cdots T^{(k-1)}\,,$$

d.h. jedes $g \in G$ ist in der Form $g = t_0 \cdots t_{k-1}$ mit $t_i \in T^{(i)}$ für $i \in k$ darstellbar. Damit kann man alle Elemente aus G mit einem Backtrack-Algorithmus durchlaufen. Weiter gilt:

3.1.2 Lemma. *Zwei Elemente t, t' aus $G^{(i)}$ liegen genau dann in der gleichen $G^{(i+1)}$-Linksnebenklasse, wenn gilt:*

$$tx_i = t'x_i\,.$$

Für die Transversale $T^{(i)}$ von $G^{(i)}/G^{(i+1)}$ sind also die Elemente tx_i, $t \in T^{(i)}$ paarweise verschieden. Insbesondere ist $|T^{(i)}| \leq n$ (sogar $\leq n-i$). Ist $g \in G^{(i)}$, so

[1] Es sei angemerkt, dass der Begriff Basis nicht im Sinne der Matroid-Struktur ist, die wir in Kapitel 2 hervorgehoben haben. Die Basen im Sinne der Matroidtheorie heißen Transversalen.

gibt es andererseits genau ein $t \in T^{(i)}$ mit $gx_i = tx_i$. Damit kann man testen, ob ein $g \in S_X$ zur Gruppe G gehört, oder nicht: Das g liegt wie bereits erwähnt genau dann in G, wenn es von der Form $g = t_0 \cdots t_{k-1}$ mit $t_i \in T^{(i)}$ ist. Nach Lemma 3.1.2 können wir iterativ von links die $t_i \in T^{(i)}$ identifizieren und wegkürzen, solange, bis man entweder zu $g' = \mathrm{id}$ gelangt, dann ist $g \in G$ erfüllt, oder bis man zu einem g' gelangt mit $g'x_j = x_j$ für alle $j < i$, und $g'x_i \neq tx_i$ für alle $t \in T^{(i)}$; dann ist $g \notin G$.

Dieser Test benötigt einen Aufwand von $O(kn)$, wobei k die Länge der Basis $\vec{b}$ ist, und $n = |X|$. (Wir gehen davon aus, dass Berechnung des Inversen und Multiplikation zweier Gruppenelemente mit $O(n)$ sowie die Auswertung von gx_i mit konstantem Aufwand möglich ist.)

Für eine Ausführung der Algorithmen, siehe Standardliteratur zu Permutationsgruppen, z.B. [HEO05]. Es sei darauf hingewiesen, dass die inversen Elemente einer Linkstransversale eine Rechtstransversale bilden, man kann die Begebenheiten also auf Rechtstransversalen übertragen. Beispielsweise durchläuft ein Backtrack-Algorithmus alle Elemente der Form $t_{k-1} \cdots t_0$, mit t_i aus der Rechtstransversalen $T^{(i)}$, und zwei $t, t' \in G^{(i)}$ liegen genau dann in der gleichen $G^{(i+1)}$-Rechtsnebenklasse, wenn $t^{-1}x_i = t'^{-1}x_i$ ist.

3.2 Kurze Erzeugendensysteme, Labelled Branching

Wir führen im Folgenden *Labelled Branchings* nach [Jer86] ein. Damit kann man zum einen jedes Erzeugendensystem E von G in ein kurzes Erzeugendensystem mit höchstens $n - 1$ Elementen umwandeln. Es folgt insbesondere:

3.2.1 Satz. *Operiert G treu auf der Menge X mit $n := |X|$, so gibt es ein Erzeugendensystem von G mit höchstens $n - 1$ Elementen.*

Beweis: Folgt aus der Existenz eines Labelled Branchings zu G, siehe folgende Diskussion. □

Die *vollständigen* Labelled Branchings bieten weiterhin die Stärken der Stabilisatorkette, wie den Durchlauf durch G mittels Backtracking, und einen Test auf Enthaltensein mit $O(kn)$. Zusätzlich bietet ein vollständiges Labelled Branching zu G die Möglichkeit, eine Transversale von S_X/G (bzw. von G'/G mit $G \leq G' \leq S_X$) effizient zu durchlaufen.

Es sei X eine treue G-Menge mit $n := |X|$, und es sei $\vec{b} = (x_0, \ldots, x_{n-1})$ eine Basis der Länge n. (Bzgl. Aufwandsabschätzungen sei weiterhin k minimal, sodass bereits $G_{(x_0,\ldots,x_{k-1})} = \{\mathrm{id}\}$ ist.) Durch die Wahl einer Basis der Länge n sind insbesondere die Elemente aus X angeordnet, mit

$$x_i \leq x_j :\Longleftrightarrow i \leq j\,.$$

Für Aufwandsabschätzungen setzen wir voraus, dass die Zuordnungen $x_i \mapsto i$ und $i \mapsto x_i$ mit konstantem Aufwand bestimmbar seien.

Ausgehend von der Basis $\vec{b} \in X^n_{\text{inj}}$ sowie von der zugehörigen Stabilisatorkette können wir jedem $g \in G$ zwei Indizes zuordnen:

3.2.2 Definition. Sei G Gruppe, X treue G-Menge sowie $\vec{b}=(x_0, \ldots, x_{n-1})$ eine Basis der Länge n. Der *Typ* von $g \in G$ sei das Paar von Indizes $\mathrm{typ}(g) := (i, j)$, sodass gilt:

$$i = \max\{i' \mid g \in G^{(i')}\} \qquad \text{und} \qquad gx_i = x_j\,.$$

Der Index i heißt der *Stabilisatorindex*, und j heißt der *Transversalenindex* von g.

Ist $\mathrm{typ}(g) = (i, j)$, so lässt g die ersten i Elemente von $\vec{b}$ fix und bildet x_i auf x_j ab. Dabei gilt für $g \neq \mathrm{id}$ stets $i < j$, und $\mathrm{typ}(\mathrm{id}) = (n-1, n-1)$. Der Typ einer Permutation lässt sich offensichtlich mit einem Aufwand $O(n)$ bestimmen.

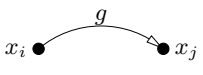
Durch die Anordnung der Elemente von X sind auch die $g \in G$ bzgl. der Listenschreibweise lexikographisch angeordnet. Diese Ordnung auf G lässt sich auf den Typ übertragen. Wir definieren entsprechend für zwei Typen (i, j), (i', j'):

$$(i, j) \leq (i', j') :\Leftrightarrow i > i' \text{ oder } i = i', j \leq j'\,.$$

Es gilt also:

$$g \leq_{\text{lex.}} g' \Longrightarrow \mathrm{typ}(g) \leq \mathrm{typ}(g')\,.$$

Mit Hilfe des Typs ordnen wir jedem Erzeugendensystem einen gerichteten Graphen zu: Sei $\Gamma(X, \vec{b}, E)$ der Digraph mit Knotenmenge X, sodass zu jedem Erzeuger $g \in E$ genau eine Kante von x_i nach x_j existiert, wobei (i, j) der Typ von g ist. Wir benutzen die $g \in E$ als Kantenbeschriftungen von $\Gamma(X, \vec{b}, E)$. (Im Gegensatz zum Cayley-Action-Graphen besitzt $\Gamma(X, \vec{b}, E)$ nur eine Kante zu jedem Erzeuger, anstatt $|X|$ Kanten.)



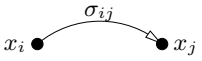
Entfernt man gegebenenfalls aus E den trivialen Erzeuger $g = \mathrm{id}$, so gilt $i < j$ für jede Kante (x_i, x_j) von $\Gamma(X, \vec{b}, E)$. Der Digraph ist also azyklisch, insbesondere ohne Schleifen. Haben zwei Erzeuger $g_1, g_2 \in E$, $g_1 \neq g_2$, den gleichen Typ (i, j), so können wir g_1 durch $g_1' := g_2^{-1} g_1$ von einem Typ (i', j') mit $i' > i$ ersetzen (da $g_1'(x_i) = x_i$), also mit einer Permutation von echt kleinerem Typ. Das resultierende Erzeugendensystem $E' = E \setminus \{g_1\} \cup \{g_1'\}$ (bzw. $E' = E \setminus \{g_1\}$, falls $g_1' = \mathrm{id}$ ist), erzeugt die gleiche Gruppe wie E. Durch Iteration ist es also möglich, ein Erzeugendensystem E zu konstruieren, sodass $\Gamma(X, \vec{b}, E)$ keine Mehrfachkanten besitzt. Wir können das Erzeugendensystem mit dieser Idee noch weiter modifizieren, sodass in jedem Knoten in $\Gamma(X, \vec{b}, E)$ höchstens eine Kante einmündet: Haben nämlich zwei Erzeuger $g_1, g_2 \in E$ die Typen $\mathrm{typ}(g_1) = (i_1, j)$, $\mathrm{typ}(g_2) = (i_2, j)$, $i_1 < i_2$, so können wir analog g_1 durch $g_1' := g_2^{-1} g_1$ ersetzen, wobei g_1' dann den Typ (i_1, i_2) hat, also von echt kleinerem Typ als g_1 ist.

Ein iteratives Ersetzen von Erzeugern verkleinert stets den Typ eines Erzeugers, ohne das Erzeugnis selbst zu ändern. Das Verfahren terminiert also, wir erhalten schließlich ein Erzeugendensystem, in dem keine zwei Erzeuger den gleichen Transversalenindex besitzen. Der Graph $\Gamma(X, \vec{b}, E)$ zu einem derart modifizierten Erzeugendensystem E ist dann ein *Branching*, d.h. ein azyklischer Digraph mit maximalem Eingangseckengrad 1. Durch das Branching ist eine Teilordnung auf den Punkten definiert, wir schreiben

$$x_i \preccurlyeq x_j \iff \text{Es gibt in } \Gamma(X, \vec{b}, E) \text{ einen (gerichteten) Pfad von } x_i \text{ nach } x_j.$$
$$x_i \dot{\preccurlyeq} x_j \iff \text{Es gibt in } \Gamma(X, \vec{b}, E) \text{ eine (gerichtete) Kante von } x_i \text{ nach } x_j.$$

Es leuchtet ein, dass ein derartiges Erzeugendensystem E höchstens $n - 1$ Erzeugende hat ($n = |X|$). Da die Abbildung $\mathrm{typ} : E \to \mathbb{N}^2$ insbesondere injektiv ist, führen wir folgende Schreibweise für die Erzeuger $g \in E$ ein:

$$\sigma_{ij} := g \iff typ(g) = (i, j)$$

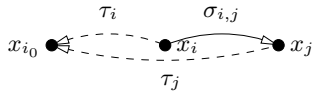


In $\Gamma(X, \vec{b}, E)$ ist also die Kante von x_i nach x_j mit dem Erzeuger σ_{ij} beschriftet. Wir bezeichnen $\Gamma(X, \vec{b}, E)$ zusammen mit der Kantenbeschriftung E als *Labelled Branching*.

Bevor wir die endgültige formale Definition für Labelled Branching bringen, noch eine weitere Überlegung: Ist E ein Erzeugendensystem, sodass $\Gamma(X, \vec{b}, E)$ ein Labelled Branching ist, so ersetzen wir E durch ein äquivalentes Erzeugendensystem. Dies ermöglicht uns, einen effizienten Test mit Aufwand $O(n^2)$ auf Enthaltensein einer Permutation $g \in S_X$ in dem Erzeugnis $G = \langle E \rangle$ durchzuführen. Darüber hinaus eignet sich die Datenstruktur, um gleichzeitig eine kanonisierende Abbil-

dung zu codieren. (Die letztgenannte Möglichkeit der Codierung einer kanonisierenden Abbildung wurde in der Literatur bisher nicht beachtet.)

Sei dazu B eine G-Transversale von X (oder eine G'-Transversale für eine Obergruppe $G' \geq G$). Wir ordnen jeder Wurzel x_i im Branching $\Gamma(X, \vec{b}, E)$ ein kanonisierendes Element τ_i zu, sodass $\tau_i \cdot x_i = x_{i_0} \in B$. Für die restlichen Knoten x_j, in die eine Kante mit Beschriftung σ_{ij} einmündet, setzen wir induktiv $\tau_j := \tau_i \cdot \sigma_{ij}^{-1}$. Dann gilt schließlich $\tau_i \cdot x_i \in B$ für alle $i \in n$. Die Abbildung



$$\tau : x_i \mapsto \tau_i$$

ist also eine kanonisierende Abbildung für G (bzw. für die Obergruppe G').

Es sollte vielleicht betont werden, dass die Menge der Wurzeln von $\Gamma(X, \vec{b}, E)$ im Allgemeinen zwar keine G-Transversale, aber eine Obermenge einer G-Transversale von X ist. Man kann also die Transversale B als Teilmenge der Wurzeln von $\Gamma(X, \vec{b}, E)$ wählen. Die kanonischen Elemente aus B sind dann genau diejenigen Punkte x_i mit $\tau_i = \mathrm{id}$.

Wir bezeichnen die τ_i als die Knotenbeschriftung des Branching. Aus ihr ist die Kantenbeschriftung, also die $\sigma_{ij} \in E$ rekonstruierbar, es gilt $\sigma_{ij} = \tau_j^{-1}\tau_i$. Wir können sogar allgemeiner für jeden Pfad in $\Gamma(X, \vec{b}, E)$, etwa von x_i nach x_j über $x_i x_j \cdots x_l x_k$, eine Pfadbeschriftung definieren, nämlich das Produkt der Kantenbeschriftungen, $\sigma_{ik} := \sigma_{lk} \cdots \sigma_{ij}$. Diese Pfadbeschriftung passt in dem Sinne in die bisherige Nomenklatur, dass der Typ von σ_{ik} gleich (i, k) ist. Allgemein gilt für $x_i \preccurlyeq x_k$:

$$\sigma_{ik} = \tau_k^{-1}\tau_i \,.$$

Anstatt dem Erzeugendensystem $E = \{\sigma_{ij} \mid x_i \dot{\preccurlyeq} x_j\}$ speichern wir die Abbildung τ, aus der die σ_{ij} effizient (in $O(n)$) berechnet werden können. Das Branching $\Gamma(X, \vec{b}, E)$ lässt sich in Form eines Vektors $\mathit{father} = (\mathit{father}_0, \ldots, \mathit{father}_{n-1})$ implementieren, wobei father_i den Vaterknoten der Eingangskante $(x_{\mathrm{father}_i}, x_i)$ spezifiziert, falls diese existiert, und ansonsten (d.h. falls x_i Wurzel eines Teilbaumes ist) setzen wir $\mathit{father}_i = -1$. Da in jeden Knoten höchstens eine Kante mündet, ist diese Darstellung möglich.

3.2.3 Definition. Sei G Gruppe und X treue G-Menge, $n := |X|$. Ein *Labelled Branching* von G ist ein Tripel $(\vec{b}, \Gamma, \tau)$ mit:

1. $\vec{b} = (x_0, \ldots, x_{n-1})$ ist eine Basis von X (eine Anordnung der Elemente).
2. Γ ist ein Branching mit Knotenmenge X, und für jede Kante (x_i, x_j) gilt: $i < j$.

3. $\tau = (\tau_0, \ldots, \tau_{n-1})$ ist ein Vektor von Permutationen, und es gilt:
   - Ist (x_i, x_j) eine Kante in Γ, so ist $\sigma_{ij} := \tau_j^{-1}\tau_i$ vom Typ (i, j).
   - Die Menge $E := \{\sigma_{ij} \mid (x_i, x_j) \text{ ist Kante in } \Gamma\}$ erzeugt G.

Insbesondere besagt Punkt 3, dass $\Gamma = \Gamma(X, \vec{b}, E)$ ist. Weiter definieren wir:

- Ist $\tau : x_i \mapsto \tau_i$ zusätzlich eine kanonisierende Abbildung für ${}_{G'}X$, $G' \geq G$, so heißt $(\vec{b}, \Gamma, \tau)$ auch *G'-kanonisierendes* Labelled Branching.
- Bildet zu jedem $i \in \underline{n}$:

  $$T^{(i)} := \{\sigma_{ik} \mid x_i \preccurlyeq x_k\}$$

  eine Links-Transversale von $G^{(i)}/G^{(i+1)}$, so heißt das Labelled Branching *vollständig.* $\Diamond$

Wir haben nicht gefordert, dass die $\tau_i \in G$ sind. Gilt dies zusätzlich, so ist natürlich auch die Menge der τ_i ein Erzeugendensystem. Ein vollständiges Labelled Branching codiert Transversalen der Stabilisatorkette zu $\vec{b}$, ermöglicht also insbesondere das Durchlaufen der ganzen Gruppe G mittels Backtracking und den schnellen Test auf Enthaltensein.

Algorithmen für Labelled Branchings

Aus den Linkstransversalen einer Stabilisatorkette zur Basis $\vec{b}$ kann man ein vollständiges Labelled Branching direkt ablesen (d.h. mit Aufwand $O(kn)$ zum Durchlaufen der Transversalen). Die Vereinigung der Linkstransversalen entspricht nämlich genau den Pfadbeschriftungen eines vollständigen Labelled Branchings L, d.h $x_i \preccurlyeq x_j$ gilt in L genau dann, wenn es ein Transversalenelement $t \in T^{(i)}$ vom Typ (i, j) gibt. Die Kanten sind dabei genau die minimalen Pfade, d.h. wir bestimmen diejenigen Transversalenelemente $t \in T^{(i)}$ vom Typ (i, j), sodass es in keiner anderen Transversale $T^{(i')}$ mit $i' > i$ ein Element vom Typ (i', j) gibt. Dazu durchlaufen wir die $T^{(i)}$ in absteigender Reihenfolge und speichern im Labelled Branching nur die $g \in T^{(i)}$ mit Typ (i, j), für die bisher in x_j keine Kante einmündet. Wir fügen die Kante (x_i, x_j) ein mit $\sigma_{ij} := g$. Die τ_i des Labelled Branchings können wir abschließend aus den σ_{ij} berechnen.

Hat man hingegen ein beliebiges Erzeugendensystem gegeben, etwa Schreier--Erzeuger, so kann man mit einem leeren Labelled Branching beginnend (ein Erzeugendensystem der Gruppe $\{\mathrm{id}\}$), der Reihe nach mit dem Algorithmus *sift*() (Sieb; siehe [Jer86] oder [Lan92]) neue Erzeuger hinzufügen. Man stellt so sicher,

dass das Erzeugendensystem kurz bleibt. Ist $m = |E|$ und $n = |X|$, so erhält man auf diese Art und Weise mit einem Aufwand von maximal $O(mn^2+n^4)$ ein kurzes Erzeugendensystem in Form eines (nicht vollständigen) Labelled Branchings.

M. Jerrum schlägt in [Jer86] vor, durch n-maliges Wiederholen dieses Vorgangs mit $O(mn^3 + n^5)$ ein vollständiges Labelled Branching aufzubauen. Hierzu gibt es auch eine Alternative: Für den von G. Cooperman und L. Finkelstein in [CF91] vorgestellten Algorithmus ist zwar keine theoretische Laufzeitabschätzung bekannt, in der Praxis erweist er sich jedoch als sehr effizient. Die beiden Algorithmen wurden in [Lan92] ausführlich verglichen.

Im Rahmen der Kanonisierung haben Labelled Branchings gegenüber der direkten Speicherung der Stabilisatorkette insbesondere den Vorteil, dass es einen effizienten Algorithmus zum Basiswechsel gibt. Brown et. al. beschreiben in [BLP89] (siehe auch [Grü95]), wie man aus einem Labelled Branching $(\vec{b}, \mathit{father}, \tau)$ zur Basis $\vec{b} = (x_0, \ldots, x_{n-1})$ ein Labelled Branching zu einer modifizierten Basis $\vec{b}'$ (d.h. einer modifizierten Anordnung der Elemente in X) berechnet. Der Algorithmus erlaubt insbesondere den Wechsel von $\vec{b}$ zu $\vec{b}'$, wenn $\vec{b}'$ aus $\vec{b}$ durch eine zyklische Vertauschung von k aufeinander folgenden Elementen hervorgeht:

$$\begin{aligned}\vec{b}' &= (i, i+1, \ldots, i+k-1) \cdot \vec{b}. \\ &= (x_0, \ldots, x_{i-1}, x_{i+k-1}, x_i, \ldots, x_{i+k-2}, x_{i+k}, \ldots, x_{n-1})\end{aligned}$$

Die Berechnung des Labelled Branchings $(\vec{b}', \mathit{father}', \tau')$ benötigt dann einen Zeitaufwand von $O(n^2)$. Durch maximal n derartige zyklische Vertauschungen ist jede Umsortierung der Elemente möglich, also ist mit $O(n^3)$ ein Labelled Branching zu jeder beliebigen Anordnung $\vec{b}'$ berechenbar.

3.3 Eine Transversale von G/G'

Sei X eine G-Menge und $\vec{b} = (x_0, \ldots, x_{n-1})$ eine Basis von X der Länge $n = |X|$. Vermöge

$$S_X \rightarrowtail X^n_{\text{inj}},\ \pi \mapsto \pi \circ \vec{b} := (\pi(x_0), \ldots, \pi(x_{n-1}))$$

induziert jede Permutation eine Anordnung auf X (in Abhängigkeit der gewählten Basis $\vec{b}$). Nach M. Jerrum kann man die Transversale der lexikographisch minimalen Elemente aus jeder Linksnebenklasse von G in S_X bestimmen, indem man

genau die Permutationen π auswählt, welche, als Anordnung interpretiert, eine topologische Anordnung zum vollständigen Labelled Branching von G bilden:

3.3.1 Satz. *(Jerrum, 1986) Sei $G \leq S_X$ und $(\vec{b}, \Gamma, \tau)$ ein vollständiges Labelled Branching zu G. Ein $\pi \in S_X$ ist genau dann lexikographisch minimal in der Linksnebenklasse πG, wenn für $x_i, x_j \in X$ gilt:*

$$x_i \preccurlyeq x_j \Longrightarrow \pi(x_i) \leq \pi(x_j)\,.$$

Beweis: Nach [Jer86]; wir zeigen die Kontrapositionen.
„⇒“. Falls π keine topologische Anordnung von Γ ist, so gibt es eine Kante (x_i, x_j) mit $\pi(x_i) > \pi(x_j)$. Die Kantenbeschriftung σ_{ij} führt dann zu einer kleineren Permutation in derselben Bahn: $\pi\sigma_{ij}$ und π stimmen auf $x_0, \ldots, x_{i-1}$ überein, da σ_{ij} diese Punkte fix lässt. Und $\pi\sigma_{ij}(x_i) = \pi(x_j) < \pi(x_i)$, also ist $\pi\sigma_{ij}$ lexikographisch kleiner als π.

„⇐“. Sei π nicht minimal in der Bahn πG, etwa $\pi g < \pi$. Es ist $g \neq \mathrm{id}$, sei $(i, j) := \mathrm{typ}(g)$. Dann ist wieder $\pi g(x_k) = \pi(x_k)$ für alle $k < i$ und $\pi g(x_i) = \pi(x_j) \neq \pi(x_i)$. Da πg nach Annahme lexikographisch kleiner als π ist, muss also $\pi(x_j) < \pi(x_i)$ sein. Andererseits gibt es in Γ einen Pfad von x_i nach x_j, da Γ vollständiges Labelled Branching ist. Also ist π keine topologische Anordnung. □

3.3.2 Folgerung. Ist $G' \leq G \leq S_X$, und $(\vec{b}, \Gamma, \tau)$ ein vollständiges Labelled Branching von G', so erhält man eine Transversale T_Γ der Linksnebenklassen G/G' durch

$$T_\Gamma = \{\pi \in G \mid \forall x_i, x_j \in X : x_i \preccurlyeq x_j \Rightarrow \pi(x_i) \leq \pi(x_i)\}\,.$$

Diese Aussage ist für die Kanonisierung diskreter Strukturen sehr wichtig, da wir dort eine Transversale von $G/\langle S\rangle$ durchlaufen, wobei G die operierende Gruppe, und S eine Menge von bekannten Automorphismen der diskreten Struktur $x \in X$ ist: $S \subseteq G_x$ (siehe Kapitel 4). Wir speichern also zu $\langle S\rangle$ das vollständige Labelled Branching $\Gamma = \Gamma(X, \vec{b}, \tilde{S})$ und durchlaufen G in Form eines Backtrack--Baums entlang der Rechtstransversalen einer Untergruppenkette. Dabei schneiden wir Teilbäume ab, sobald wir erkennen, dass darin keine topologischen Anordnungen zu Γ enthalten sind.

M. Jerrum verweist auf D. E. Knuth's Algorithmus aus [Knu79] zum effizienten lexikographischen Durchlaufen aller topologischen Anordnungen zu Γ. Dieser Algorithmus ist in folgender Hinsicht optimal: Ist π eine topologische Anordnung zu Γ, so wird die nachfolgende topologische Anordnung π' mit einem Aufwand $O(n - m)$ berechnet, falls m gleich dem ersten Index i ist, an dem sich $\pi(x_i)$

und $\pi'(x_i)$ unterscheiden. Leider besitzt Knuth's Algorithmus auch eine starke Voraussetzung: Es wird gefordert, dass die Basis $\vec{b}$ einem PREFIX-Durchlauf des Labelled Branchings Γ entspricht, d.h. zu jedem Knoten $x_i \in X$ bildet die Menge $\{j \mid x_i \preccurlyeq x_j\}$ ein Intervall. (Man überzeuge sich, dass dies für Branchings zu der in [Knu79] angegebenen Bedingung (3) äquivalent ist.)

Diese Voraussetzung könnten wir einerseits durch einen Basiswechsel erreichen. Es ist hierbei zu berücksichtigen, dass wir die Menge S während des Durchlaufs durch $G/\langle S\rangle$ dynamisch anpassen: Wir fügen neu gefundene Automorphismen als Erzeuger hinzu und modifizieren das Branching Γ, sodass wir ein vollständiges Labelled Branching zur neuen Gruppe $\langle S'\rangle$, mit $\langle S\rangle \leq \langle S'\rangle \leq G_x$ erhalten. Dabei muss darauf geachtet werden, dass die Basis $\vec{b}$ weiterhin kompatibel zum modifizierten Branching bleibt, d.h. es ist evtl. ein Basiswechsel nach jedem Hinzufügen neuer Automorphismen nötig. Findet die Kanonisierung jedoch im Rahmen einer ordnungstreuen Erzeugung statt, so ist ein Basiswechsel nicht ohne weiteres möglich. Es ist nämlich nicht klar, auf welche Art die zur Erzeugung wichtigen Lerneffekte durch den Basiswechsel beeinflusst werden. (Bei der ordnungstreuen Erzeugung ist es wichtig, dass das kanonische Element das minimale in seiner Bahn ist, und zwar bzgl. der lexikographischen Anordnung zu einer fest vorgegebenen Basis.)

Weiter gibt es Argumente, eine andere als die lexikographische Reihenfolge zu wählen: Ein Backtrack-Durchlauf durch G gemäß der *Rechts*transversalen einer Untergruppenkette ermöglicht ein frühzeitiges Erkennen von nicht-optimalen Teilbäumen (siehe Abschnitt 4.1, Unterabschnitt „Abschneiden von Teilbäumen“ auf Seite 86). Dies ist jedoch nicht mit der lexikographischen Reihenfolge kompatibel, sondern eher mit einem lexikographischen Durchlauf durch die Inversen der erreichten Elemente zu vergleichen. Wir durchlaufen also G wie besprochen als Backtrack-Baum entlang einer Rechtstransversalenkette, und geben ein effektives Kriterium an, um Teilbäume abzuschneiden, welche keine topologischen Anordnungen enthalten.

Sei $\vec{b} = (x_0, \ldots, x_{n-1})$ eine Basis der Länge n von X (d.h. eine Anordnung der Elemente, es gelte $x_i \leq x_j :\Longleftrightarrow i \leq j$) und Γ ein Branching mit Knotenmenge X. Wir schreiben wie bisher:

$$x_i \preccurlyeq x_j \;:\Longleftrightarrow\; \text{Es gibt in } \Gamma \text{ einen Pfad von } x_i \text{ nach } x_j.$$
$$x_i \dot{\preccurlyeq} x_j \;:\Longleftrightarrow\; \text{Es gibt in } \Gamma \text{ eine Kante von } x_i \text{ nach } x_j.$$

(Es ist $\preccurlyeq$ eine partielle Ordnung auf $\{x_0, \ldots, x_{n-1}\}$.) Die Basis entspreche einer topologischen Anordnung von Γ, d.h. es gelte:

$$x_i \preccurlyeq x_j \Longrightarrow i \leq j\,,$$

und das Branching Γ sei durch einen Vater-Vektor $(\mathit{father}_i)_{i\in n}$ implementiert:

$$\mathit{father}_i := \begin{cases} -1 & \text{falls } x_i \text{Wurzel von } \Gamma \text{ ist} \\ k < & \text{falls } x_k \dot{\preccurlyeq} x_i \,. \end{cases}$$

3.3.3 Hilfssatz. *Sei $\pi \in S_n$ und $S_n^{(i)}$ der punktweise Stabilisator von $x_0, \ldots, x_{i-1}$. Dann gibt es in der Rechtsnebenklasse $S_n^{(i)}\pi$ genau dann topologische Anordnungen zu Γ, wenn für alle $j < i$, und $x_k = \pi^{-1}(x_j)$ gilt:*

$$\mathit{father}_k \geq 0 \Longrightarrow \pi(x_{\mathit{father}_k}) < x_j \,.$$

Beweis: In der Nebenklasse $S_n^{(i)}\pi$ bleiben die Urbilder $\sigma^{-1}(x_j)$ für $j < i$, $\sigma \in S_n^{(i)}\pi$ konstant. Sei $j < i$ beliebig, und $x_k = \pi^{-1}(x_j)$, dann ist für alle $\sigma \in S_n^{(i)}\pi$: $\sigma(x_k) = x_j$. Ist $k' := \mathit{father}_k \neq -1$, so ist entweder $\sigma(x_{k'}) = x_l < x_j$ konstant für alle $\sigma \in S_n^{(i)}\pi$, oder für alle $\sigma \in S_n^{(i)}\pi$ gilt: $\sigma(x_{k'}) > x_j$. Die für topologische Anordnung notwendige Bedingung $\sigma(x_{k'}) < x_j$ $(= \sigma(x_k))$ muss also bereits für π erfüllt sein.
Wir zeigen, dass die angegebenen Bedingungen auch hinreichend für die Existenz einer topologischen Anordnung sind: Aufgrund der Voraussetzung bildet die Urbildmenge $I := \pi^{-1}(\{x_0, \ldots, x_{i-1}\})$ ein Ordnungsideal bzgl. der Teilordnung $\preccurlyeq$, d.h. ist $x_k \in I$, $x_{k'} \preccurlyeq x_k \Longrightarrow x_{k'} \in I$. Das Komplement $J := X \setminus I = \{x_j \mid \pi(x_j) \geq x_i\}$ bildet entsprechend einen unteren Teilwald von Γ. Wir können J topologisch anordnen bzgl. dem Teilgraphen $\Gamma{\downarrow_J}$ und erhalten so eine topologische Anordnung auf ganz Γ. Die Umsortierung von J entspricht einer Linksmultiplikation mit einem $\sigma \in S_n^{(i)}$, d.h. die gefundene topologische Anordnung liegt in $S_n^{(i)}\pi$. □

Die folgenden Überlegungen dienen zur Formulierung eines Algorithmus zum lexikographischen Durchlaufen einer Transversale von S_n/G analog zu [Knu79] (jedoch mit weniger restriktiven Anforderungen an die Basis $\vec{b}$), bzw. zum Abschneiden von Teilbäumen beim Backtrack-Durchlauf gemäß einer *Links*transversalenkette. Evtl. können diese Überlegungen auch auf den Backtrack-Durchlauf gemäß Rechtstransversalen übersetzt werden, um weitere Lerneffekte zur Verfügung zu stellen. Im weiteren Verlauf dieser Arbeit werden sie jedoch nicht genutzt.

Zur Vereinfachung der Argumentation erweitern wir das Branching Γ um eine Wurzel x_{-1}, sodass aus dem Branching ein Baum wird. Weiter sei:

$$J := \{j \in n \mid x_j \text{ ist Endknoten in } \Gamma\} \,,$$

zu jedem Knoten x_i sei J_i die Menge der Indizes $j > i$ von Endknoten, die nicht unterhalb von x_i liegen:

$$J_i := \{j \in J \mid i < j, x_i \not\preccurlyeq x_j\},$$

und schließlich sei für $i \in n$, $j \in J$:

$$h(i,j) := \max\{h \leq i \mid x_h \preccurlyeq x_j\}\,,$$

d.h. $h(i,j)$ bezeichne den ersten Knoten auf dem Pfad vom Endknoten x_j zurück zur Wurzel x_{-1} mit einem Index $\leq i$. Entspricht $\vec{b}$ wie bei Knuth einem PREFIX-Durchlauf von Γ, so gilt $x_{h(i,j)} \preccurlyeq x_i$. Im allgemeinen Fall gilt dies jedoch nicht zwingend.

Wir argumentieren im Folgenden auch mit Permutationen aus S_n, anstatt mit den $\pi \in S_X$. Jedem $\alpha \in S_n$ sei ein $\pi_\alpha \in S_X$ zugeordnet durch:

$$\pi_\alpha(x_i) := x_{\alpha(i)}\,.$$

Wir sagen auch, α sei eine topologische Anordnung, falls π_α eine topologische Anordnung zu Γ ist.

3.3.4 Hilfssatz. *Sei $\pi \in S_X$ eine topologische Anordnung. Gilt für $i \in n$ und $j \in J_i$ mit $h := h(i,j)$:*

$$\pi(x_h) < \pi(x_i) < \pi(x_j)\,,$$

so gibt es eine topologische Anordnung $\pi' \in S_X$ mit $\pi <_i \pi'$.

Beweis: Sei $\alpha \in S_n$ mit $\pi = \pi_\alpha$, und es gelte $\alpha(h) < \alpha(i) < \alpha(j)$. Weiter bezeichne x_l den direkten Nachfolger von x_h auf dem Pfad von x_h nach x_j. Dann ist $l > i$, im Teilbaum mit Wurzel x_l sind also nur Knoten mit Indizes $> i$ enthalten. Vertauschen wir nun die Beschriftungen von x_i und x_j, so ist es möglich, durch Umsortieren der Bewertungen in den Teilbäumen mit Wurzel x_i bzw. x_l wieder zu einer topologischen Anordnung β zu gelangen. Hierbei ist zu beachten, dass für die Beschriftung des Vaters x_h von x_l nach Voraussetzung $\alpha(h) < \alpha(i)$ gilt, die neue Beschriftung von x_l ist also weiterhin $> \alpha(h)$. Durch die Umsortierung wurden nur Indizes $\geq i$ geändert, insbesondere ist $\beta(i) = \alpha(j) > \alpha(i)$. Es gilt also $\alpha <_i \beta$, und somit ist $\pi' := \pi_\beta$ topologische Anordnung mit $\pi <_i \pi'$. □

3.3.5 Hilfssatz. *Sind $\pi, \pi' \in S_X$ topologische Anordnungen mit $\pi <_i \pi'$, so gibt es ein $j \in J_i$, sodass mit $h := h(i,j)$ gilt:*

$$\pi(x_h) < \pi(x_i) < \pi(x_j)\,.$$

Beweis: Es sei α mit $\pi = \pi_\alpha$ und β mit $\pi' = \pi_\beta$ und $\sigma := \alpha^{-1}\beta$. (d.h. $\beta(i) = \alpha(\sigma(i))$). Wir zeigen, dass für ein $j \in J_i$ mit $h := h(i,j)$ gilt: $\alpha(h) < \alpha(i) < \alpha(j)$.
Sei $c := (i, \sigma(i), \ldots)$ der Zykel in σ, welcher den Index i enthält. Da $\alpha <_i \beta$ gilt, ist i der kleinste im Zykel vorkommende Index, und es gilt $\alpha(\sigma(i)) > \alpha(i)$. Da $\beta(\sigma^{-1}(i)) = \alpha(i) < \beta(i)$ ist, und β eine topologische Anordnung ist, gilt $x_i \not\prec x_{\sigma^{-1}(i)}$. Der Zykel c durchläuft also insbesondere auch Punkte, die mit x_i bzgl. $\preccurlyeq$ unvergleichbar sind (kein direkter Pfad im Branching). Sei k der letzte Index in der Kette $\sigma^{-1}(i)$, $\sigma^{-2}(i), \ldots, \sigma^{-r}(i) = k$ von solchen mit x_i unvergleichbaren Indizes, d.h. $x_i \not\preccurlyeq x_k$, aber $x_i \preccurlyeq x_{\sigma^{-1}(k)}$. Es gilt $\alpha(k) = \beta(\sigma^{-1}(k)) \geq \beta(i) > \alpha(i)$. Im Zykel c gibt es also zwischen $k, \sigma(k), \ldots, i$ ein l mit $\alpha(l) > \alpha(i)$ und $\alpha(\sigma(l)) \leq \alpha(i)$. Mit $\beta(l) = \alpha(\sigma(l))$ haben wir einen Index l mit

$$\beta(l) \leq \alpha(i) < \alpha(l) \text{ und } x_i \not\preccurlyeq x_l\,.$$

Sei x_j ein Endknoten unterhalb von x_l, so ist $j \in J_i$. Sei weiter $h := h(i,j)$. Da $h < i$ ist, gilt $\alpha(h) = \beta(h)$, und weiter $\alpha(h) = \beta(h) < \beta(l) \leq \alpha(i)$. Andererseits ist $\alpha(i) < \alpha(l) \leq \alpha(j)$, insgesamt haben wir also gezeigt: Es gibt ein $j \in J_i$ mit $\alpha(h) < \alpha(i) < \alpha(j)$. □

Im Weiteren soll der Weg zu einem Algorithmus zum lexikographischen Durchlaufen einer Transversale von S_n/G aufgezeigt werden. Der erste Schritt in Knuth's Algorithmus ist die Bestimmung eines maximalen $m \in n$, sodass es eine topologische Anordnung β gibt mit $\alpha <_m \beta$. Dieser Teil wird modifiziert, und β kann dann analog zu [Knu79] bestimmt werden. Für den Fall $i >= m$ müssen wir das Kriterium $\alpha(h) < \alpha(i) < \alpha(j)$ aus Hilfssatz 3.3.4 nicht für die ganze Menge J_i testen, wir können die Mengen J_i vielmehr aufgrund folgender Aussage reduzieren:

3.3.6 Hilfssatz. *Für $i \in n$ sei $J_i' \subseteq J_i$ eine Teilmenge, sodass es für jedes $j \in J_i$ ein $j' \in J_i'$ gibt mit:*

$$h(j', j) \leq j' \qquad \textit{und} \qquad (\ h(i,j') = h(i,j) \quad \textit{oder} \quad x_{h(i,j')} \prec x_i\ )\,.$$

Es sei m der Index, sodass für die zu α lexikographisch nächstgrößere topologische Anordnung β gilt: $\alpha <_m \beta$. Dann gibt es bereits ein $j' \in J_m'$, sodass mit $h' := h(m, j')$ gilt:

$$\alpha(h') < \alpha(m) < \alpha(j')\,.$$

Beweis: Nach dem vorangehenden Hilfssatz gibt es ein $j \in J_m$ mit $\alpha(h) < \alpha(m) < \alpha(j)$. Ist $j \in J'$, so ist nichts zu zeigen. Ansonsten gibt es nach Vorausset-

Algorithmus 3.1 *calc_m*(α)

Vorberechnete Daten aus $(\vec{b}, \Gamma)$:
$j_0, \ldots, j_n$, Indizes
J, die Intervalle $[J[j_i] \ldots J[j_{i+1}])$ bezeichnen die Knoten aus J'_i
H, die Intervalle $[H[j_i] \ldots H[j_{i+1}])$ bezeichnen die Werte $h(i,j)$ zu $j \in J'_i$

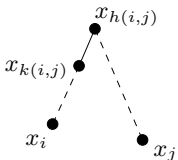
Eingabe:
$\alpha \in S_n$, eine topologische Anordnung von Γ.

Ausgabe:
m: Index, für lex. nächstgrößere topologische Anordnung β gilt: $\alpha <_m \beta$.

```
1 begin
2   for m = n − 1, ..., 0 do
3     for j = j_{m+1} − 1, ..., j_m do
4       if α(H_j) < α(m) ∧ α(m) < α(J_j) then return m; fi
5     od
6   od
7 end
```

zung ein $j' \in J'_m$ mit $h(j', j) \leq j'$ sowie $h(m, j') = h(m, j)$ oder $x_{h(m,j')} \prec x_i$. Da weiter $j' > m$ ist, ist die Bedingung $\alpha(h(j', j)) < \alpha(j') < \alpha(j)$ *nicht* erfüllt (Hilfssatz 3.3.4), d.h. entweder $\alpha(j') < \alpha(h(j', j))$, oder $\alpha(j') > \alpha(j)$. Die erste Alternative scheidet aus, da nach Voraussetzung $h(j', j) \leq j'$ ist. Also ist $\alpha(j') > \alpha(j) > \alpha(m)$. Aus beiden Alternativen für die zweite Eigenschaft von j' folgt $\alpha(h') < \alpha(m)$. □

Dies können wir z.B. wie folgt nutzen: Für $i \in n$, $j \in J$ sei



$$k(i,j) := \begin{cases} -1 & \text{falls } x_{h(i,j)} \not\preccurlyeq x_i \\ \min\{k > h(i,j) \mid x_k \preccurlyeq x_i\} & \text{sonst.} \end{cases}$$

d.h. $k(i,j)$ bezeichne den ersten Knoten auf dem Pfad von $x_{h(i,j)}$ nach unten in Richtung zum Knoten x_i, falls x_i echt unterhalb von $x_{h(i,j)}$ liegt. Ansonsten ist $k(i,j) = -1$.

3.3.7 Hilfssatz. *Für alle $j, j' \in J$, $i < k(j, j')$ gilt:*

$$h(i,j) = h(i,j') .$$

Beweis: Ist $l < i$, so gilt $x_l \preccurlyeq x_j \Leftrightarrow x_l \preccurlyeq x_{h(j,j')} \Leftrightarrow x_l \preccurlyeq x_{j'}$ nach der Definition von $h(j, j')$. □

Mit $J_i' = \{j_0, \ldots, j_{r-1}\}$, $j_0 < \cdots < j_{r-1}$ können wir also die Endknoten mit $k(j_{l-1}, j_l) \leq i$ streichen:

$$J_i' := \{j_l \in J_i' \mid l = 0 \text{ oder } i \geq k(j_{l-1}, j_l)\}\,.$$

Es gibt sicher noch weitere Möglichkeiten, um aus den Mengen J_i Elemente zu streichen. Im Fall, dass $\vec{b}$ einem PREFIX-Durchlauf durch Γ entspricht, kann man sogar als J_i' den ersten Endknoten $j > i$ mit $x_i \not\succcurlyeq x_i$ wählen, da für alle $j \in J_i$ gilt: $x_{h(i,j)} \preccurlyeq x_i$. Man erhält so genau den von D.E. Knuth vorgestellten Algorithmus. In diesem Sinne ist der hier vorgestellte Algorithmus also eine Verallgemeinerung.

4 Kanonisierung und Konstruktion diskreter Strukturen

Wir verallgemeinern in diesem Kapitel die bekannten Algorithmen zur Kanonisierung und Konstruktion von Graphen auf weitere Klassen diskreter Strukturen (d.h. endliche G-Mengen). Da die Menge $\mathcal{G}(n)$ aller Graphen mit n Knoten mittels Adjazenzmatrix in die G-Menge $2^{(n^2)}$einbettbar ist, ist eine Verallgemeinerung auf G-Mengen der Form $W^{(n^k)}$ naheliegend, $G \leq S_W \wr S_n \wr S_k$, W endliche Menge, $k \in n$. Insbesondere sind orientierte Matroide vom Rang k über n durch Chirotope codierbar, d.h. als Elemente von $3^{(n^k)}$. Ebenfalls in diese Klasse diskreter Strukturen passt die Menge $2^n = 2^{(n^1)}$ aller Teilmengen von n.

Es stellt sich heraus, dass der Algorithmus zur Kanonisierung von Graphen nach B. D. McKay [McK81] via iterierter Verfeinerung auch als iterierte Anwendung des Homomorphieprinzips gesehen werden kann. In der Tat wird dabei eine Folge von immer feiner werdenden H_i-Homomorphismen $f_i : X \to Y_i$ mit Definitionsbereich X und G-Mengen Y_i als Wertebereiche betrachtet, $H_i \leq G$. Während der Kanonisierung einer diskreten Struktur x erhält man so eine Kette von Untergruppen von G oberhalb des Stabilisators G_x:

$$G \geq G_{f_0(x)} \geq G_{f_1(x)} \geq \cdots \geq G_x \,,$$

entlang der ein kanonisches Element x_c mittels Homomorphieprinzip bestimmt wird. Die H_i-Homomorphismen werden dabei aus einem einzigen G-Homomorphismus $\Phi : G \times \mathcal{L}(G) \to Y^X$ gewonnen, welcher jedem Paar (g, H) eine Abbildung von X nach Y zuordnet, Y eine G-Menge.

Beim originalen Algorithmus für Graphen sind alle Wertemengen Y_i von der Form W_i^n, W_i Menge, und so sind die Stabilisatoren $G_{f_i(x)}$ Young-Untergruppen der S_n, lassen sich also durch eine Mengenpartition von n eindeutig beschreiben. Dies erklärt, wieso der Algorithmus für Graphen ursprünglich ohne explizites Erwähnen

der G-Homomorphismen formuliert werden konnte: Es wurde mit den Mengenpartitionen von n argumentiert. Für allgemeine diskrete Strukturen braucht man sich aber nicht auf Wertebereiche der Form W_i^n zu beschränken, und die Untergruppenkette kann beliebige Stabilisator-Gruppen (d.h. nicht nur Young-Untergruppen) enthalten. Durch geeignete Wahl der Homomorphismen kann man so eine feinere Kette von Zwischengruppen zwischen G und dem Stabilisator G_x bestimmen. Die Formulierung eines Kanonisierungsalgorithmus mittels Gruppenhomomorphismen ist also allgemeiner, als die sonst häufig verfolgte Strategie, diskrete Strukturen zunächst in Graphen umzuwandeln und die kanonische Form darauf zu berechnen.

Wir zeigen weiter den ebenso engen Zusammenhang zwischen dem Homomorphieprinzip und McKay's Ansatz aus [McK98] zur Konstruktion diskreter Strukturen auf. Hier ist insbesondere eine Analogie zu dem in Bayreuth von B. Schmalz entwickelten Leiterspiel [Sch93] hervorzuheben. B. D. McKay vermutete diesen Zusammenhang bereits in [McK98]: „There are complicated relationships between these methods, but to a large extent they have not been explored." Für das bereits länger bekannte Leiterspiel ergibt sich aus den Überlegungen McKay's heraus die neue Möglichkeit, mehrstufige Konstruktionen diskreter Strukturen als Backtrack-Algorithmus in Tiefensuche zu formulieren.

4.1 Kanonisierung

Eine der wichtigsten Aufgabenstellungen im Rahmen der Konstruktion von diskreten Strukturen ist die Bestimmung von kanonischen Repräsentanten sowie die damit eng verwandte Berechnung der Automorphismengruppe. Da wir häufig bzgl. einer eingeschränkten Gruppenoperation kanonisieren, definieren wir:

4.1.1 Definition. Ein *System kanonischer Transversalen* auf der G-Menge X sei eine Abbildung

$$\mathcal{C}an : \mathcal{L}(G) \to 2^X ,$$

sodass für alle $H \leq G$ gilt:

1. $\mathcal{C}an(H)$ ist H-Transversale von X,
2. $H_1 \leq H_2 \Longrightarrow \mathcal{C}an(H_1) \supseteq \mathcal{C}an(H_2)$.

Wir bezeichnen $\mathcal{C}an(H)$ als die *H-kanonische Transversale* bzgl. $\mathcal{C}an$, und die $x \in \mathcal{C}an(H)$ heißen *H-kanonische Repräsentanten* bzgl. $\mathcal{C}an$. Die G-kanonische

Transversale nennen wir auch kurz die kanonische sowie die G-kanonischen Repräsentanten kanonisch.

Bei der Bestimmung eines kanonischen Repräsentanten zu x kann gleichzeitig der Stabilisator G_x mitberechnet werden. Wir legen deshalb für Implementationen von Kanonisierern folgende Schnittstelle fest:

4.1.2. Interface: Sei X eine G-Menge und $\mathcal{C}an$ ein System kanonischer Transversalen. Ein Algorithmus

$$(g, E') \leftarrow \mathit{can}(E, x)$$

heißt *Kanonisierer* auf der G-Menge X bzgl. $\mathcal{C}an$, falls die Schnittstelle folgende Ein- und Ausgabe hat:
Eingabe:
E ist Erzeugendensystem einer Untergruppe $H \leq G$,
$x \in X$ ist eine diskrete Struktur,

Ausgabe:
g ist H-kanonisierendes Element, d.h. es gilt: $gx \in \mathcal{C}an(H)$, und
E' ist ein Erzeugendensystem des Stabilisators H_x.

Ein Kanonisierer ist also eine Abbildung der folgenden Form:

$$\mathit{can} : \mathcal{L}(G) \times X \to G \times \mathcal{L}(G)\,.$$

Benutzt man beispielsweise das System kanonischer Transversalen, welches als H-kanonische Repräsentanten die Minima der H-Bahnen wählt (bzgl. einer vorgegebenen Totalordnung $\leq$ auf X),

$$\mathcal{C}an_{\min}(H) := \{x \in X \mid \forall h \in H : x \leq hx\}\,,$$

so kann man mittels Bahnenalgorithmus alle $\frac{|H|}{|H_x|}$ Elemente der Bahn durchlaufen, und das Minimum bestimmen. Eine Kanonisierung bzgl. $\mathcal{C}an_{\min}$ bezeichnen wir als Minimierung, $\mathcal{C}an_{\min}$ heißt auch das System der minimalen Transversalen.

Bekanntlich kann man aus einem G-Homomorphismus $f : X \to Y$ und gegebenen Systemen kanonischer Transversalen $\mathcal{C}an_X$ bzw. $\mathcal{C}an_Y$ auf X bzw. Y mit Kanonisierern can_X und can_Y einen effizienteren Kanonisierer can_f auf X formulieren.

4.1.3 Satz. (Homomorphieprinzip: Kanonisierung, Schmalz '93)
Seien X, Y G-Mengen und $\mathcal{C}an_X$, $\mathcal{C}an_Y$ Systeme kanonischer Transversalen auf X bzw. Y. Dann ist $\mathcal{C}an_f$ mit

$$\mathcal{C}an_f(H) := \{x \in X \mid f(x) \in \mathcal{C}an_Y(H) \wedge x \in \mathcal{C}an_X(H_{f(x)})\}$$

ein System kanonischer Transversalen auf X. Sind weiter can$_X$ und can$_Y$ Kanonisierer zu $\mathcal{C}an_X$ bzw. $\mathcal{C}an_Y$, so ist der folgende Algorithmus can$_f$ ein Kanonisierer zu $\mathcal{C}an_f$:

```
1 proc can_f(E, x)
2   begin
3     (g, E_1) := can_Y(E, f(x));
4     (h, E') := can_X(E_1^g, gx);
5     return (hg, E'^{g^{-1}})
6   end.
```

Unter der Voraussetzung, dass $H > H_{f(x)} > H_x$ ist, kann man davon ausgehen, dass die Kanonisierung mit *can*$_f(E, x)$ geringeren Aufwand benötigt als die direkte Kanonisierung mit *can*$_X(E, x)$: Im Fall, dass *can*$_X$ und *can*$_Y$ mittels Bahnenalgorithmus implementiert sind, wird bei *can*$_f(E, x)$ die H-Bahn von $f(x)$ sowie die $H_{f(gx)}$-Bahn von gx durchlaufen, also $\frac{|H|}{|H_{f(x)}|} + \frac{|H_{f(x)}|}{|H_x|}$ Elemente, im Gegensatz zu den $\frac{|H|}{|H_x|} = \frac{|H|}{|H_{f(x)}|} \cdot \frac{|H_{f(x)}|}{|H_x|}$ Elementen bei der direkten Kanonisierung mit *can*$_X(E, x)$. In diesem Sinne spricht man auch von einer Logarithmierung des Aufwands.

In ähnlicher Weise ist auch für direkte Produkte von G-Mengen ein System kanonischer Transversalen gegeben:

4.1.4 Folgerung. Seien X, Y G-Mengen und $\mathcal{C}an_X$ und $\mathcal{C}an_Y$ Systeme kanonischer Transversalen auf X bzw. Y. Dann ist auf der G-Menge $X \times Y$ ein System kanonischer Transversalen $\mathcal{C}an_{X\times Y}$ induziert mit

$$\mathcal{C}an_{X\times Y}(H) := \{(x, y) \in X \times Y \mid x \in \mathcal{C}an_X(H) \wedge y \in \mathcal{C}an_Y(H_x)\}\,.$$

Sind *can*$_X$ und *can*$_Y$ Kanonisierer zu $\mathcal{C}an_X$ bzw. $\mathcal{C}an_Y$, so ist der folgende Algorithmus *can*$_{X\times Y}$ ein Kanonisierer zu $\mathcal{C}an_{X\times Y}$:

```
1 proc can_{X×Y}(E, (x, y))
2   begin
3     (g, E_1) := can_X(E, x);
4     (h, E') := can_Y(E_1^g, gy);
5     return (hg, E'^{g^{-1}})
6   end.
```

Sind dabei can_X und can_Y per Bahnenalgorithmus implementiert, so sind zur Berechnung des H-kanonischen Repräsentanten $\frac{|H|}{|H_x|} + \frac{|H_x|}{|H_{(x,y)}|}$ Elemente zu durchlaufen, es findet also wieder eine Logarithmierung im Vergleich zur direkten Implementierung des Bahnenalgorithmus auf $X \times Y$ statt (durchläuft $\frac{|H|}{|H_{(x,y)}|}$ Elemente).

Iterativ erhält man ein System kanonischer Transversalen auf dem n-fachen direkten Produkt von G-Mengen:

4.1.5 Folgerung. Seien X_i G-Mengen und $\mathcal{C}an_{X_i}$ Systeme kanonischer Transversalen auf X_i, $i \in n$. Dann ist auf der G-Menge $\bigtimes_{i\in n} X_i$ ein System kanonischer Transversalen $\mathcal{C}an_{\bigtimes_{i\in n} X_i}$ induziert mit

$$\mathcal{C}an_{\bigtimes_{i\in n} X_i}(H) := \{(x_i)_{i\in n} \in \bigtimes_{i\in n} X_i \mid \forall i \in n : x_i \in \mathcal{C}an_{X_i}(\bigcap_{j<i} H_{x_j})\}\,.$$

Ist can_{X_i} ein Kanonisierer zu $\mathcal{C}an_{X_i}$, $i \in n$, so ist der folgende Algorithmus $can_{\bigtimes_{i\in n} X_i}$ ein Kanonisierer zu $\mathcal{C}an_{\bigtimes_{i\in n} X_i}$:

```
1 proc can_{⨉_{i∈n} X_i}(E, (x_i)_{i∈n})
2   begin
3     g := id;
4     for i = 0 to n − 1 do
5       (h, E) := can_{X_i}(E, g x_i);
6       E := E^h; g := hg;
7     od;
8     return (g, E^{g^{-1}})
9   end.
```

Ist nun eine Folge $f_0, f_1, \ldots, f_{n-1}$ von G-Homomorphismen mit Definitionsbereich X gegeben, $f_i : X \to Y_i$, so ist $\bigtimes_{i\in n} Y_i$ eine G-Menge, und Folgendes ist ein G-Homomorphismus:

$$\bar{f} : X \to \bigtimes_{i\in n} Y_i, x \mapsto (f_0(x), \ldots, f_{n-1}(x))\,.$$

Wir erhalten ein System kanonischer Transversalen $\mathcal{C}an_{(f_i)_{i\in n}} := \mathcal{C}an_{\bar{f}}$ nach 4.1.3 und 4.1.5 aus den Systemen kanonischer Transversalen $\mathcal{C}an_X$ und $\mathcal{C}an_{Y_i}$, $i \in n$:

4.1.6 Folgerung. Seien X, Y_i, $i \in n$ G-Mengen und $\mathcal{C}an_X, \mathcal{C}an_{Y_i}$, $i \in n$, entsprechende Systeme kanonischer Transversalen sowie $f_i : X \to Y_i$ G-Homomorphismen. Dann ist auf der G-Menge X ein System kanonischer Transversalen $\mathcal{C}an_{(f_i)_{i\in n}}$ induziert mit

$$\mathcal{C}an_{(f_i)_{i\in n}}(H) := \{x \in X \mid (f_i(x))_{i\in n} \in \mathcal{C}an_{\bigtimes_{i\in n} Y_i} \wedge x \in \mathcal{C}an_X(H_{(f_i(x))_{i\in n}})\}\,.$$

Sind can_X, can_{Y_i},$i \in n$ Kanonisierer zu den Systemen kanonischer Transversalen, so ist der folgende Algorithmus $\mathit{can}_{(f_i)_{i\in n}}$ ein Kanonisierer zu $\mathcal{C}an_{(f_i)_{i\in n}}$:

```
1  proc can_{(f_i)_{i∈n}}(E, x)
2    begin
3      g := id;
4      for i = 0 to n − 1 do
5          (h, E) := can_{Y_i}(E, f_i(gx));
6          E := E^h; g := hg;
7      od;
8      (h, E) := can_X(E, x);
9      E := E^h; g := hg;
10     return (g, E^{g^{-1}})
11   end.
```

Die Kanonisierung wird dabei in $n+1$ Schritte unterteilt, bei günstiger Verteilung der Indizes in der Untergruppenkette

$$G \geq G_{f_0(x)} \geq G_{(f_0(x),f_1(x))} \geq \cdots \geq G_{(f_i(x))_{i\in n}} \geq G_x\,,$$

erhält man eine starke Aufwandsreduktion.

Anstatt direkt mit dem Bahnenalgorithmus auf X zu minimieren, kanonisieren wir zunächst Bilder unter G-Homomorphismen in einfacheren G-Mengen Y_i, um den Stabilisator möglichst klein zu bekommen. Erst im letzten Schritt fallen wir dann auf eine Kanonisierung auf X zurück, wobei nur noch für eine Untergruppe von H kanonisiert wird. Aber auch diese letzte Kanonisierung auf X kann weiter aufgeteilt werden, indem wir für ein $G' \leq G$ einen G'-Kanonisierer $\mathcal{C}an$ auf X benutzen, d.h. genauer: einen davon induzierten G-Kanonisierer. Der verwendete G'-Kanonisierer kann auf weitere G'-Homomorphismen zur Effizienzsteigerung zurückgreifen. Wir beschreiben das als *Heben* eines G'-Kanonisierers zu einem G-Kanonisierer:

4.1.7 Bemerkung. Sei $G' \leq G$, X G-Menge und $\mathcal{C}an : \mathcal{L}(G') \to 2^X$ ein System kanonischer Transversalen auf X als G'-Menge. Weiter sei X totalgeordnet mittels „$\leq$“. Dann wird auf X als G-Menge ein System kanonischer Transversalen $\mathcal{C}an{\uparrow^G}$: $\mathcal{L}(G) \to 2^X$ induziert, wenn wir für $H \leq G$ mit $H' := H \cap G'$ als H-kanonische Repräsentanten (bzgl. $\mathcal{C}an{\uparrow^G}$) genau die minimalen H'-kanonischen Repräsentanten (bzgl. $\mathcal{C}an$) der H-Bahn wählen:

$$\mathcal{C}an{\uparrow^G}\,(H) := \{x \in X \mid x \in \mathcal{C}an(H') \wedge \forall h \in H, hx \in \mathcal{C}an(H') : x \leq hx\}.$$

Sei weiter *can* ein Kanonisierer zu $\mathcal{C}an$. Ein $H \leq G$ sei stets durch ein Erzeugendensystem E gegeben derart, dass wir aus E sowohl ein Erzeugendensystem E' von $H' := H \cap G'$ als auch eine Rechtstransversale T von $H'\backslash H$ ablesen können (z.B. ein starkes Erzeugendensystem, falls $G \geq G' \geq \cdots$ Teil einer Stabilisatorkette ist). Dann ist der folgende Algorithmus *can*$\uparrow^G$ ein Kanonisierer zu $\mathcal{C}an{\uparrow^G}$:

```
1  proc can↑^G (E, x)
2    begin
3      E' := (Erzeugendensystem von H');
4      T := (Transversale von H'\H);
5      g := NIL;
6      foreach t ∈ T do
7        (h, E_h) := can(E', tx);
8        if g = NIL ∨ hx < gx then
9          g := h; E_g := E_h;
10       fi
11     od;
12     return (g, E_g)
13   end.
```

Eine H-Kanonisierung wird also in $\frac{|H|}{|H'|}$ H'-Kanonisierungen unterteilt.

Kanonisierung entlang einer Untergruppenkette

Ist G mit einer Untergruppenkette $G = G^{(0)} \geq \cdots \geq G^{(r)} = \mathrm{id}$ gegeben, so erhält man iterativ ein System kanonischer Transversalen:

$$(\cdots(\mathcal{C}an_{\mathrm{id}}{\uparrow^{G^{(r-1)}}}){\uparrow^{G^{(r-2)}}}\cdots){\uparrow^{G^{(0)}}},$$

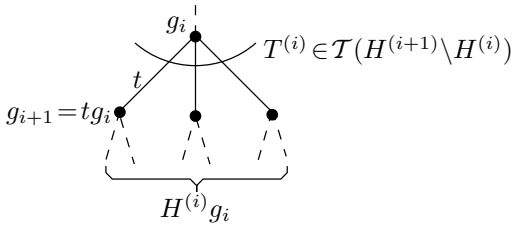


Abbildung 4.1: Der Backtrack-Baum zum Kanonisieren

wobei $\mathcal{C}an_{\mathrm{id}}$ das triviale System kanonischer Transversalen zur Gruppe $G^{(r)} = \{\mathrm{id}\}$ ist, d.h.

$$\mathcal{C}an_{\mathrm{id}} : \mathcal{L}(\{\mathrm{id}\}) \to 2^X, \{\mathrm{id}\} \mapsto X\,.$$

Offensichtlich ist $\mathcal{C}an_{\mathrm{id}}\uparrow^{G^{(r-1)}} \cdots \uparrow^{G^{(0)}} = \mathcal{C}an_{\min}$, d.h. es wird das Minimum aus jeder Bahn als kanonisch ausgewählt. Der Kanonisierer $can_{\mathrm{id}}\uparrow^{G^{(r-1)}} \cdots \uparrow^{G^{(0)}}$ durchläuft zur Minimierung eines x innerhalb der Bahn Hx einen Backtrack--Baum entlang von Rechtstransversalen $T^{(i)}$ der Untergruppenkette $H = H^{(0)} \geq \cdots \geq H^{(r)} = \{\mathrm{id}\}$ mit $H^{(i)} := H \cap G^{(i)}$, $i \leq r$. Wir beschriften im Baum jeden Knoten auf Ebene i mit einem Repräsentanten der entsprechenden $H^{(i)}$-Bahn: Die Wurzel sei mit der Identität beschriftet. Ist $T^{(i)}$ eine Rechtstransversale von $H^{(i+1)} \backslash H^{(i)}$, so hat ein Knoten mit Beschriftung g_i auf Ebene i für jedes Transversalenelement $t \in T^{(i)}$ genau eine ausgehende Kante, die zu einem Sohn auf Ebene $i+1$ mit der Beschriftung $g_{i+1} := tg_i$ führt (siehe Abbildung 4.1). Der Kanonisierer bestimmt dann das Minimum der $g_r x$ zu allen Knotenbeschriftungen der untersten Ebene.

Auf jeder Ebene i des Backtrack-Baums kann man nun zusätzlich $G^{(i)}$-Homomorphismen $f_{i,0}, \ldots, f_{i,j_i-1} : X \to Y$, $j_i \in \mathbb{N}$, $i \in r$, heranziehen. Wir führen folgende Notation ein: Es seien I und $\bar{I}$ die Mengen von Index-Paaren:

$$\begin{aligned} I &:= \{(i,j) \mid i < r, j < j_i\}, \quad \text{und} \\ \bar{I} &:= \{(i,j) \mid i \leq r, j \leq j_i\}\,. \end{aligned}$$

Mit der lexikographischen Anordnung auf $\mathbb{N}^2$ setzen wir für $(i,j) \in \bar{I}$:

$$\begin{aligned} I_{(i,j)} &:= \{(i',j') \in I \mid (i',j') < (i,j)\}, \\ k_{(i,j)} &:= |I_{(i,j)}|\,. \end{aligned}$$

Zu $(i,j) \in \bar{I}$ betrachte nun die $G^{(i)}$-Homomorphismen:

$$\bar{f}_{i,j} : X \to Y^{k_{(i,j)}}, x \mapsto (f_{i',j'}(x))_{(i',j') \in I_{(i,j)}}\,.$$

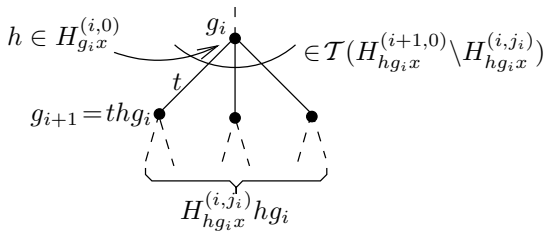


Abbildung 4.2: Der reduzierte Backtrack-Baum

(Es gilt $\bar{f}_{i,j_i} = \bar{f}_{i+1,0}$; die Abbildungen werden jedoch bzgl. unterschiedlicher Gruppenoperationen interpretiert.) Es sei weiter $H^{(i)} := H \cap G^{(i)}$. Für ein festes $x \in X$ erhalten wir die Folge der $H^{(i)}$-Stabilisatoren zu den $G^{(i)}$-Homomorphismen $\bar{f}_{i,j}(x)$, $(i,j) \in \bar{I}$:

$$H_x^{(i,j)} := H^{(i)}_{\bar{f}_{i,j}(x)} .$$

Dies ergibt folgende Untergruppenkette in H:

$$H = H_x^{(0,0)} \geq H_x^{(0,1)} \geq \cdots \geq H_x^{(0,j_0)} \geq H_x^{(1,0)} \geq H_x^{(1,1)} \geq \cdots \\ \cdots \geq H_x^{(r-1,j_{r-1})} \geq H_x^{(r,0)} = \{\mathrm{id}\} .$$

Weiter bezeichnen wir für $H \leq G$, $(i,j) \in \bar{I}$ ein $x \in X$ als (H,i,j)-*semikanonisch*, falls es (H,i',j')-semikanonisch für jedes $(i',j') \in \bar{I}$ mit $(i',j') < (i,j)$ ist , und falls zusätzlich gilt: $\bar{f}_{i,j}(x) \in \mathcal{C}an_{Y^{k_{(i,j)}}}(H^{(i)})$.

Wir betrachten nun folgendes wichtige System kanonischer Transversalen:

4.1.8 Satz. *Seien X, Y G-Mengen, $G = G^{(0)} \geq \cdots \geq G^{(r)} = \{\mathrm{id}\}$ eine Untergruppenkette, $\mathcal{C}an_Y$ ein System kanonischer Transversalen auf Y, und $f_{i,j} : X \to Y$ $G^{(i)}$-Homomorphismen. Das System kanonischer Transversalen*

$$((\cdots(\mathcal{C}an_{\mathrm{id}}\uparrow^{G^{(r-1)}})_{\bar{f}_{r-1,j_{r-1}}} \cdots)\uparrow^{G^{(0)}})_{\bar{f}_{0,j_0}}$$

hat als H-kanonische Repräsentanten genau die minimalen $(H,r,0)$-semikanonischen x aus jeder H-Bahn.

Mit 4.1.6 und 4.1.7 erhalten wir dazu einen Kanonisierer anhand eines Backtrack-Baums. Dabei wird auf jeder Ebene des Baums zunächst j_i mal gemäß Homomorphieprinzip das Bild unter einem $G^{(i)}$-Homomorphismus kanonisiert.

Der Backtrack-Baum ändert sich im Vergleich zu obiger Version wie folgt: Die Wurzel sei weiterhin mit der Identität beschriftet. Für die Beschriftung g_i eines Knotens auf Ebene i gelte: $g_i x$ ist $(H, i, 0)$-kanonisch. Es wird mittels 4.1.6 ein $h \in H_{g_i x}^{(i,0)}$ bestimmt, derart dass $hg_i x$ (H, i, j_i)-kanonisch ist. Für jedes Element t einer Transversale von $H_{hg_i x}^{(i+1,0)} \backslash H_{hg_i x}^{(i,j_i)}$ führt nun eine Kante zu einen Knoten auf Ebene $i+1$ mit Beschriftung $g_{i+1} := thg_i$. Dabei ist g_{i+1} $(H, i+1, 0)$-semikanonisch. Siehe auch Abbildung 4.2.

Auf diese Weise erreicht man auf Ebene r im Baum genau die $(Hr, 0)$-semikanonischen Elemente $g_r x$. Das Minimum davon ist der gesuchte kanonische Repräsentant und die entsprechende Knotenbeschriftung ist kanonisierendes Element.

Iterierte Verfeinerung

Wir haben skizziert, wie man bei gegebenen $G^{(i)}$-Homomorphismen $f_{i,j}$, $j \in j_i$,$i \in r$, effizient kanonisiert, und weiter unten wird sich herausstellen, dass die bei der Graph-Kanonisierung verwendete iterierte Verfeinerung implizit solche $f_{i,j}$ verwendet. Wir beschreiben, wie die $f_{i,j}$ aus einem G-Homomorphismus $\Phi : G \times \mathcal{L}(G) \to Y^X$ gewonnen werden.

Der Untergruppenverband $\mathcal{L}(G)$ ist eine G-Menge vermöge Konjugation, also auch $G \times \mathcal{L}(G)$ mit

$$g(h, H) := (gh, gHg^{-1}) .$$

Die Menge Y^X für G-Mengen X, Y ist ebenfalls G-Menge vermöge:

$$g \star f := gf(g^{-1}x) .$$

4.1.9 Definition. Seien X, Y G-Mengen. Ein *Verfeinerer* zu Y^X ist ein G-Homomorphismus $\Phi : G \times \mathcal{L}(G) \to Y^X$, sodass für alle $(g, H) \in G \times \mathcal{L}(G)$ gilt:

$$H \leq G_{\Phi(g,H)}$$

4.1.10 Bemerkung. Seien X, Y G-Mengen und $\Phi : G \times \mathcal{L}(G) \to Y^X$ ein Verfeinerer. Sei weiter $(g, H) \in G \times \mathcal{L}(G)$. Es gelten folgende Aussagen:

1. $\Phi(g, H) : X \to Y$ ist H-Homomorphismus.
2. Für alle $h \in H$ gilt: $\Phi(hg, H) = \Phi(g, H)$.

Beweis:
1. Sei $h \in H$ und $x \in X$ beliebig. Nach Definition ist h im Stabilisator von $f := \Phi(g, H)$, d.h. $f(x) = (h \star f)(x) = hf(h^{-1}x)$, es folgt $h^{-1}f(x) = f(h^{-1}x)$, und damit die H-Homomorphie von $f = \Phi(g, H)$.
2. Es ist $\Phi(hg, H) = \Phi(hg, hHh^{-1}) = \Phi(h(g, H))$ und wegen der G-Homomorphie von Φ ist letzteres gleich $h \star \Phi(g, H)$. Da h nach Definition im Stabilisator von $\Phi(g, H)$ liegt, folgt die Behauptung. □

Sei Z eine weitere G-Menge, und auf Z^X operiere G vermöge „$\star$“ wie oben. Mittels eines Kanonisierers auf Z und eines Verfeinerers zu Y^X ordnen wir mit $H \leq G$ jedem $f \in Z^X$ ein $f_H \in Y^X$ zu:

4.1.11 Definition. Seien X, Y, Z G-Mengen, $\mathcal{C}an$ ein System kanonischer Transversalen auf Z samt Kanonisierer *can* : $\mathcal{L}(G) \times Z \to G \times \mathcal{L}(G)$, und Φ : $G \times \mathcal{L}(G) \to Y^X$ ein Verfeinerer. Wir definieren zu $f \in Z^X$ und $H \leq G$ eine Abbildung $f_H \in Y^X$, indem wir für $x \in X$ setzen:

$$\begin{aligned}(h_{f(x)}, H_{f(x)}) &:= \mathit{can}(H, f(x))\,,\\ f_H(x) &:= \Phi(h_{f(x)}^{-1}, H_{f(x)})(x)\,.\end{aligned}$$

Die Wahl des kanonisierenden Elements $h_{f(x)}$ beeinflusst nicht die Abbildung f_H: Ist $h'_{f(x)} = h_{f(x)}h$ mit $h \in H_{f(x)}$, so ist nämlich nach vorangehender Bemerkung $\Phi((h_{f(x)}h)^{-1}, H_{f(x)}) = \Phi(h_{f(x)}^{-1}, H_{f(x)})$. Die Definition ist also unabhängig vom gewählten Kanonisierer *can* zu $\mathcal{C}an$.

4.1.12 Lemma. *Seien X, Y, Z G-Mengen, $\mathcal{C}an$ ein System kanonischer Transversalen auf Z und $\Phi : G \times \mathcal{L}(G) \to Y^X$ ein Verfeinerer. Ist $f \in Z^X$ ein G-Homomorphismus, so ist auch $f_G \in Y^X$ ein G-Homomorphismus.*

Beweis: Sei $g \in G$ und $g_{f(x)}$, $g_{f(gx)}$ kanonisierende Elemente zu $f(x)$ bzw. $f(gx)$. Da f G-Homomorphismus ist, gelten folgende äquivalente Aussagen:

$$\begin{aligned}g_{f(gx)}f(gx) &= g_{f(x)}f(x)\\ g_{f(x)}^{-1}g_{f(gx)}gf(x) &= f(x)\\ g_{f(x)}^{-1}g_{f(gx)}g &\in G_{f(x)}\\ \exists h \in G_{f(x)} : g_{f(gx)} &= g_{f(x)}hg^{-1}.\end{aligned}$$

Weiter ist $G_{f(gx)} = gG_{f(x)}g^{-1}$, und wir erhalten:

$$\begin{aligned} f_G(gx) &= \Phi((g_{f(x)}hg^{-1})^{-1}, gG_{f(x)}g^{-1})(gx) \\ &= \Phi(gh^{-1}g_{f(x)}^{-1}, gG_{f(x)}g^{-1})(gx) \\ &\underset{\Phi\ G\text{-Hom.}}{=} (g \star \Phi(h^{-1}g_{f(x)}^{-1}, G_{f(x)}))(gx) \\ &= g\Phi(h^{-1}g_{f(x)}^{-1}, G_{f(x)})(g^{-1}gx) \\ &\underset{h^{-1}\in G_{f(x)}}{=} g\Phi(g_{f(x)}^{-1}, G_{f(x)})(x) \qquad = gf_G(x). \end{aligned}$$

□

Sei $G = G^{(0)} \geq \cdots \geq G^{(r)} = \{\mathrm{id}\}$, $\mathcal{C}an_Y$ ein System kanonischer Transversalen auf Y samt Kanonisierer can_Y und $\Phi : G \times \mathcal{L}(G) \to Y^X$ ein Verfeinerer. Auf Y^k seien Kanonisierer zu $\mathcal{C}an_{Y^k}$ gemäß 4.1.6 induziert. Wir konstruieren nun eine Folge von $G^{(i)}$-Homomorphismen $f_{i,j}$. Dabei betrachten wir wie oben die Menge der Index-Paare:

$$I := \{(i,j) \mid i \in r, j \in j_i\},$$

wobei die j_i, $i \in r$, während der Konstruktion induktiv bestimmt werden. Die Bezeichnungen $\bar{I}, I_{(i,j)}, k_{(i,j)}$ und $\bar{f}_{i,j}$ seien wie auf Seite 76 eingeführt. Wir setzen rekursiv:

4.1.13 $\quad f_{0,0} := \Phi(\mathrm{id}, G), \qquad$ und für $(i,j) \in \mathbb{N}^2 : \qquad f_{i,j} := (\bar{f}_{i,j})_{G^{(i)}}$

Die j_i, $i \in r$ sind bestimmt durch:

$$j_i := \min\{j \geq 1 \mid f_{i,j} = f_{i,j-1}\}.$$

4.1.14 Bemerkung.

1. Die Indexmenge I ist endlich, d.h. für jedes $i \in r$ gibt es ein j_i mit $f_{i,j_i} = f_{i,j_i-1}$.
2. Für jedes $(i,j) \in I$ ist $f_{i,j}$ ein $G^{(i)}$-Homomorphismus.

Beweis:
1. Sei $i \in r$. und $x \in X$ beliebig. Die Folge von Stabilisatoren der Funktionswerte $G^{(i)}_{\bar{f}_{i,0}(x)}, G^{(i)}_{\bar{f}_{i,1}(x)}, \ldots$ ist monoton fallend, da $\bar{f}_{i,j}(x)$ stets ein Teiltupel von $\bar{f}_{i,j+1}(x)$ ist. Da G endlich ist, gibt es ein minimales $j_{i,x}$ mit $G^{(i)}_{\bar{f}_{i,j_{i,x}-1}(x)} =$

$G^{(i)}_{\bar{f}_{i,j_{i,x}}(x)}$. Da $\bar{f}_{i,j_{i,x}-1}(x)$ im Tupel $\bar{f}_{i,j_{i,x}}(x)$ enthalten ist, haben beide Werte neben gleichem Stabilisator auch gleiches kanonisierendes Element, und damit ist nach Konstruktion $f_{i,j_{i,x}-1}(x) = f_{i,j_{i,x}}(x)$. Die Gleichheit gilt auch für jedes weitere $j \geq j_{i,x}$. Da X endlich ist, ist auch $j_i := \max\{j_{i,x} \mid x \in X\} \in \mathbb{N}$, und es gilt $f_{i,j_i-1} = f_{i,j_i}$.
2. $f_{0,0}$ ist $G^{(0)}$-Homorphismus nach Bemerkung 4.1.10. Sind alle $f_{i',j'}$, $(i',j') < (i,j)$ (insbesondere) $G^{(i)}$-Homomorphismen, so ist auch das Produkt $\bar{f}_{i,j}$ ein $G^{(i)}$-Homomorphismus, und nach Lemma 4.1.12 auch $f_{i,j} = (\bar{f}_{i,j})_{G^{(i)}}$. □

Beispiel: Graphen

Wir erläutern die Konstruktion der $f_{i,j}$ aus dem Verfeinerer anhand der originalen Situation der Kanonisierung von Graphen mit n Punkten. Sei dazu $W(n)$ die Menge aller endlichen Tupel mit Werten aus n, also die Menge aller *Worte* über n:

$$W(n) := \dot{\bigcup_{l\in\mathbb{N}}} n^l \,.$$

Für $\vec{a} \in W(n)$ bezeichne $|\vec{a}|$ die Länge des Wortes. Wir benötigen die G-Menge der geordneten Mengenpartitionen von n.

$$\mathrm{OPar}(n) := \{\mathfrak{p} = (p_0, \ldots, p_{l-1}) \in W(2^n) \mid n = \dot{\bigcup_{i\in l}} p_i\},$$

$$\text{mit} \qquad g\mathfrak{p} := (gp_0, \ldots, gp_{l-1}), \text{ für } \mathfrak{p} = (p_0, \ldots, p_{l-1}) \in \mathrm{OPar}(n)\,.$$

Wir schreiben:

$$\mathfrak{p}(a) = i :\Longleftrightarrow a \in p_i \,.$$

Sortieren wir zu $(g, H) \in G \times \mathcal{L}(G)$ die Bahnen in $H\backslash\backslash n$ derart, dass

$$Ha \leq H\tilde{a} \Leftrightarrow \min(g^{-1}(Ha)) \leq \min(g^{-1}(H\tilde{a}))\,,$$

so erhalten wir eine geordnete Mengenpartition $(Hg(0), \ldots) \in \mathrm{OPar}(n)$. Die Abbildung

$$\mathrm{orb} : G \times \mathcal{L}(G) \to \mathrm{OPar}(n), (g, H) \mapsto (Hg(0), \ldots)$$

ist ein G-Homomorphismus, es gilt nämlich

$$\mathrm{orb}(hg, hHh^{-1}) = (hHg(0), \ldots) = h(Hg(0), \ldots) = h\mathrm{orb}(g, H)\,.$$

Wir betrachten nun die folgenden induzierten G-Mengen X und Y:

- $X := \mathcal{G}(n) \subseteq 2^{(n^2)}$, die Menge der Graphen mit n Punkten. Es ist (a_1, a_2) genau dann eine Kante im Graphen Γ, wenn $\Gamma(a_1, a_2) = 1$. Die Gruppenoperation ist gegeben durch:

$$g\Gamma(a_1, a_2) := \Gamma(g^{-1}a_1, g^{-1}a_2)\,.$$

- $Y := \mathcal{D}eg(n) := W(\mathbb{N})^n$, die Menge aller Abbildungen, die einem Knoten ein endliches Tupel von Zahlen zuordnen. Zu $y \in W(\mathbb{N})^n$, $a \in n$, $g \in G$ sei:

$$(gy)(a) := y(g^{-1}a)\,.$$

Wir betrachten in $Y^X = \mathcal{D}eg(n)^{\mathcal{G}(n)}$ speziell die Abbildungen $\deg_{\mathfrak{p}}$, $\mathfrak{p} \in \mathrm{OPar}(n)$, welche einem Graphen die verfeinerte Knotengradfolge zur Partition $\mathfrak{p}$ zuordnen, d.h. zu $\Gamma \in \mathcal{G}(n)$, $a \in n$, sei $\deg_{\mathfrak{p}}(\Gamma)(a)$ das Wort der Länge $l := |\mathfrak{p}|$, dessen i-te Komponente, $i \in l$, die Anzahl der Nachbarn von a im i-ten Block von $\mathfrak{p}$ angibt:

$$(\deg_{\mathfrak{p}}(\Gamma)(a))_i := |\{\tilde{a} \in \mathfrak{p}^{-1}(i) \mid \Gamma(a, \tilde{a}) = 1\}|\,.$$

4.1.15 Satz. *Die Abbildung*

$$\deg : \mathrm{OPar}(n) \to \mathcal{D}eg(n)^{\mathcal{G}(n)}, \mathfrak{p} \mapsto \deg_{\mathfrak{p}}$$

ist ein G-Homomorphismus, d.h. $\deg_{g\mathfrak{p}} = g \star \deg_{\mathfrak{p}}$.

Beweis: Die Behauptung ist genau dann erfüllt, wenn für beliebiges $\Gamma \in \mathcal{G}(n)$ gilt: $\deg_{g\mathfrak{p}}(\Gamma) = g \deg_{\mathfrak{p}}(g^{-1}\Gamma)$, d.h. wenn für $a \in n$ gilt:

$$\deg_{g\mathfrak{p}}(\Gamma)(a) = \deg_{\mathfrak{p}}(g^{-1}\Gamma)(g^{-1}a)\,.$$

Wir zeigen äquivalent, dass die i-ten Komponenten, $i < |\mathfrak{p}|$, für folgende Wörter gleich sind:

$$\deg_{g\mathfrak{p}}(g\Gamma)(ga)_i = \deg_{\mathfrak{p}}(\Gamma)(a)_i\,.$$

Die linke Seite ist nach Definition gleich

$$|\{\tilde{a} \in (g\mathfrak{p})^{-1}(i) \mid (g\Gamma)(ga, \tilde{a}) = 1\}|\,.$$

Da $\tilde{a} \in (g\mathfrak{p})^{-1}(i) \iff g^{-1}\tilde{a} \in \mathfrak{p}^{-1}(i)$, und $(g\Gamma)(ga, g\tilde{a}) = \Gamma(a, \tilde{a})$, ist dies gleich

$$|\{\tilde{a} \in \mathfrak{p}^{-1}(i) \mid (\Gamma)(a, \tilde{a}) = 1\}|\,,$$

also dem Wert der rechten Seite. □

Der Verfeinerer Φ ist schließlich wie folgt gegeben:

4.1.16 Satz. *Folgende Abbildung ist ein Verfeinerer für die Menge $\mathcal{G}(n)$ aller Graphen mit n Punkten:*

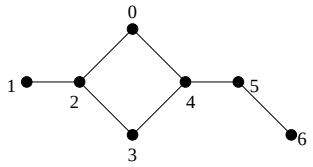
$$\Phi : G \times \mathcal{L}(G) \to Y^X, (g, H) \mapsto \deg_{orb(g,H)} .$$

Beweis: Das Φ ist G-Homomorphismus, da Komposition von G-Homomorphismen: $\Phi = \deg \circ \text{orb}$. Ist $\mathfrak{p} := \text{orb}(g, H)$ und $h \in H$, so gilt $h\mathfrak{p} = \mathfrak{p}$, also

$$h \star \Phi = h \star \deg_{\text{orb}(g,H)} = \deg_{h\text{orb}(g,H)} = \deg_{\text{orb}(g,H)} = \Phi .$$

□

Zur Veranschaulichung betrachten wir ein explizites Beispiel.



Beispiel. Sei Γ der abgebildete Graph mit 7 Knoten. Wir führen den hier besprochenen Algorithmus zur Kanonisierung über $G := S_n$ gemäß iterierten Verfeinerung vor und zeigen den Zusammenhang zur klassischen Version auf. Es ist $f_{0,0} := \deg_{\text{orb}(\text{id},S_n)} = \deg$ die übliche Knotengradfunktion, welche einem Graphen seine Knotengradfolge zuordnet. Es ist:

$$y_{0,0} := f_{0,0}(\Gamma) = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ 2 & 1 & 3 & 2 & 3 & 2 & 1 \end{pmatrix}$$

Wir kanonisieren (d.h. hier: sortieren) y_0 und wir erhalten mit $(g_c, H_c) := \mathit{can}(G, y_{0,0})$ ein kanonisierendes Element sowie den Stabilisator zu $y_{0,0}$:

$$\begin{aligned} g_c &= (0\,2\,5\,4\,6\,1)(3)\,, \\ H_c &= S_{\{0,3,5\}} \times S_{\{1,6\}} \times S_{\{2,4\}}\,. \end{aligned}$$

Wir erhalten einen $(G, 0, 1)$-semikanonischen Graphen $g_0\Gamma$ sowie alle $(G, 0, 1)$--semikanonischen Graphen in $G^{(0)}x$ als $H^{(0,0)}_{g_0\Gamma}$-Bahn, mit

$$\begin{aligned} g_{0,0} &:= g_c = (0\,2\,5\,4\,6\,1)(3)\,, \\ H^{(0,0)}_{g_{0,0}\Gamma} &:= H_c^{g_c} = S_{\{0,1\}} \times S_{\{2,3,4\}} \times S_{\{5,6\}}\,. \end{aligned}$$

Üblicherweise wird die Knotenmenge in die Urbildbereiche unter $\deg(\Gamma) = f_{0,0}(\Gamma)$ partitioniert, und die Blöcke der Partition werden gemäß der Bilder unter $\deg(\Gamma)$ angeordnet:

$$\mathfrak{p}_0 := (\{1,6\},\{0,3,5\},\{2,4\})\,.$$

Man kann sich leicht überzeugen, dass

$$g_{0,0}\mathfrak{p}_0 = \mathrm{orb}(\mathrm{id}, H^{(0,0)}_{g_{0,0}\Gamma}) = (\{0,1\},\{2,3,4\},\{5,6\})$$

gilt. Es sei angemerkt, dass die Speicherung der geordneten Partition $\mathfrak{p}_0$ im Fall von Graphen äquivalent zu dem Paar $(g_{0,0}, H^{(0,0)}_{g_{0,0}\Gamma}) \in G \times \mathcal{L}(G)$ ist, da hier der Stabilisator $H^{(0,0)}_{g_{0,0}\Gamma}$ stets eine Young-Untergruppe ist. Dies ist aber in allgemeineren Situationen nicht mehr der Fall.

Es ist weiter:

$$f_{0,1}(g_{00}\Gamma) := \Phi(\mathrm{id}, H^{(0,0)}_{g_{0,0}\Gamma})(g_{00}\Gamma) = \deg_{(\{0\,1\},\{2\,3\,4\},\{5\,6\})}(g_{00}\Gamma)\,.$$

(Da $g_{0,0}\Gamma$ bereits $(H,0,1)$-semikanonisch ist, ist die Identität ein kanonisierendes Element.)

$$y_{0,1} := f_{0,1}(g_{0,0}\Gamma) = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ (0\,0\,1) & (0\,1\,0) & (0\,0\,2) & (0\,0\,2) & (1\,0\,1) & (1\,2\,0) & (0\,3\,0) \end{pmatrix}$$

Mit $(g_c, H_c) = \mathit{can}(H^{(0,0)}_{g_0\Gamma}, y_{0,1})$ erhalten wir

$$\begin{aligned} g_c &= (5\,6) \\ H_c &= S_{\{2,3\}} \end{aligned}$$

sowie

$$\begin{aligned} g_{0,1} := g_c g_{0,0} &= (0\,2\,6\,1)(3)(4\,5) \\ H^{(0,1)}_{g_{0,1}\Gamma} := H_c^{g_c} &= S_{\{2,3\}} \end{aligned}$$

Im originalen Algorithmus für Graphen wird die Partition $\mathfrak{p}_0$ verfeinert. Dies geschieht, indem man jeden Block anhand der Nachbarschaftsbeziehungen in Γ zu den übrigen Blöcken aus $\mathfrak{p}_0$ aufteilt.

$$\mathfrak{p}_1 := (\{1\},\{6\},\{0,3\},\{5\},\{4\},\{2\})$$

Es gilt jedoch: $g_{0,1}\mathfrak{p}_1 = \mathrm{orb}(\mathrm{id}, H^{(0,1)}_{g_{0,1}x})$.

Der nächste Schritt:

$$f_{0,2} := \Phi(\mathrm{id}, H^{(0,1)}_{g_{0,1}\Gamma})(g_{0,1}\Gamma) = \deg_{(\{0\},\{1\},\{2,3\},\{4\},\{5\},\{6\})}(g_{0,1}\Gamma)$$

$$y_{0,2} := f_{0,2}(g_{0,1}\Gamma) = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ (000001) & (000100) & (000011) & (000011) & (010010) & (002100) & (102000) \end{pmatrix}$$

Es stellt sich während der Kanonisierung $(g_c, G_c) = \mathit{can}(H^{(0,1)}_{g_{0,1}\Gamma}, y_{0,2})$ heraus, dass $G_c = H^{(0,1)}_{g_{0,1}\Gamma}$ ist (und demnach $g_c = \mathrm{id}$), d.h. ein weiterer Schritt bringt keine Änderungen. Es wird im Backtrack-Baum die nächste Ebene besucht. Da der Zusammenhang zu Graph-Kanonisierern bereits hinreichend aufgezeigt ist, sparen wir uns den weiteren Verlauf des Algorithmus.

Der Fall $X = W^{(n^k)}$

Wie bei Graphen lässt sich eine verallgemeinerte Gradfunktion auch für diskrete Strukturen der Form $W^{(n^k)}$ angeben:

- Es sei $X \subseteq W^{(n^k)}$ die Menge der zu kanonisierenden diskreten Strukturen.
- Wir setzen:
$$Y := \mathcal{D}eg_k(n) := W(\mathbb{N})^{(n^{k-1})}\,.$$

Wir betrachten in Y^X speziell die Abbildungen $\deg_{\mathfrak{p}}$, $\mathfrak{p} \in \mathrm{OPar}(n)$, welche einem $x \in X$ die verallgemeinerte Gradfolge zur Partition $\mathfrak{p}$ zuordnen. Zu $\vec{a} \in n^{k-1}$, $a \in n$ und $i \in k$ sei

$$\vec{a} \underset{i}{\leftarrow} a := (a_0, \ldots, \underset{\substack{\uparrow \\ i\text{-te Stelle}}}{a}, \ldots, a_{k-1}) \in n^k\,.$$

Damit sei für $f \in W^{(n^k)}$ der verallgemeinerte Grad $\deg_{\mathfrak{p}}(f)(\vec{a})$ eines $\vec{a} \in n^{k-1}$ definiert als ein Wort der Länge $|\mathfrak{p}| \cdot |W|$, auf deren Komponenten durch Paare (i,w), $i \in |\mathfrak{p}|$, $w \in W$, zugegriffen wird. Dabei sei die (i,w)-te Komponente

$$(\deg_{\mathfrak{p}}(f)(\vec{a}))_{(i,w)} := |\{(\tilde{a}, j) \in \mathfrak{p}^{-1}(i) \times k \mid f(\vec{a} \underset{j}{\leftarrow} \tilde{a}) = w\}|\,.$$

Man kann für $G \leq S_n$ analog zu Satz 4.1.15 zeigen, dass dies ein G-Homomorphismus ist. Entsprechend erhalten wir einen Verfeinerer

$$\Phi : G \times \mathcal{L}(G) \to Y^X, (g, H) \mapsto \deg_{\mathrm{orb}(g,H)} .$$

Eine noch offene Frage ist, wie die hier erarbeitete Theorie mit der Arbeit von Frau Wang [Wan] in Zusammenhang steht.

Abschneiden von Teilbäumen: Lexikographische Anordnung, Iterative Deepening, Automorphismen

Wir führen kurz weitere Strategien vor, um aus dem Backtrack-Baum Teile abzuschneiden. Diese sind größtenteils bereits von Graph-Kanonisierern bekannt, bis auf die in Abschnitt 3.3 besprochene Ausnutzung einer Linkstransversale der Automorphismengruppe gemäß den Ideen von M. Jerrum. Ob das Prinzip des Iterative Deepening bei Graphkanonisierern bisher benutzt wurde, ist mir nicht bekannt.

Operiert $G \leq S_n$ auf $X \subseteq W^{(n^k)}$, und ist $G = G^{(0)} \geq \cdots \geq G^{(n)} = \{\mathrm{id}\}$ die Stabilisatorkette, so kann man bekanntlich, bei geeigneter lexikographischer Anordnung auf X, von dem durchlaufenen Backtrack-Baum Teile abschneiden. Dabei seien die Komponenten eines n^k-Tupels derart angeordnet, dass die k--Tupel über n revers lexikographisch durchlaufen werden (zur Definition von revers lexikographisch, siehe Seite 104). Im Fall von Matrizen $x \in W^{(n^2)}$ werden so zunächst die oberen $i \times i$-Teilmatrizen verglichen, bevor die Einträge der $i+1$-ten Zeilen/Spalten berücksichtigt werden.

Wir führen für $x_1, x_2 \in W^{(n^k)}$ folgende Schreibweise ein:

$$\begin{aligned}
x_1 \sim_i x_2 &:\Longleftrightarrow \forall \vec{a} \in i^k : x_1(\vec{a}) = x_2(\vec{a}) , \\
x_1 <_i x_2 &:\Longleftrightarrow x_1 \sim_i x_2 \wedge \begin{cases} x_1, x_2 \text{ unterscheiden sich erstmalig} \\ \text{an einem Tupel } a \in (i+1)^k \text{ mit} \\ \qquad x_1(\vec{a}) < x_2(\vec{a}) \end{cases} , \\
x_1 \leq_i x_2 &:\Longleftrightarrow x_1 = x_2 \text{ oder } x_1 <_{i'} x_2 \text{ mit } i \leq i' < n .
\end{aligned}$$

Damit gilt: $x_1 \leq x_2 \iff x_1 \leq_0 x_2 .$

Weiter schreiben wir für zwei $G^{(i)}$-Bahnen:

$$G^{(i)}x_1 \leq G^{(i)}x_2 :\Longleftrightarrow \forall g_1, g_2 \in G^{(i)} : g_1 x_1 \leq g_2 x_2 .$$

Gilt für zwei Knoten des Backtrack-Baums auf Ebene $i \geq 1$ mit Beschriftungen g_i und $\tilde{g}_i$ die Ungleichung $g_i x <_{i'} \tilde{g}_i x$ mit $i' < i$, so folgt für die entsprechenden $G^{(i)}$-Bahnen: $G^{(i)} g_i x < G^{(i)} \tilde{g}_i x$, und der Teilbaum unterhalb $\tilde{g}_i$ kann übersprungen werden. Wir können also auf jeder Ebene zunächst die optimalen $G^{(i)}$-Bahnen $G^{(i)} g_i x$ bestimmen und so auf Ebene $i+1$ nur die Knoten berücksichtigen, deren Vater g_i zu einer optimalen $G^{(i)}$-Bahn $G^{(i)} g_i x$ gehört.

Obwohl diese Idee schon lange bekannt ist, wird sie in bisherigen Algorithmen zur Graph-Kanonisierung oft nur zum Teil ausgenutzt, da man im Backtrack--Algorithmus die optimalen Knoten auf Ebene i nicht rechtzeitig bestimmen kann. Eine Breitensuche wäre in dieser Hinsicht geeigneter, jedoch ist dies aufgrund der Größe der zu durchforstenden Bäume unpraktikabel. Als Lösung bietet sich an, den Baum iterativ mehrmals zu durchlaufen, und in jedem Durchlauf eine Ebene weiter vorzudringen. In Durchlauf i werden also nur die Knoten bis zur Ebene i besucht und das Optimum der erreichten $g_i x$ bzgl. $<_{i-1}$ bestimmt. So kann man bei den darauffolgenden Durchläufen alle auf Ebene i nicht optimalen Knoten überspringen.

Dieses als *Iterative Deepening* bekannte Vorgehen benötigt erstaunlicherweise kaum mehr theoretischen Aufwand als eine Breitensuche. (Für Bäume mit einer festen Anzahl k an Söhnen für jedes Nicht-Blatt geht der Aufwand für große Baumtiefen gegen das $\frac{k}{k-1}$-fache des Aufwands einer Breitensuche, siehe [Nil03, Seiten 135f].)

Ist weiter $\langle S \rangle \leq G_x$ eine Gruppe von bekannten Automorphismen von x (S wird während des Algorithmus dynamisch ergänzt, sodass am Ende $\langle S \rangle = G_x$ gilt), so reicht es, wenn im Backtrack-Durchlauf eine Transversale von $G/\langle S \rangle$ durchlaufen wird, da für $s \in \langle S \rangle$ gilt: $gsx = gx$. Wir haben in Abschnitt 3.3 gesehen, dass durch das vollständige Labelled Branching $\Gamma := \Gamma(X, \vec{b}, \tilde{S})$ zu $\langle S \rangle$ die Transversale $T_{\min}$ von G/S genau durch die topologischen Anordnungen zu Γ bestimmt ist. Hilfssatz 3.3.3 on page 63 gibt ein notwendiges Kriterium dafür, ob die Suche im Teilbaum unterhalb von g_i zu Transversalenelementen führt.

Wir speichern während des Algorithmus neu gefundene Automorphismen in S ab: Ist g_{opt} der bisher optimale Kandidat, und gilt auf Ebene i: $g_{\text{opt}} x \sim_{i-1} g_i$, mit $a := g_{\text{opt}}^{-1} g_i \in G_{(x_i, \ldots, x_{n-1})}$, so ist $a \in G_x$, und wir fügen a zu S als weiteren Erzeuger hinzu (per *sift*$()$; das Γ wird modifiziert zu einem vollständigen Labelled Branching zu $\langle S' \rangle$). Man kann direkt auf Ebene $i'-1$ zurückspringen, mit $(i', j) = \mathrm{typ}(a)$ (Lerneffekt), denn der Teilbaum, der g enthält, ist isomorph zu den bereits durchlaufenen Teilbaum mit g_{opt}.

Implementierung des Algorithmus

Der Algorithmus ist auf den Seiten 90 – 91 als Pseudocode formuliert.

Dabei ist zunächst zu beachten, dass eine Stabilisatorkette als Untergruppenkette von $G \leq S_n$ vorausgesetzt wird, und Erzeugendensysteme von Untergruppen $H \leq G$ als entsprechende Rechtstransversalenketten implementiert sind. In den Funktionen canonize() und refine() erscheinen Variablen E als Erzeugendensysteme, dahinter verbirgt sich aber stets eine Kette $(T_0, \ldots, T_{n-1})$ von Transversalen. In der Funktion canstep(), welche den eigentlichen Backtrack-Baum implementiert, werden diese Transversalenketten explizit benötigt.

Die Startfunktion canonize() implementiert das Prinzip des Iterative Deepening (siehe Seite 87). Als Eingabe erhält die Funktion eine diskrete Struktur $x \in X$ sowie ein Erzeugendensystem E (eine Rechtstransversalenkette!) der operierenden Gruppe. Optional kann eine Gruppe von Automorphismen S angegeben werden, und zwar als vollständiges Labelled Branching (bzgl. der Linkstransversalen!). Ansonsten sei $S := \{\mathrm{id}\}$ vorinitialisiert. Während des Durchlaufs der for-Schleife in den Zeilen 3–5 wird mit der Variablen g stets ein bis zur Ebene $i-1$ optimaler Kandidat an die Unterfunktion canstep() übergeben, samt aktueller Abbruch--Ebene i. Als Rückgabe wird g mit einem optimalen Kandidaten bis zur Ebene i überschrieben. Weiter enthält S nun zusätzlich von canstep() gefundene Automorphismen, sodass in den nachfolgenden Backtrack-Läufen diese Information bereits von Anfang an zur Verfügung steht. Die Funktion gibt schließlich nach Erreichen der Ebene n ein kanonisierendes Element $g \in G$ zurück, samt einem vollständigen Labelled Branching S der Automorphismengruppe G_x.

Die Funktion canstep() implementiert den eigentlichen Backtrack-Baum. Dazu erhält er neben der Benutzereingabe E, x und S auch die Informationen des Iterative Deepening, und zwar die Ebene r_0, auf welcher der Baum abgeschnitten werden soll, sowie den bisherigen optimalen Kandidaten g_{opt} (optimal bis zur Ebene $r_0 - 1$). Der Baum wird wie auf Seite 78 besprochen durchlaufen: Die Wurzel wird in den Zeilen 2 – 3 behandelt, die restlichen Knoten des Backtrack-Baums innerhalb der for-Schleife. In den Zeilen 6 – 7 wird die Knotenbeschriftung berechnet, d.h. auf Ebene i wird ein Repräsentant aus einer $H_{g_i x}^{(i,0)}$-Bahn gewählt. Nach einer Reihe von Tests zum Abschneiden in den Zeilen 8 – 19 wird in Zeile 21 (bzw. Zeile 2 für die Wurzel) bzgl. der $G^{(i)}$-Homomorphismen $f_{i,j}$, $j \in j_i$, kanonisiert, und g_i wird modifiziert, sodass g_i Repräsentant einer $H_{g_i x}^{(i,j_i)}$-Bahn ist, mit kanonischen $(f_{i,j})_{(i,j)}$-Bildern. Des weiteren wird in $E_i := (T_{i,i}, \ldots, T_{i,n-1})$ eine Transversalenkette für $H_{g_i x}^{(i,j_i)}$ gespeichert. Dabei ist $T_{i,i}$ genau eine Trans-

versale von $H_{g_i x}^{(i+1,0)} \backslash H_{g_i x}^{(i,j_i)}$, wir können also alle Söhne von g_i auf der nächsten Ebene durchlaufen (Zeilen 22 bzw. 3).

In den Zeilen 8 – 19 werden Teile des Baums abgeschnitten: Zeilen 8 – 10 setzen Hilfssatz 3.3.3 um, d.h. hier wird der Baum auf eine Transversale von G/S zurechtgestutzt. In Zeile 12 werden Teile abgeschnitten, die aufgrund vorheriger Läufe des Iterated Deepening bereits als nicht optimal erkannt sind. In Zeilen 13 – 17 werden schließlich neue Automorphismen erkannt und gegebenenfalls zu S hinzugefügt. Wie auf Seite 87 besprochen, kann man in diesem Fall sogar auf eine höhere Ebene zurückspringen. Zeile 18 hält den optimalen Kandidaten g_{opt} aktuell, und Zeile 19 schneidet den Backtrack-Algorithmus auf der durch das Iterated Deepening vorgegebenen Ebene ab.

Die Funktion refine() implementiert schließlich die iterative Verfeinerung. Im j--ten Schleifendurchlauf, $j \geq 0$, wird $f_{i,j}(gx) = \Phi(\mathrm{id}, H_{gx}^{(i,j)})(gx)$ kanonisiert (Zeile 4). Aus der gewonnenen Information wird in Zeile 6 g derart modifiziert, dass es $(H, i, j+1)$-kanonisch ist, und E wird zu einem Erzeugendensystem von $H_{gx}^{(i,j+1)}$. Somit sind die Voraussetzungen für den nächsten Schleifendurchlauf gewährt. Die Schleife bricht ab, sobald $H_{gx}^{(i,j)} = H_{gx}^{(i,j+1)}$ ist. Dann gilt $j = j_i$, und es wird ein (H, i, j_i)-semikanonisches Element samt Erzeugendensystem von $H_{gx}^{(i,j)}$ zurückgegeben.

Während der Prozedur muss die Gruppe von Automorphismen S stets per Basiswechsel angepasst werden, sodass der Kanonisierer can_Y diese Information zur Kanonisierung von gx nutzen kann. Da der Basiswechsel selbst auch einen nichttrivialen Aufwand benötigt, kann man auf diese Information u.U auch verzichten, falls die Kanonisierung in can_Y schnell geht (etwa Sortieren bei $Y = W^n$).

Der Verfeinerer $\Phi : G \times \mathcal{L}(G) \to Y^X$ sowie ein Kanonisierer auf Y sind zur Implementierung in geeigneter Weise zur Verfügung zu stellen.

Variationen des Algorithmus

Es gibt weitere Strategien zur Beschleunigung der Kanonisierung, auf die hier nicht näher eingegangen wurde. So kann man beispielsweise durch wiederholten Basiswechsel im Backtrack-Durchlauf dafür sorgen, dass stets zunächst eine kleinste Transversale behandelt wird. Weiter kann man auf die aufwändige iterierte Verfeinerung verzichten, sobald die Bahnen klein genug sind. Solche Strategien sollten natürlich auch hier umgesetzt werden.

Algorithmus 4.2 *canstep*$(E, x, r_0, g_{\text{opt}}, S)$

Eingabe:
$E = \langle T_0, T_1 \ldots, T_{n-1}\rangle$, Rechtstransversalen der Stabilisatorkette von G
$x \in X$, eine diskrete Struktur
r_0, durchlaufe Backtrack-Baum nur bis zur Ebene r_0 (einschließlich)
g_{opt}, es ist $g_{\text{opt}}x$ optimal bis zur Ebene $r_0 - 1$
$S \leq G_x$, Gruppe von Automorphismen, als vollst. Lab. Branching

Ausgabe:
$g \in G$, sodass gx optimal bis zur Ebene r_0 ist
Zu S werden evtl. Automorphismen hinzugefügt.

```
1  begin
2    (g, ⟨T_{0,0}, T_{0,1} ..., T_{0,n-1}⟩) := refine(⟨T_0, ..., T_{n-1}⟩, id, x, S);
3    T'_0 = T_{0,0};                          //Die Transversale für die erste Ebene.

4    for (Durchlaufe Backtrack-Baum zu ⟨T'_0, ..., T'_{n-1}⟩ ohne Wurzel) do
5      // Die Transversalen T'_i werden während des Durchlaufs bestimmt.
6      i := (Ebene im Baum); j := (Index der eingehenden Kante);
7      g_i := T'_{i-1}[j] · g_{i-1};

8      // Teilbaum abschneiden nach Hilfssatz 3.3.3:
9      k := g_i^{-1}(i − 1);
10     if S.father_k ≥ 0 ∧ g_i(S.father_k) ≥ i then continue on level i; fi
11     // Abschneiden nicht-optimaler bzw. isomorpher Teilbäume:
12     if g_i x >_{i-1} g_opt x then continue on level i; fi
13     if g_i x ∼_{i-1} g_opt x ∧ i = r_0 ∧ ∀i ≤ j < n : g_i(j) = g_opt(j) then
14       S.sift(g_opt^{-1} g_i); S.complete();        // neuer Automorphismus
15       j := max{j' ≤ i | g_opt^{-1} g_i ∈ G_{j'}}
16       continue on level j;
17     fi
18     if g_i x <_{i-1} g_opt x then g_opt = g_i fi
19     if i = r_0 then continue on level i; fi                  // Letzte Ebene

20     // kanonisiere bzgl. (f_{i,j})_{j∈j_i}, Transversale für nächste Ebene:
21     (g_i, ⟨T_{i,i}, ..., T_{i,n-1}⟩) := refine(⟨T_{i-1,i}, ..., T_{i-1,n-1}⟩, g_i, x, S);
22     T'_i = T_{i,i};
23     continue on level i + 1;
24   od
25   return (g_opt, S);
26 end
```

Algorithmus 4.1 *canonize*(E, x, S)

Eingabe:
$E = \langle T_0, T_1 \ldots, T_{n-1} \rangle$, Rechtstransversalen der Stabilisatorkette von G
$x \in X$, eine diskrete Struktur
$S \leq G_x$, Gruppe von Automorphismen, als vollst. Lab. Branching (optional)

Ausgabe:
$g \in G$, kanonisierendes Element, d.h. $gx \in \mathcal{C}an(G)$
G_x, die volle Automorphismengruppe, als vollst. Lab. Branching

```
1 begin
2   g := id
3   for i = 1 to n do
4       (g, S) := canstep(E, x, i, g, S);
5   od
6   return (g, S);
7 end
```

Algorithmus 4.3 refine(E, g, x, S) – iterierte Verfeinerung

Voraussetzung:
$\Phi : G \times \mathcal{L}(G) \to Y^X$, ein Verfeinerer. (Es sei $\Phi(g, E) := \Phi(g, \langle E \rangle)$.)
$can_Y()$, ein Kanonisierer auf Y.

Eingabe:
$E = \langle T_i, \ldots, T_{n-1} \rangle$, Rechtstransversalen der Stabilisatorkette von H
$g \in G$, Knotenbeschriftung
$x \in X$, eine diskrete Struktur; wir kanonisieren in Hgx.
$S \leq G_x$, Gruppe von Automorphismen, als vollst. Lab. Branching

Ausgabe:
g', sodass $g'x$ semikanonisch ist in Hgx bzgl. Φ.
E', Rechtstransversalen der Gruppe H', sodass $H'g'x$ Menge der semikanonischen Elemente ist.

```
1 begin
2   S = S^g;                                          //Basiswechsel
3   repeat
4     (g', E') := can_Y(E, Φ(id, E)(gx), S);
5     if E = E' then return (g, E); fi
6     g := g'g;  E := E'^{g'};  S := S^{g'};      // Basiswechsel für E und S
7   forever;
8 end
```

An dieser Stelle noch eine Bemerkung zu dem verwandten Problem der Berechnung des Stabilisators G_x: In diesem Fall kann man im Backtrack-Baum noch weitere Zweige abschneiden. Nämlich jedesmal dann, wenn auf Ebene i am Knoten g_i gilt: $g_i x <_{i-1} x$ (zusätzlich zu den bereits besprochenen Fall, dass $g_i x >_{i-1} x$ ist; der Kandidat g_{opt} bleibt dadurch stets die Identität). Aufgrund dieses Argumentes ist die Bestimmung der Automorphismengruppe weniger aufwändig als die Kanonisierung. Es bleibt zu testen, ob es sich für die Kanonisierung generell lohnt, zunächst in einem Backtrack-Durchlauf G_x zu bestimmen, und anschließend mit bereits bekannter Automorphismengruppe zu kanonisieren. Die Kanonisierung selbst würde effizienter, da nur noch eine Transversale von G/G_x durchlaufen wird. Es ist jedoch fraglich, ob der Geschwindigkeitsgewinn den Mehraufwand kompensiert.

4.2 Konstruktion

Ist X G-Menge und $\mathcal{C}an$ ein System kanonischer Transversalen, so kann man die Transversale $\mathcal{C}an(G)$ bestimmen, indem man für alle $x \in X$ einen Kanonizitätstest durchführt und genau die kanonischen Elemente speichert. Eine Strategie zur Effizienzsteigerung ist dann, die Anzahl der x zu reduzieren, für welche der Kanonizitätstest durchgeführt werden muss.

Ist beispielsweise X von der Form $W^{(n^k)}$ und (bzgl. einer Anordnung der $\vec{a} \in n^k$) lexikographisch angeordnet, so bietet die *ordnungstreue Erzeugung* nach Faradzhev [Far78a] und Read [Rea78] für den Fall $\mathcal{C}an = \mathcal{C}an_{\text{min}}$ Lerneffekte an. (Bzgl. der Schreibweise zur lexikographischen Ordnung, siehe Seite 110.) Wurde erkannt, dass ein $x \in X$ nicht minimal in seiner Bahn ist, etwa $gx <_{\vec{a}} x$ für ein $g \in G$, so sind alle weiteren x', für die sowohl $x \sim_{\vec{a}^+} x'$ als auch $gx \sim_{\vec{a}^+} gx'$ gilt, ebenfalls nicht minimal in ihrer Bahn, denn es folgt $gx' <_{\vec{a}} x'$. Dabei bezeichne $\vec{a}^+$ den Nachfolger von $\vec{a}$ bzgl. der gegebenen Anordnung von n^k. Werden die $x \in X$ in lexikographischer Reihenfolge durchlaufen, so kann man also alle x' mit $x \sim_{\vec{a}_l^+} x'$ überspringen, mit

$$\vec{a}_l := \max(\{\vec{a}\} \cup \{g^{-1}\vec{a}' \mid \vec{a}' \leq \vec{a} \wedge x_{g^{-1}\vec{a}'} \text{ ist nicht maximal}\}).$$

Damit kann man zwar die Anzahl der Kanonizitätstests stark reduzieren, jedoch ist die Kanonisierung bzgl. $\mathcal{C}an_{\min}$, d.h. eine Minimierung in der jeweiligen Bahn,

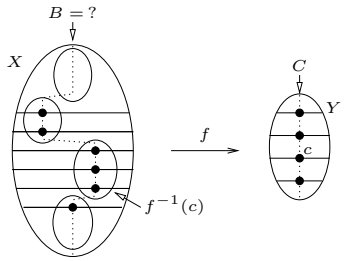


Abbildung 4.3: Der Abwärtsschritt des Leiterspiels

wesentlich aufwändiger als eine Kanonisierung nach den Strategien des vorangehenden Abschnitts. Konstruktion mittels Leiterspiel nach Schmalz [Sch93, Lau93] ist nicht auf $\mathcal{C}an_{\min}$ beschränkt, und die Anzahl der zu kanonisierenden Elemente wird mittels Homomorphieprinzip reduziert. Erfahrungen bei anderen diskreten Strukturen wie Graphen ([McK98]) und Designs ([KÖTZ, Kas]) belegen , dass diese Strategie der ordnungstreuen Erzeugung weit überlegen ist. Wir skizzieren kurz die zwei im Leiterspiel betrachteten Situationen:

4.2.1 Satz. (Homomorphieprinzip: Abwärtsschritt, Laue '82)
Seien X, Y G-Mengen, $\mathcal{C}an_X$ ein System kanonischer Transversalen auf X, C eine Transversale auf Y, und $f : X \to Y$ ein G-Homomorphismus. Dann ist die folgende Menge B eine Transversale auf X:

$$B := \{x \in X \mid f(x) = c \in C \wedge x \in \mathcal{C}an_X(G_c)\}.$$

Zur Veranschaulichung des Satzes siehe auch Abbildung 4.3. Man kann B berechnen, wenn man für alle $c \in C$ die Stabilisatoren kennt, und einen Kanonisierer auf X zur Verfügung hat. Dabei sind nur die Elemente aus $f^{-1}(C)$ zu kanonisieren, und es wird nur bzgl. der Stabilisatoren G_c, $c \in C$ kanonisiert. Bei den Kanonizitätstests erhält man gleichzeitig die Stabilisatoren G_b der $b \in B$.

Auch die umgekehrte Richtung ist von Interesse (vgl. Abbildung 4.4): Ist ein $f : X \to Y$ surjektiv, und ist auf X eine Transversale B samt Stabilisatoren G_b, $b \in B$ bekannt, so können wir eine Transversale C auf Y aus den Bildern $f(b)$, $b \in B$ bestimmen. Das einzige Problem dabei ist, eine unabhängige Teilmenge des Erzeugendensystems $f(B) \subseteq Y$ auszuwählen.

B. Schmalz benötigt für eine algorithmische Lösung dieses Problems die gesamte Transversale B im Speicher. Zu einem $f(b)$, $b \in B$, wird dann das Urbild

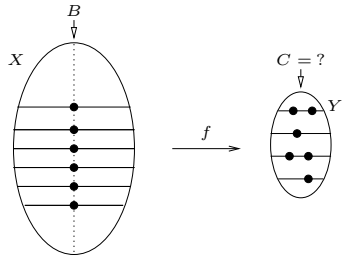


Abbildung 4.4: Der Aufwärtsschritt des Leiterspiels

$f^{-1}(f(b))$ durchlaufen, und alle Elemente aus $B \cap f^{-1}(f(b))$ werden markiert. (Die Speicherung dieser Transversalen ist zur Zeit beispielsweise der begrenzende Faktor bei der Konstruktion linearer Codes.) Nach McKay [McK98] kann man dies vermeiden: Dazu wird eine weitere Abbildung $\tilde{f} : Y \to X$ benötigt mit:

1. $\tilde{f}$ ist eine Rechtsinverse zu f:
$$f \circ \tilde{f}(y) = y\,.$$
2. Es gilt:
$$\tilde{f}(gy) \in G_{gy} g \tilde{f}(y)\,.$$
(D.h. die induzierte Abbildung $Y \to 2^X, y \mapsto G_y \tilde{f}(y)$ ist ein G-Homomorphismus.)

Es gilt nämlich:

4.2.2 Satz. (Homomorphieprinzip: Aufwärtsschritt, Schmalz '93, McKay '98) *Seien X, Y G-Mengen, B eine Transversale auf X, $f : X \to Y$ ein G-Epimorphismus und $\tilde{f} : Y \to X$ eine Rechtsinverse zu f mit (*) $\tilde{f}(gy) \in G_{gy} g \tilde{f}(y)$. Dann ist wie folgt eine Transversale auf Y definiert:*

$$C := \{f(b) \mid b \in B \wedge \tilde{f}(f(b)) \in G_{f(b)} b\}.$$

Ist $B = \mathcal{C}an_X(G)$ die Transversale zu einem System kanonischer Transversalen $\mathcal{C}an_X$ auf X, so ist insbesondere $b \in \mathcal{C}an(G_{f(b)})$ und die Bedingung $\tilde{f}(f(b)) \in G_{f(b)} b$ wird durch $G_{f(b)}$-Kanonisierung von $\tilde{f}(f(b))$ überprüft (oder bei geeignet gewählten $\tilde{f}$ noch einfacher, siehe unten). Da der Zusammenhang zu [McK98] keineswegs offensichtlich ist, geben wir einen Beweis dieses Satzes an. Anschließend zeigen wir den Zusammenhang zu der von McKay beschriebenen Situation aufgezeigt.

Beweis: Die Menge C ist ein Erzeugendensystem: Da B Transversale ist, gibt es zu jedem $y \in Y$ ein $b \in B$, $g \in G$ mit $g\tilde{f}(y) = b$. Mit der G-Homomorphie von f und der Eigenschaft $f \circ \tilde{f} = \mathrm{id}$ folgt $f(b) = f(g\tilde{f}(y)) = gy$. Mit (*) folgt weiter: $\tilde{f}(f(b)) = \tilde{f}(gy) \in G_{gy}g\tilde{f}(y) = G_{f(b)}b$. Also ist $c := f(b) \in C$ mit $gy = c$.

Die Menge C ist unabhängig: Sind $c, c' \in C$ mit $c' = gc$, $g \in G$, so sind wegen (*) $\tilde{f}(c')$ und $\tilde{f}(c)$ abhängig. Sind $b, b' \in B$ mit $f(b) = c$ und $f(b') = c'$, so sind nach Definition von C einerseits b und $\tilde{f}(c)$ abhängig, und andererseits auch b' und $\tilde{f}(c')$ abhängig, insgesamt also b und b'. Da B unabhängige Menge ist, folgt $b = b'$ und damit $c = c'$. □

Wir erläutern im Folgenden den von B. D. McKay in [McK98] bereits vermuteten, aber bisher nicht ausgearbeiteten Zusammenhang zwischen dem Leiterspiel und seiner eigenen Methode:

McKay betrachtet in besagtem Artikel *graduierte* G-Mengen X, d.h. jedes $x \in X$ habe eine auf den Bahnen invariante Ordnungszahl (einen Grad) $o(x) \in \mathbb{N}$. Wir setzen $X_n := o^{-1}(n)$ für $n \in \mathbb{N}$.

Als Beispiel stelle man sich als X die Menge aller Graphen mit $\leq N$ Knoten vor, und $o(x)$ sei die Anzahl der Knoten eines Graphen $x \in X$. Um eine Gruppenoperation von $G \leq S_N$ zu erhalten, betrachten wir formal eine Menge $\bar{X}$ von Graphen mit N Punkten samt Knotenbeschriftung, indem wir jedem $x \in X$ einen entsprechenden Graphen $\iota(x)$ zuordnen, mit $N - o(x)$ zusätzlichen, paarweise verschieden markierten Punkten vom Knotengrad 0. Wir setzen $\bar{X} := S_N \cdot \iota(X)$. Dann ist $\bar{X}$ eine S_N-Menge, und $\iota : X \to \bar{X}$ induziert offensichtlich eine Bijektion zwischen den S_N-Bahnen in $\bar{X}$ und der Vereinigung der S_n-Bahnen in X_n, $n \in N$: $S_N \backslash\backslash \bar{X} \rightarrowtail \bigcup_{n \leq N} S_n \backslash\backslash X_n$.

Weiter werden zu der graduierten G-Menge X zwei weitere graduierte G-Mengen, die der *oberen Objekte* $\hat{X}$ und die der *unteren Objekte* $\check{X}$ sowie G-Homomorphismen $\hat{f} : \hat{X} \to X$, $\check{f} : \check{X} \to X$ vorausgesetzt. Die G-Homomorphismen seien Grad-erhaltend: $o(\hat{f}(\hat{x})) = o(\hat{x})$, $o(\check{f}(\check{x})) = o(\check{x})$. Die Urbilder zu einem $x \in X$ seien mit $U(x) := \hat{f}^{-1}(x)$, den oberen Objekten *oberhalb* von x, sowie $L(x) := \check{f}^{-1}(x)$, den unteren Objekten *unterhalb* von x bezeichnet. Dabei erwähnt McKay in [McK98] nicht die G-Homomorphismen, sondern definiert die Mengen $U(x)$ und $L(x)$ direkt. Die hier wiedergegebene Situation folgt aber direkt aus den dort gestellten Forderungen der paarweisen Disjunktheit der $U(x)$, $x \in X$, der Eigenschaft, dass die Vereinigung aller $U(x)$ ganz $\hat{X}$ ergeben soll, und der Bedingung C2 in erwähntem Artikel: $U(gx) = gU(x)$, sowie analogen Forderungen für $L(x)$.

Es gebe eine Relation $R \subseteq \check{X} \times \hat{X}$ zwischen unteren und oberen Objekten. Für diese Relation werden eine Reihe von Eigenschaften gefordert, auf diese allgemeine Situation gehen wir hier jedoch nicht näher ein. Ein wichtiger Spezialfall ist gegeben, falls R durch einen G-Isomorphismus $\varphi : \check{X} \to \hat{X}$ spezifiziert ist, mit $o(\varphi(\check{x})) < o(\check{x})$. Wir setzen der Einfachheit halber noch spezieller voraus, dass $o(\varphi(\check{x})) = o(\check{x}) - 1$ gilt.

Wir identifizieren dann die Mengen $\hat{X}$ und $\check{X}$, d.h. die beiden G-Homomorphismen $\hat{f}$ und $\check{f}$ haben den gleichen Definitionsbereich $\check{X}$, wobei $\hat{f} : \check{X} \to X$ jetzt nicht mehr Grad-erhaltend ist, es gilt $o(\hat{f}(\check{x}) = o(\check{x}) - 1$. Betrachten wir an dieser Stelle die auf $\check{X}_n$ eingeschränkten G-Homomorphismen $\hat{f}_n : \check{X}_n \to X_{n-1}$ und $\check{f}_n : \check{X}_n \to X_n$ (für $1 \leq n \leq N$), so erkennt man die Situation des Leiterspiels wieder: Mittels Abwärtsschritt kann man aus einer Transversale von X_{n-1} eine Transversale von $\check{X}_n$ konstruieren, und mittels Aufwärtsschritt weiter eine Transversale in X_n. Die für den Aufwärtsschritt zusätzlich nötige Rechtsinverse $\tilde{f}$ von $\check{f}$ mit Bedingung (*) aus Satz 4.2.2 wurde in [McK98] mit $m : X \to \check{X}$ bezeichnet.

McKay unterteilt weiter X in die Mengen der *irreduziblen* und *reduziblen* Objekte

$$\begin{aligned} X^{\text{irr}} &:= \{x \in X \mid \check{f}^{-1}(x) = \emptyset\}, \\ X^{\text{red}} &:= X \setminus X^{\text{irr}}. \end{aligned}$$

Offensichtlich gilt $X_0 \subseteq X^{\text{irr}}$. (Für $x \in X_0$, $\check{x} \in \check{f}^{-1}(x)$ wäre sonst $o(\check{x}) = 0$, und somit $o(\hat{f}(\check{x})) = -1 \notin \mathbb{N}$.) In den meisten Fällen wird auch die Gleichheit gelten. Die Abbildung

$$p : X^{\text{red}} \to X, x \mapsto \hat{f} \circ \tilde{f}(x)$$

bildet Objekte $x \in X^{\text{red}}$ der Ordnung $n = o(x)$ auf Objekte der Ordnung $n-1$ ab. Die Bilder zweier abhängiger Objekte $x, gx \in X^{\text{red}}$ sind dabei wieder abhängig. (Die induzierte Abbildung $X \to 2^X$, $x \mapsto G_x p(x)$ ist ein G-Homomorphismus.) Somit induziert p eine wohldefinierte Abbildung $\bar{p}$ auf den Bahnen $G\backslash\backslash X$. Fasst man $\bar{p}$ als Vater-Vorschrift auf, so erhält man demnach eine Wald-Struktur auf $G\backslash\backslash X$ mit den Wurzeln $G\backslash\backslash X^{\text{irr}}$. Man kann also alle Transversalen mittels eines Backtrack-Algorithmus (Tiefensuche) entlang diesem Wald konstruieren. Dabei wird iterativ jeweils zu einem $x \in X$ eine Transversale des Urbilds $p^{-1}(x)$ bestimmen, was genau einem Doppelschritt des Leiterspiels entspricht.

Wir geben für solch einen Doppelschritt, einer Kombination eines Abwärts- und eines Aufwärtsschrittes, den Algorithmus 4.4 an. Der Kern davon entspricht im

Wesentlichen dem in [McK98] vorgestellten Algorithmus scan(). In der hier angegebenen Form wird aus einer Transversale von B_{n-1} eine Transversale von $B_n$auf einmal berechnet, d.h. dies entspricht einer Breitensuche. Es stellt jedoch kein Problem dar, diesen Algorithmus zu einer Tiefensuche umzuformulieren.

Anwendung.

Es sei $n \in \mathbb{N}$. Für $i \leq n$ sei $X_i := W^{(i^k)}$, $\mathcal{C}an_i$ sei ein System kanonischer Transversalen auf X_i, und can_i ein zugehöriger Kanonisierer. Wir sind an einer Transversale von X_n interessiert. Sei dazu $X := \bigcup_{i \leq n} X_i$. Die $x \in X$ können als gerichtete Hypergraphen mit Kantenbeschriftungen aus W interpretiert werden.

Um eine gemeinsame Gruppenoperation S_n auf X zu bekommen, seien die Hypergraphen $x \in X_i$ auf n Punkten definiert, jedoch $n - i$ Punkte seien an keiner Hyperkante beteiligt, und haben paarweise disjunkte Markierungen. Die markierten Knoten heißen uneigentliche Knoten von x im Gegensatz zu den i eigentlichen Knoten.

Zu $1 \leq i \leq n$ sei $\check{X}_i \subseteq X_i \times n$, wobei mit $(x, m) \in \check{X}_i$ durch m ein eigentlicher Knoten markiert sei. Das $\check{X} := \bigcup_{i \leq n} \check{X}_i$ ist G Menge auf offensichtliche Weise. Für $(x, m) \in \check{X}_i$ sei $\hat{f}((x, m)) \in X_{i-1}$ derjenige Graph, der durch Entfernen des markierten Knotens m entsteht, während bei $\check{f}((x, m)) := x \in X_i$ lediglich die Markierung selbst entfernt wird. Ist $x \in X_i$, und g_x das kanonisierende Element zu x, so sei in $\tilde{f}(x) := (x, g_x^{-1}(0))$. Offensichtlich sind $\hat{f}$ und $\check{f}$ G-Homomorphismen und $\check{f} \circ \tilde{f} = \mathrm{id}$. Ist $x' = gx$, $x, x' \in X$, so gilt für die kanonisierenden Elemente $g_x, g_{gx} \in G$: $g_{gx}^{-1} \in G_{gx} g g_x^{-1}$, d.h. es gibt ein $h \in G_{gx}$ mit $g_{gx}^{-1} = hgg_x^{-1}$. Damit ist $\tilde{f}(gx) = (gx, g_{gx}^{-1}(0)) = (gx, hgg_x^{-1}(0))$. Da $h \in G_{gx}$ ist, ist dies gleich $hg(x, g_x^{-1}(0)) = hg\tilde{f}(x) \in G_{gx} g \tilde{f}(x)$. $\hat{f}$, $\check{f}$ und $\tilde{f}$ erfüllen also die nötigen Voraussetzungen, um per Leiterspiel eine Transversale von X_n zu konstruieren.

Für eine praktische Umsetzung kann man obige Formalitäten vernachlässigen: Da wir aus jeder Bahn stets Repräsentanten wählen können, sodass die eigentlichen Knoten vor den uneigentlichen kommen, können wir X_i tatsächlich als Hypergraphen über i Punkte repräsentieren. Für eine Transversale des Urbilds $\hat{f}(x_0)$, $x_0 \in X_{i-1}$, reicht es aus, alle Möglichkeiten mit dem neu eingefügten markierten Punkt $i - 1$ zu durchlaufen (also einen Punkt anzuhängen). Damit muss die Markierung auch nicht explizit abgespeichert werden, zur Berechnung der Automorphismengruppe eines $(x, i - 1) \in \hat{f}^{-1}(x_0)$, $x_0 \in X_{i-1}$, wird von einer Untergruppe von $G_{x_0} \leq S_{i-1}$ ausgegangen, somit ist implizit die Markierung von

Algorithmus 4.4 *gen*$(E, B, stab_B)$

Voraussetzung:
Algorithmus zum Durchlaufen von $\hat{f}_n^{-1}(x)$, $x \in X_{n-1}$,
Berechnung von $\check{f}_n(\check{x})$, $\check{x} \in \check{X}_n$,
Berechnung von $\tilde{f}_n(x)$, samt G_x, $x \in X_n$ (modifizierter Kanonisierer: *can′*)

Eingabe:
E, Erz.-System von G (Rechtstransversalenkette)
B, Transversale von X_{n-1}
$(E_b)_{b\in B}$, Erz.-Systeme der Stabilisatoren zu $b \in B$

Ausgabe:
C, Transversale von X_n
$(E_c)_{c\in D}$, Erz.-Systeme der Stabilisatoren zu $c \in C$

```
1  proc gen(E, B, (E_b)_{b∈B})
2    begin
3      foreach b ∈ B do
4        foreach x̂ ∈ f̂^{-1}(b) do
5          (g, E_x̂) := can(E_b, x̂);
6          if g x̂ ≠ x̂ then continue fi;                  // evtl. Lerneffekt.
7          y := f̌(x̂);
8          (x̌, E_y) := can′(E, y, E_x̂);                  // Es ist x̌ = f̃(y).
9          (g, ·) = can(E_y, x̌);
10         if (g x̌ ≠ x̂) then continue fi                 // evtl. Lerneffekt.
11         C += y;  Speichere E_y in (E_c)_{c∈C};
12       od
13     od
14     return (C, (E_c)_{c∈C})
15   end.
```

Algorithmus 4.5 *gen_ori*(E, x_0, E_{x_0})

Eingabe:
E, Erz.-System von G (Rechtstransversalenkette)
$x_0 \in W^{((i-1)^k)}$, kanonischer Repräsentant,
E_{x_0}, Erz.-System des Stabilisators G_{x_0}

Ausgabe:
Teil einer Transversale von X_i, samt Stabilisatoren

```
1  proc gen(E, x_0, E_{x_0})
2    begin
3      for (durchlaufe Vervollständigungen von x_0 zu einem x ∈ W^{(i^k)}) do
4          (g, E_x̂) := can(E_{x_0}, x);                    // Es ist E_x̂ ≤ S_{i-1}.
5          if gx ≠ x then continue fi;
6          (g, E_x) := can(E, x, E_x̂);   // Abbruch, falls g_x^{-1}(0) ∉ G_x(i-1).
7          if (g_x^{-1}(0) ∉ G_x(i-1)) then continue fi;
8          Speichere (x, E_x);
9      od
10   end.
```

$i-1$ sichergestellt. Zur Berechnung von $\check{f}((x,m)) = x$ ist demnach gar nichts zu tun, und die Berechnung von $\tilde{f}(x)$ ist durch eine (teilweise) Kanonisierung von x geschehen. Dabei kann u.U. bereits während der Kanonisierung erkannt werden, ob der Test $\tilde{f}(\check{f}((x,i-1))) = (x,i-1)$ fehlschlägt. Es gilt nämlich $\tilde{f}(\check{f}((x,i-1))) = (x,i-1)$ genau dann, wenn evtl. nach Anwendung eines Automorphismus $h \in G_x$, durch $\tilde{f}$ wieder der letzte Knoten markiert wird, d.h. wenn $g_x^{-1}(0) \in G_x(i-1)$ ist. (Insbesondere entfällt bei dieser speziellen Kombination von $\check{f}$ und $\tilde{f}$ die G_x-Kanonisierung in $\check{X}$!) Eine Implementierung als Pseudocode ist in Algorithmus 4.5 angegeben. Man beachte, dass die in Zeile 4 berechnete Informationen über die Automorphismengruppe $G_{\hat{x}}$ in Zeile 6 bei der Kanonisierung von x mitverwendet werden kann.

5 Implementierung

Wir besprechen in diesem Kapitel den implementierten Generator für orientierte Matroide.

Dem Autor waren zum Zeitpunkt der Implementierung bereits zwei ausgearbeitete Generatoren orientierter Matroide bekannt. J. Bokowski und A. Guedes de Oliveira beschreiben in [BGdO00] einen Algorithmus basierend auf den in Abschnitt 1.5 beschriebenen Hypergeradenlisten. Desweiteren wird in der Dissertation von L. Finschi [Fin01] ein graphentheoretischer Generator entwickelt, der von Kokreis-Graphen zu orientierten Matroiden ausgeht.

Der letztgenannte Ansatz erlaubt die Generierung aller orientierten Matroide zu gegebenem Rang k und Grundmenge n bis auf Isomorphie, unter Verwendung der „reverse search“ Methode von Avis und Fukuda [AF96]. Demgegenüber werden bei Bokowski / Guedes de Oliveira lediglich uniforme Matroide generiert, und es wird auch keine Isomorphie berücksichtigt. Dafür bietet dieser Generator aber die komfortable Möglichkeit, Restriktionen anhand verbotener Kreise anzugeben, und somit gezielt Einfluss auf die Generierung zu nehmen. Für unsere angestrebte Anwendung in der Chemie ist die Einschränkung auf uniforme Matroide akzeptabel, und die Verwendung der Restriktionen äußerst erwünscht. Zudem liegt die geometrische Interpretation und die Möglichkeit, das Chirotop daraus direkt abzulesen, näher an unseren Anforderungen, als die bei Finschi verwendete Datenstruktur. Es bleibt jedoch das Problem, isomorphe Lösungen zu vermeiden.

Keiner der genannten Generatoren kann also ohne Erweiterung zur Generierung von Konformeren in der Chemie verwendet werden. Nach Abwägung der Vor- und Nachteile erschien der Algorithmus von Bokowski und Guedes de Oliveira als geeigneter Ausgangspunkt für einen *Generator bis auf Isomorphie unter Berücksichtigung von Nebenbedingungen*, welcher in dieser Arbeit entwickelt wird. Der Source-Code für einen Generator unitärer orientierter Matroide vom Rang $k = 4$ wurde mir freundlicherweise zur Verfügung gestellt und diente mir insbesondere für Vergleichsläufe.

Der Algorithmus in [BGdO00] generiert schrittweise, von einem orientierten Matroid über n vom Rang k ausgehend, neue orientierte Matroide über $n + 1$. Dies

geschieht durch Hinzufügen eines neuen Elements auf jede mögliche Weise. Die Effizienz dieses Generators beruht hauptsächlich auf Satz 1.5.6: Danach ist für abstrakte Hypergeradenlisten im Wesentlichen nur die alternierende Eigenschaft von χ_{hll} zu überprüfen; die Gültigkeit der dreiwertigen Grassmann-Plücker-Relationen ergibt sich dann implizit. Mittels *Constraint propagation*, dem vorausschauenden Auswerten der Einschränkungen (siehe [Nil03], S183f), werden nun aus der erforderlichen alternierenden Eigenschaft von χ_{hll} die möglichen Positionen des neuen Elements in den jeweiligen Hypergeradenanordnungen eingeschränkt. Interessant ist dabei, dass die erlaubten Positionen zu jedem Zeitpunkt ein Intervall bilden.

Für den hier beschriebenen Algorithmus wurde nicht schrittweise vorgegangen, sondern es wurde eine direkte Generierung aller orientierten Matroide über n vom Rang k durch einem Backtrack-Algorithmus gewählt. Dadurch ist eine übersichtliche Struktur des Codes erzielt worden. Die Effizienz wird durch Ausnutzung von *Lerneffekten* und von *ordnungstreuer Erzeugung* sichergestellt.

Im Laufe der Arbeit stellte sich heraus, dass sich die naive Datenstruktur des Chirotops als Abbildung von $\binom{n}{k} \to \{0, 1\}$ besser eignet als die Hypergeradenlisten. Zwar benötigt ein Chirotop theoretisch für große n mehr Speicherplatz als eine Hypergeradenliste, jedoch sind im Algorithmus nach Bokowski und Guedes de Oliveira während der Generierung weitere Daten zu verwalten, sodass trotzdem ein Speicherbedarf von $O(n^k)$ benötigt wird.

Bevor wir den Generator orientierter Matroide angehen können, besprechen wir Implementierungen einiger kombinatorischer Hilfsmittel. Diese sind nötig, um k-Tupel im Computer komfortabel und effizient zu handhaben.

5.1 Vorarbeiten zur Implementierung

Als Programmiersprache wurde C++ gewählt. Wir greifen unter anderem auf Konzepte der objektorientierten und der algorithmischen Programmierung zurück.

Um Überschreitungen von Bereichsgrenzen zu vermeiden, benutzen wir im Zusammenhang mit Binomialkoeffizienten den Datentyp long long. Diesen kürzen wir wie folgt ab:

```
typedef long long llong;
```

Binomialkoeffizienten

Wir berechnen den Binomialkoeffizienten $\binom{n}{k}$ mit dem allgemein bekanntem Algorithmus 5.1. Um „benachbarte“ Binomialkoeffizienten zu berechnen, benutzen wir jedoch folgende wesentlich effizienteren Formeln für $n \geq 0, k \geq 0$:

$$\binom{n}{k+1} = \frac{n-k}{k+1}\binom{n}{k}$$

$$\binom{n}{k-1} = \begin{cases} 1 & \text{falls } k = n+1 \\ \frac{k}{n-k+1}\binom{n}{k} & \text{sonst.} \end{cases} \qquad (\text{für } k > 0)$$

$$\binom{n-1}{k} = \frac{n-k}{n}\binom{n}{k} \qquad (\text{für } n > 0)$$

$$\binom{n+1}{k} = \begin{cases} 1 & \text{falls } k = n+1 \\ \frac{n+1}{n-k+1}\binom{n}{k} & \text{sonst.} \end{cases}$$

(Bei der Implementierung ist darauf zu achten, dass man nur ganzzahlig rechnet.)

Algorithmisch interessanter ist hingegen die folgende Umkehrfunktion des Binomialkoeffizienten: Wir führen den *binomialen Logarithmus* ein als:

$$\mathrm{binlog}_k(x) := \max\{n \in \mathbb{Z} \mid \binom{n}{k} \leq x\}\,.$$

Um dies zu berechnen, geben wir zunächst eine Abschätzung an: Für $x > k$ ist natürlich $\mathrm{binlog}_k(x) \geq k+1$. Eine weitere Abschätzung ergibt sich aus:

$$\binom{n}{k} \lesssim \frac{n^k}{k!}\,.$$

Für $n := \mathrm{binlog}_k(x)$ gilt damit $x < \binom{n+1}{k} \leq \frac{(n+1)^k}{k!}$. Lösen wir diese Ungleichung nach n auf, so erhalten wir $n > \sqrt[k]{x \cdot k!} - 1$. Darin können wir den Faktor $\sqrt[k]{k!}$ mit Hilfe der Stirling Näherungsformel umformen:

$$n! \gtrsim \sqrt{2\pi n}(\frac{n}{e})^n\,.$$

Es folgt $\sqrt[k]{k!} \gtrsim \sqrt[2k]{2\pi k} \cdot \frac{k}{e}$ und damit $n > \sqrt[k]{x} \cdot \sqrt[2k]{2\pi k} \cdot \frac{k}{e} - 1$. Unter Verwendung der Gaussklammern $\lfloor x \rfloor$, welche für $x \in \mathbb{R}$ die größte ganze Zahl kleiner oder

Algorithmus 5.1 Berechnung des Binomialkoeffizienten

```
llong choose( int k, int n);                    // Berechne (n k)
  if(k>n/2) k=n-k;
  if(k<0) return 0;
  llong ret=1;
  for(int i=0;i<k;++i) ret=ret*(n–i)/(i+1);
  return ret;
}
```

Algorithmus 5.2 Berechnung des binomialen Logarithmus

```
static const double PI=3.141592654;
static const double E =2.718281828;

void binlog( int k, llong x, int &n, llong &y); // Berechne n := binlog_k(x) und y := (n k).
  // (Behandle die F¨alle k = 0 und x ≤ k gesondert.)
  ...
  n=max( k+1, int(pow(x,1.0/k)*pow(2*PI*k, 0.5/k)*k/E) );
  for(y=choose(k,n); y<=x; y=y*n/(n-k))
    ++n;
  y=y*(n-k)/n; –n;                              // mache den letzten Schritt r¨uckg¨angig
}
```

gleich x angeben, erhalten wir die Abschätzung:

$$\text{Für } x > k \text{ gilt: } \ \mathrm{binlog}_k(x) \geq \max(k+1, \left\lfloor \sqrt[k]{x} \cdot \ \sqrt[2k]{2\pi k} \cdot \frac{k}{e} \right\rfloor).$$

Um $\mathrm{binlog}_k(x)$ zu berechnen, starten wir mit n bei der angegebenen unteren Schranke, berechnen $y := \binom{n}{k}$, und erhöhen iterativ n solange, bis für das neu berechnete $y' := \binom{n}{k} = \frac{n}{n-k} \cdot \binom{n-1}{k} = \frac{n \cdot y}{n-k}$ die Ungleichung $y' > x$ gilt. Dann ist $n - 1 = \mathrm{binlog}_k(x)$. Da wir häufig neben $n = \mathrm{binlog}_k(x)$ gleichzeitig auch den Binomialkoeffizienten $y := \binom{n}{k}$ benötigen, geben das wir in Algorithmus 5.2 mit zurück.

Die Menge $\binom{\mathbb{N}}{k}$

Wir bezeichnen die Menge aller streng monoton steigenden k-Tupel über $\mathbb{N}$ als $\binom{\mathbb{N}}{k}$. Diese ist durch die reverse lexikographische Ordnung (revlex) angeordnet:

Algorithmus 5.3 Eine rank-Funktion für $\binom{\mathbb{N}}{k}$

```
llong rank( int k, int* a);
  llong idx=0;
  for(int i=0;i<k;++i)
    idx+=choose(i+1,a[i]);
  return idx;
}
```

Algorithmus 5.4 Eine unrank-Funktion für $\binom{\mathbb{N}}{k}$

```
void unrank( int k, llong idx, int* a);
  llong y;                          // Der Speicher für a muss bereits allokiert sein.
  for(int i=k-1; i>=0; --i) {
    binlog(i+1,idx,a[i],y);
    idx-=y;
  }
}
```

Für $\vec{a}, \vec{b} \in \binom{\mathbb{N}}{k}$ sei

$$\vec{a} <_{\text{rev}} \vec{b} :\Longleftrightarrow \exists\, i \in k : a_{k-1-i} < b_{k-1-i} \text{ und } \forall j < i : a_{k-1-j} = b_{k-1-j}\,.$$
$$\vec{a} \leq_{\text{rev}} \vec{b} :\Longleftrightarrow \vec{a} <_{\text{rev}} \vec{b} \text{ oder } \vec{a} = \vec{b}\,.$$

Diese Anordnung hat gegenüber der gewöhnlichen lexikographischen Anordnung den Vorteil, dass man die k-Tupel durchnummerieren kann:

5.1.1 Satz. *Folgende Abbildung ist ein Ordnungsisomorphismus:*

$$\mathit{rank} : \left(\tbinom{\mathbb{N}}{k}, \leq_{\mathit{rev}}\right) \to (\mathbb{N}, \leq)\,, \mathit{rank}(\vec{a}) = \sum_{i \in k} \binom{a_i}{i+1}$$

Beweis: [Wil85, Theorem 1.59 auf Seite 34]. □

Die entsprechende Implementierung sowie die zugehörige inverse unrank() Funktion (vgl. [Wil85, Theorem 1.60]) sind in den Algorithmen 5.3 und 5.4 aufgelistet.

Die gewählte Anordnung bietet die Möglichkeit, mittels der Funktionen choose() und binlog() aus dem Ranking eines k-Tupel $\vec{a}$ direkt die Rankings von

verlängerten $(k+1)$-Tupeln oder von verkürzten $(k-1)$-Tupeln zu berechnen:

$$idx = \text{rank}(k, (a_0, \ldots, a_{k-1})) \text{ und } a_k > a_{k-1} \Longrightarrow$$
$$\text{Mit } delta := \text{choose}(k+1, a_k) \text{ gilt:}$$
$$idx + delta = \text{rank}(k+1, (a_0, \ldots, a_k)) .$$

$$idx = \text{rank}(k, (a_0, \ldots, a_{k-1})) \Longrightarrow$$
$$\text{Mit binlog}(k, idx, c, delta) \text{ gilt:}$$
$$c = a_{k-1} \text{ und } idx - delta = \text{rank}(k-1, (a_0, \ldots, a_{k-2})) .$$

Alternativ seien die folgenden Funktionen zum Durchlaufen von $\binom{\mathbb{N}}{k}$ gegeben:

void first tupel(**int** k, **int*** a); *// Setzt $a = (0,\ldots,k-1)$; Speicher muss bereits allokiert sein.*
void nexttupel(**int** k, **int*** a); *// Modifiziert das k-Tupel a zu seinem Nachfolger.*

Alternierende Funktionen

Es ist offensichtlich hochredundant, alle Funktionswerte einer alternierenden Funktion $\chi : n^k \to \{0, \pm 1\}$ zu speichern. Stattdessen betrachten wir Funktionen $\varphi : \binom{n}{k} \to \{0, \pm 1\}$ und deren alternierende Fortsetzung $\overline{\varphi} : n^k \to \{0, \pm 1\}$.

Der Funktionswert von $\overline{\varphi}$ für ein beliebiges k-Tupel $\vec{a} \in n^k$ wird dann bestimmt, indem wir das Tupel $\vec{a}$ sortieren. Stellen wir dabei fest, dass $\vec{a}$ nicht injektiv ist, so ist $\overline{\varphi}(\vec{a}) = 0$. Ansonsten sei π die sortierende Permutation, d.h. $\pi\vec{a}$ sei streng monoton steigend. Dann ist $\overline{\varphi}(\vec{a}) = \text{sgn}(\pi)\varphi(\pi\vec{a})$. Hierzu wurden folgende Funktionen implementiert:

int xswap(**int** &i, **int** &j); *// Vertauscht gegebenenfalls die Werte i und j, sodass $i \leq j$. Der Rückgabewert ist 0, falls $i = j$ gilt, -1, falls die Werte vertauscht wurden und 1 sonst.*
int xsort(**int** k, **int*** a); *// Sortiert ein k-Tupel von Integer-Werten. Der Rückgabewert ist 0 (Tupel ist nicht injektiv) oder ± 1 (das Signum der angewendeten Permutation)*

Für die Implementierung von xsort wurde ein naiver Bubble-Sort verwendet, da in der hiesigen Anwendung nicht mit großen k zu rechnen ist und somit auf unnötigen Overhead verzichtet wird. Wir können damit die alternierende Fortsetzung $\overline{\varphi}(\vec{a})$ für beliebige $\vec{a} \in n^k$ auswerten:

5.1.2 $$\overline{\varphi}(\vec{a}) = \text{xsort}(\vec{a}) \cdot \varphi(\vec{a}) ,$$

Algorithmus 5.5 Die Basisklasse alternating_fct

```
class alternating_fct {
protected:
  virtual int call(llong idx)=0;  // Zu überladen: Funktionsaufruf für geordnete k-Tupel
public:
  int k,n;                        // Eine Abbildung von n^k → ℤ.
  int operator()(llong idx)       { return call(idx); }
                                  // idx repräsentiert geordneten Tupel
  int operator()(int* a);         // Aufruf für a⃗ ∈ n^k; a⃗ wird sortiert wie in Gl. 5.1.2
  // Weitere Elementfunktionen, wie Ein-/Ausgabe
};
```

wobei an φ der bereits durch den Aufruf von xsort($\vec{a}$) sortierte Tupel zu übergeben ist.

In Algorithmus 5.5 ist eine abstrakte Basisklasse *alternating_fct* deklariert, welche die grundlegende Funktionalität für alternierende Funktionen von $\mathbb{Z}^k \to \mathbb{Z}$ festlegt. Wie besprochen, werden sortierte k-Tupel grundsätzlich mittels der rank() Funktion aus Satz 5.1.1 als Zahl im Datentyp llong dargestellt, und es bietet sich dadurch an, eine von alternating_fct abgeleitete Klasse alternierende Funktionen von $n^k \to \{0, \pm 1\}$ als vector<int> der Länge $\binom{n}{k}$ zu implementieren.

Eine derartige Datenstruktur ist noch allgemeiner als Chirotope, da die Grassmann-Plücker-Relationen nicht überprüft werden. Wir nennen die Implementation einer alternierenden Funktion als Vektor *prechirotope*:

```
class prechirotope:public alternating_fct, public vector<int> {
protected:
  int  call(llong idx) const  { return at(idx); }
public:
  // Konstruktoren
  void init(int __k, int _n)    { n=_n; k=__k; resize(choose(k,n); }
  void set(llong idx, int sgn) { at(idx)=sgn; }
};
```

Neben der Realisierung als vector<int> der Länge $\binom{n}{k}$ sind noch andere Implementierungen für orientierte Chirotope denkbar, beispielsweise als Hypergeradenliste in einem vector<int> der Länge $\binom{n}{k-1}$. Dann ist die Elementfunktion alternating_fct::call() gemäß Definition 1.5.4 zu implementieren. Ein nicht zu vernachlässigender Vorteil der elementaren Implementierung als Chirotop ist die Möglichkeit, während der Generierung einzelne Funktionswerte direkt zu ändern.

Algorithmus 5.6 Schema zur Generierung diskreter Strukturen

```
generator<T> gen;
gen.init(/*...*/);                    // Evtl. Initialisierung des Generators
for(gen.begin(); !gen.end(); ++gen) {
  cout << *gen;                       // Zugriff auf die aktuelle Struktur
}
```

5.2 Generatoren für diskrete Strukturen

Unter *diskreten Strukturen* verstehen wir hier schlicht die Elemente einer endlichen Menge X. Wir generieren also alle Elemente $x \in X$, wobei X von vornherein nicht explizit gegeben ist. Programmiertechnisch seien dabei alle zu generierenden Strukturen, also alle Elemente $x \in X$ in einem Datentyp *T* darstellbar.

Die Generierung aller $x \in X$ kann man dann als iterativen Durchlauf durch einen virtuellen Container X sehen. Wir können uns also beim Design an dem entsprechenden Konzept der Standard C++ Klassenbibliothek, der „Standard Template Library“ (STL) orientieren. Diese arbeitet mit *Iteratoren*, welche beim Durchlaufen durch Container stets auf eines der Elemente „zeigen“.

Entsprechend können wir *Generatoren* einführen. Der Generator „zeigt“ während eines Generierungsprozesses stets auf eine aktuelle diskrete Struktur. Es sind im weiteren drei Operationen von Bedeutung: der Zugriff auf das erste Element, ein Nachfolge-Operator und schließlich ein Test, ob das Ende der Generierung erreicht wurde.

Im Gegensatz zur STL, wo auf die Elementfunktionen begin() und end() der Containerklasse zurückgegriffen wird, liegt im Fall der Generierung natürlich kein Container aller zu generierenden Strukturen vor. Deshalb sind hier alle Operationen als Elementfunktionen der Generator-Klasse zu implementieren.

Wir definieren also eine abstrakte Template-Klasse *generator*<*T*> zum Generieren aller Strukturen vom Typ *T*. Spezielle Generierungsprobleme können mittels einer davon abgeleiteten Klasse implementiert werden. Dadurch erhalten Generierungsprozesse schließlich ein einheitliches Muster, wie in Algorithmus 5.6 schematisiert.

Zum Vergleich ist hier eine typische Implementierung des Standard-Interfaces der

STL zum Durchlaufen von Containern dargestellt:

```
vector<int> vec(10,0);                    // Ein Beispiel-Vektor der L¨ange 10
vector<int>::const_iterator it;           // der Iterator durchl¨auft den Vektor
for(it=vec.begin(); it!=vec.end(); ++it) {
  cout << *it;                            // Zugriff auf ein Element des Vektors
}
```

Als Schnittstelle für einen Generator legen wir also folgende drei Elementfunktionen fest:

```
generator& begin()        // setzt die aktuelle Struktur auf das „erste“ Element x_0 ∈ X.
generator& operator++()   // aktuelle Struktur x wird durch dessen Nachfolger ersetzt.
bool       end() const    // Test ob das Ende der Generierung erreicht wurde.
```

Des weiteren verh¨alt sich der Generator wie ein Zeiger auf die aktuelle Struktur.

Als Unterschied sei nochmals darauf hingewiesen, dass bei dem STL-Interface zwei Klassen unterschiedlichen Typs auftreten, der Container und der Iterator. In unserem Fall tritt nur eine Klasse auf, nämlich der Generator.

Dies hat auch Auswirkungen auf die Handhabung von Generatoren: Während ein Iterator in den meisten Fällen lediglich als ein Zeiger angesehen werden kann, also kaum eigenen Speicher benötigt, beansprucht ein Generator hingegen durchaus Speicher: Einerseits ist die aktuell generierte diskrete Struktur darin abgelegt. Darüber hinaus sind eventuelle Parameter für den Generierungsprozess sowie weitere Hilfsinformationen zur effizienten Generierung gespeichert. Ein Generator sollte also als umfangreiche Datenstruktur angesehen werden und im Gegensatz zu Iteratoren nicht by value an Funktionen übergeben werden (sondern by reference).

Um diesem Umstand Rechnung zu tragen, und um versehentlich ineffizient programmiertem Code vorzubeugen, wird in der Implementierung der Basisklasse generator$<T>$ (siehe Algorithmus 5.7) auf einen Copy-Constructor sowie auf Vergleichsoperatoren verzichtet.

Es sei noch erwähnt, dass das hier vorgeschlagene Interface zur Generierung eine recht übersichtliche Verschachtelung mehrerer hintereinanderliegender Generierungsprozesse (gemäß dem Homomorphieprinzip) erlaubt, und das auch ohne Speicherung der generierten Zwischenergebnisse in einem Container. Ein Generierungsprozess ist lediglich eine for-Schleife, alle Details sind in der Klassenimplementation versteckt.

Bevor wir die tatsächliche Implementierung der Basisklasse generator$<T>$ vorstellen, besprechen wir eine spezielle Variante von Generierungsprozessen, das

Backtracking:

Lexikographische Generierung (Backtracking)

In vielen Fällen werden diskrete Strukturen entlang einer *lexikographischen Ordnung* generiert, in einem so genannten *Backtrack-Algorithmus*.

Dies ist insbesondere dann gebräuchlich, wenn (wie auch in unserem Fall) X von der Form $X \subseteq Y^A$ ist, wobei $Y = \{y_0, \ldots, y_{n-1}\}$ und $A = \{a_0, \ldots, a_{k-1}\}$ endliche, angeordnete Mengen der Mächtigkeiten n bzw. k sind, die diskreten Strukturen also als k-Tupel mit Werten aus Y darstellbar sind.

Bezüglich der lexikographischen Anordnung von X verwenden wir folgende Schreibweisen:

$$\begin{aligned}
x_1 \sim_i x_2 &:\Longleftrightarrow \forall j < i : x_1(a_j) = x_2(a_j) \\
x_1 <_i x_2 &:\Longleftrightarrow x_1 \sim_i x_2 \text{ und } x_1(a_i) < x_2(a_i) \\
x_1 \leq_i x_2 &:\Longleftrightarrow x_1 = x_2 \text{ oder } x_1 <_{i'} x_2 \text{ mit } i' \geq i \\
x_1 \leq\ x_2 &:\Longleftrightarrow x_1 \leq_0 x_2
\end{aligned}$$

Dabei ist durch jedes $\sim_i$, $i \in k$ eine Äquivalenzrelation auf X definiert. Wir haben also eine Kette von immer feiner werdenden Äquivalenzrelationen. Bezeichnet $\Delta X^2 := \{(x, x) \mid x \in X\}$ die Diagonale in X^2, so gilt:

$$X^2 = \sim_0 \supseteq \sim_1 \supseteq \cdots \supseteq \sim_{k-1} \supseteq \sim_k = \Delta X^2$$

Wir benutzen weiter:

5.2.1 Definition. Seien $x, x' \in X \subseteq Y^A$. Dann heißt

$$\text{diff}(x, x') := \max\{i \mid x \sim_i x'\}$$

der *unterscheidende Index* von x und x'.

Die Äquivalenzrelationen $\sim_i$ sind in folgendem Sinne mit der Anordnung verträglich:

5.2.2 Ist $x_1 \sim_i x_1' \not\sim_i x_2 \sim_i x_2'$, so gilt: $x_1 \leq x_2 \Leftrightarrow x_1' \leq x_2'$.

Es sind also die Äquivalenzklassen $[x]_i := [x]_{\sim_i}$ selbst wiederum lexikographisch angeordnet.

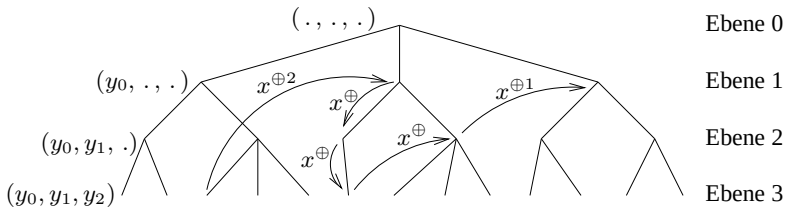


Abbildung 5.1: Die lexikographische Ordnung;
Ausschnitt eines Backtrack-Durchlaufs mit Sprüngen

Die lexikographische Anordnung der Menge X kann man auch als Baum interpretieren: Die Knoten des Baumes sind genau die Äquivalenzklassen der $\sim_i$, welche auch durch „unvollständige“ k-Tupel, also durch die i-Tupel $Y^{\{a_0,\ldots,a_{i-1}\}}$ charakterisiert werden können:

$$[X] := \{[x]_i \mid i \leq k\} \cong \bigcup_{i \in k} Y^i \,.$$

Die Baumstruktur ist durch die Enthaltensein-Relation der Klassen $[x]_i$, $x \in X$, $i \in k$ definiert. Dabei entsprechen die minimalen Elemente (die Endknoten) genau den Elementen aus X: $[x]_k \stackrel{!}{=} x$. Weiter entspricht die Wurzel dem leeren Tupel $() \in Y^0$. Wenn die *Tiefe* eines Baumes wie üblich die Anzahl der Ebenen ohne die Wurzel bezeichnet, so hat der Baum genau Tiefe k.

Zwei Elemente $x_1, x_2 \in X$ haben also einen gemeinsamen Vaterknoten auf Ebene i genau dann, wenn $x_1 \sim_i x_2$, also wenn die ersten i Komponenten von x_1 und x_2 übereinstimmen. Der unterscheidende Index $\mathrm{diff}(x_1, x_2)$ gibt dabei die unterste Ebene an, auf der x_1 und x_2 noch einen gemeinsamen Vaterknoten besitzen.

Man kann den Baum so zeichnen, dass die Blätter $x \in X$ gemäß der lexikographischen Ordnung von links nach rechts angeordnet sind. Aufgrund der oben beschriebenen Verträglichkeit der Äquivalenzrelationen mit der Ordnung (Gleichung 5.2.2) kann man dabei Kantenüberschneidungen im Baum vermeiden (vgl. Abb. 5.1; die Pfeile werden später erläutert.).

Die Interpretation als Baum liefert die Grundidee des *Backtrack-Algorithmus*: Wir durchlaufen die Knoten des Baums in Tiefensuche. Jeder erreichte Endknoten ist dann eine diskrete Struktur. Zur Formalisierung bezeichnen wir den Nachfolger eines Knotens $[x]_i$ gemäß Tiefensuche mit $[x]_i^{\oplus}$. Dadurch, dass wir die Zwischenknoten mit durchlaufen, haben wir die Möglichkeit, Teilbäume komplett auszu-

lassen, und somit ganze Bereiche des Suchraumes zu überspringen. Wir führen deshalb einen weiteren Operator ein: Ist $[x]_i$ ein Knoten im Baum, so bezeichne

$$[x]_i^{\oplus j} \;=\; [x']_{i'}$$

den nächsten Knoten im Backtrack-Durchlauf, der auf einer Ebene $\leq j$ liegt. Für die zwei (beliebig gewählten) Repräsentanten x und x' gilt diff$(x, x') < j$. Mit einem Sprung von $[x]_i$ nach $[x]_i^{\oplus j}$ überspringt man also alle noch nicht generierten diskreten Strukturen aus der Äquivalenzklasse $[x]_j$. In Abbildung 5.1 sind als Beispiel einige Nachfolge-Operatoren eingezeichnet.

Die Klasse generator<*T*>

Wir nehmen diese Funktionalität in das Interface der Klasse generator<T> auf, und erhalten folgende öffentliche Schnittstelle:

generator& begin()	*// springe zum ersten Knoten unterhalb der Wurzel* $[x]_0^{\oplus}$.
generator& operator++()	*// Gehe vom aktuellen Knoten* $[x]_i$ *zum Nachfolger* $[x]_i^{\oplus}$.
generator& next(int *j*)	*// Springe von* $[x]_i$ *zu dessen Nachfolger* $[x]_i^{\oplus j}$.
bool end() const	*// Test, ob das Ende des Durchlaufs erreicht wurde.*
int level() const	*// Gibt die Ebene* i *von* $[x]_i$ *zurück.*
bool complete() const	*// Gibt an, ob Struktur vollständig spezifiziert ist, d.h. ob* i=k *ist.*

Des weiteren verhält sich der Generator wie ein Zeiger auf die aktuelle Struktur.

Man kann jeden Generierungsprozess auch formal als Backtrack-Baum der Tiefe 1 sehen. Die Wurzel hat dann jede der Strukturen $x \in X$ als direkten Sohn.

Zur Implementierung eines Generators für diskrete Strukturen eines speziellen Typs *T* muss dann eine von generator<T> abgeleitete Klasse implementiert werden. Hierfür ist folgende Schnittstelle vorgesehen:

T **sol*;	*// Die aktuelle Struktur (eine „Lösung“)*
virtual int depth() const	*// Tiefe des Backtrack-Baums. (Default:1, kein Backtracking)*
virtual bool gen_first(int *i*)	*// Berechne ersten Sohn (wenn möglich).*
virtual bool gen_next(int *i*)	*// Berechne den nächsten Bruder (wenn möglich).*

Während der Initialisierung muss dafür gesorgt werden, dass für die Variable *sol* Speicherplatz reserviert und gegebenenfalls initialisiert wird. Der Speicher wird im Destruktor von generator<T> automatisch wieder freigegeben.

Mit der Funktion depth() wird die Tiefe des Backtrack-Baums spezifiziert, dadurch können Endknoten (auf Ebene depth()) von Zwischenknoten unterschieden werden. Weiter müssen die Elementfunktion gen_first(int) und gen_next(int) implementiert werden, um einen Depth-First-Durchlauf durch den Baum zu ermöglichen. Die Rückgabewerte dieser beiden Funktionen geben an, ob die entsprechende Aktion möglich war, d.h. ob es einen Sohn bzw. einen weiteren Bruder gab, zu dem gewechselt wurde.

Die vollständige Implementierung der abstrakten Template-Klasse generator<T> ist in Algorithmus 5.7 zu sehen. Ein Generator aller Prechirotope zum uniformen Matroid vom Rang k über n ist schließlich in Algorithmus 5.8 angegeben. Eine Generierung (noch ohne spezielle Tests, und ohne Sprünge) könnte wie folgt geschehen:

```
gen_prechirotope gen(k,n);
for(gen.begin(); !gen.end(); ++gen){
  if(gen.complete()) {
    cout << *gen;                    // neue alternierende Funktion gefunden
  }
}
```

Neben der Konstruktion weiterer diskreter Strukturen wie Partitionen, Permutationen, Summenformeln, Molekülen, usw. wird diese Basisklasse im MOLGEN5 Source-Code auch zur Tiefen- bzw. Breitensuche in Graphen, oder dem Durchlaufen aller Elemente einer Gruppe (gegeben als Transversalenkette) verwendet.

5.3 Lerneffekte

Es sei im Folgenden

$$X \subseteq \bar{X} \subseteq Y^A\,,$$

wobei $Y = \{y_0, \ldots, y_{n-1}\}$ und $A = \{a_0, \ldots, a_{k-1}\}$ wie oben endliche, angeordnete Mengen der Mächtigkeiten n bzw k sind. Die Menge $\bar{X}$ sei bereits lexikographisch generierbar mittels der Operatoren $x^{\oplus}$ bzw. $x^{\oplus i}$, wobei die Berechnung dieser Nachfolger effizient durchführbar sei.

Wir sind nun an einer Auswahl diskreter Strukturen $x \in X$ interessiert, wobei die Zugehörigkeit zu X für eine Struktur $x \in \bar{X}$ mittels einer Reihe von notwendigen Test-Funktionen entschieden wird. Verläuft einer der Tests negativ, so ist $x \notin X$.

Algorithmus 5.7 Die Klasse generator<*T*>

```
template<class T> class generator {
  private:
    int l;                          // speichert aktuellen Level für Backtrack-Generierung
  protected:
    T *sol;                         // die aktuelle Struktur
    virtual int depth() const       { return 1; } // Default: Kein Backtrack
    virtual int gen_first(int l)    =0;
    virtual int gen_next(int l)     =0;
  public:
            generator()             { l=0; sol=NULL; }
    virtual ~generator()            { if(sol) delete sol; }
    generator& begin()              { return next(0); }
    generator& operator++()         { return next(depth()); }
    generator& next(int lvl)        { bool b;
                                      if(lvl==0) { lvl=1; l=0; }
                                      if(lvl>depth()) lvl=depth();
                                      if    (lvl>l)   b=gen_first(++l);
                                      else if(lvl>0) b=gen_next(l=lvl);
                                      while(!b && ---l>0) b=gen_next(l);
                                      return *this;   }
    bool end() const                { return l==0; }
    int  level() const              { return l; }
    bool complete() const           { return l==depth(); }
    T& operator*()                  { return *sol; }
    T* operator->()                 { return  sol; }
    T const& operator*() const      { return *sol; }
    T const* operator->() const     { return  sol; }
};
```

Algorithmus 5.8 Die Klasse gen_prechirotope

```
class gen_prechirotope :public generator<prechirotope> {
protected:
  int depth() const                 { return sol->size(); }
  bool gen_first(int lvl)           { sol->set(lvl-1,+1); return true; }
  bool gen_next(int lvl)            { if(sol->chi(lvl-1)!=+1) return false;
                                      sol->set(lvl-1,-1); return true; }
public:
  gen_prechirotope()                { init(0,0); }
  gen_prechirotope(int k, int n)    { init(k,n); }
  void init(int k, int n)           { if(sol) delete sol; sol=new prechirotope(k,n); }
};
```

5.3.1 Beispiel.

1. Wir können bereits die Menge $\bar{X}$ aller alternierenden Funktionen $\varphi : n^k \to \{\pm 1\}$ lexikographisch generieren, d.h. hier ist $\bar{X} = Y^A$ mit $Y := \{\pm 1\}$ und $A = \binom{n}{k}$. Eigentlich sind wir aber an der Teilmenge X aller *Chirotope* interessiert, d.h. der alternierenden Funktionen, die zusätzlich die dreiwertigen Grassmann-Plücker-Relationen erfüllen.
2. Operiert eine Gruppe G auf A und ist $\bar{X} \subseteq Y^A$ abgeschlossen bzgl. der daraus induzierten Gruppenoperation auf Y^A, so ist auch die Generierung einer Transversale $X \subseteq \bar{X}$ (d.h. die Generierung von kanonischen Repräsentanten aller $x \in \bar{X}$) von äußerstem Interesse. Dabei entscheidet man mittels *Kanonizitätstest*, ob ein x in der Transversalen X liegt.
3. Die Kombination beider Aspekte ergibt die Generierung einer Transversalen aller Chirotope. Dies ist unser Ziel.

Die Kunst der Generierung liegt nun darin, die notwendigen Tests derart zu gestalten, dass man möglichst frühzeitig, d.h. an möglichst hohen Ebenen im Baum erkennen kann, welche Teilbäume abzuschneiden sind. Scheitert ein Test auf Ebene *lvl*, so kann die gesamte Äquivalenzklasse $[x]_{lvl}$ übersprungen werden.

Test der dreiwertigen Grassmann-Plücker-Relationen

Um aus der Menge aller alternierenden Funktionen letztendlich nur Chirotope zu generieren, überprüfen wir an jedem Knoten auf Ebene $lvl = \text{index}(\vec{a}) + 1$ mit $\vec{a} \in \binom{n}{k}$, ob alle für diese Ebene relevanten dreiwertigen Grassmann-Plücker-Relationen erfüllt sind. Eine (GP3)-Relation ist auf Ebene lvl relevant, wenn der maximale in ihr vorkommende k-Tupel $\vec{a}$ (gemäß der revers lexikographischen Ordnung auf n^k) genau Rang $\text{rank}(\vec{a}) = lvl$ hat. Wir überprüfen also die (GP3)-Relationen zu $(b_1, b_2, c_1, c_2) \in n^4$ und $\vec{x} \in n^{k-2}$ mit $\{b_1, b_2, c_1, c_2, \vec{x}\} = \{i, j, \vec{a}\}$ und $i < j < a_0$.

Erreicht man einen Endknoten, d.h. waren die Tests für alle darüberliegenden Knoten inklusive des Endknotens erfolgreich, so sind alle GP3-Relationen für die aktuelle alternierende Abbildung überprüft, und die Abbildung ist ein Chirotop.

Es sei

bool test_one_GP3(alternating_fct **const**& *chi*, **int const*** *t*, **int**[4] *idx*)

der Test auf eine GP3-Relation, wobei die $a[idx[0]]$, …, $a[idx[3]]$ die Werte b_1, b_2, c_1 und c_2, und die restlichen Werte aus dem $k+2$-Tupel $\vec{t}$ das $\vec{x}$ bezeichnen;

Algorithmus 5.9 Test aller auf Ebene *lvl*=*idx* relevanten GP3-Relationen

```
bool test_GP3(alternating_fct const& chi, int const* a) {   // idx = rank(ā), ā ∈ (n k)
  vector<int> vt(k+2);                  // Dies vereinfacht die Speicherverwaltung
  int *t=&vt[0];                        // Das k + 2-Tupel aller vorkommenden Elemente
  int b[4];
  memcpy(a,t+2,k*sizeof(int);           // ā ist im Bereich t_2, ..., t_{k+1}
  // Durchlaufe alle k + 2-Fortsetzungen (t_0, ..., t_{k+1}):
  for(fi rsttupel(2,t); t[1]<t[2]; nexttupel(2,t)) {
    // Durchlaufe jede 4-Auswahl aus t̄:
    for(fi rsttupel(4,b); b[3]<k+2; nexttupel(4,b)) {
      if(!test_GP3(chi,t,b) return false;
    }
  }
}
```

vgl Satz 1.4.6 on page 24 sowie die darauffolgende Bemerkung. Dann überprüft Algorithmus 5.9 alle auf einer Ebene *lvl* relevanten (GP3)-Relationen.

Wir erhalten also mit folgendem Code alle Chirotope vom Rang k über n:

```
gen_prechirotope gen(k,n);
int lvl;
vector<int> va(k);  int *a=&va[0];
for(gen.begin(); !gen.end(); gen.next(lvl)){
  lvl=gen.level();
  unrank(k,lvl,a);
  if(!test_GP3(*gen,a)) continue;
  if(!gen.complete()) { ++lvl; continue; }
  cout << *gen;                         // neues Chirotop gefunden
}
```

Es können in diesen Code weitere Tests eingebaut werden.

Durchlauf der Kreise eines Chirotops

Wie in Kapitel 1 besprochen, sind wir bei affinen Punktkonfigurationen lediglich an den azyklischen Chirotopen interessiert. Dabei erinnern wir uns, dass ein Chirotop azyklisch ist, falls $\mathcal{C}_\chi$ keine rein positiven Kreise enthält. Weiter wurde bereits angedeutet, dass speziell in der Chemie eine weitere Liste von Kreisen ausgeschlossen werden soll, und evtl. auch eine Liste vorgeschriebener Kreise existiert. Dazu ist es sinnvoll, alle auf einer Ebene *lvl* relevanten Kreise zu durchlaufen, sodass die Tests wieder zum frühestmöglichen Zeitpunkt ausgeführt werden können.

Gemäß Satz 1.2.14 on page 9 lässt sich jeder Kreis in der Form $C_{\vec{t}}$ aus einem $k+1$-Tupel berechnen. Wir erhalten also alle auf Ebene *lvl* relevanten Kreise, indem wir $C_{\vec{t}}$ zu allen $k+1$-Tupeln $\vec{t} = (i, \vec{a})$ mit $i < a_0$ berechnen, wobei $\vec{a}$ das k-Tupel vom Rang *lvl* sei.

Erweiterte Lerneffekte

Unter Umständen kann man bei einem fehlgeschlagenen Test sogar auf eine höhere Ebene zurückspringen. Wir beschreiben das Konzept der Lerneffekte nach R. Grund ([Gru95, GKL96]; vgl. auch *backjumping*, [Gin93]). Danach besagt ein *Lerneffekt* l eines gescheiterten Tests an Knoten $K = [x]_i$ (mit $l \leq i$), dass nicht nur alle $x \in [x]_i$ zu verwerfen sind, sondern dass auch die $x \in [x]_l$ nicht in X enthalten sind. Wir können in so einem Fall also direkt zu $x^{\oplus l}$ springen. Wir führen formal ein:

5.3.2 Definition. Sei $x \in \bar{X} \setminus X$ und $l \in \{0, \ldots, n-1\}$. Dann ist l *Lerneffekt* von x bzw. x *erfüllt den Lerneffekt* l genau dann, wenn

$$\forall x' \in \bar{X}, x \leq_{l+1} x' : x' \notin X\,.$$

Weiter erfülle jedes $x \in X$ per Definition den Lerneffekt $l = \infty$.
Insgesamt setzen wir:

$$\text{learn}(x) := \begin{cases} \infty & \text{falls } x \in X \\ \min\{l \mid x \text{ erfüllt den Lerneffekt } l\} & \text{sonst.} \end{cases}$$

Wir bezeichnen $\text{learn}(x)$ als den *optimalen Lerneffekt* von x.

Jedes $x \in \bar{X} \setminus X$ erfüllt trivialerweise den Lerneffekt $n-1$, es gilt also

5.3.3 Bemerkung. Sei $x \in \bar{X}$. Dann gilt:

$$x \notin X \iff \text{learn}(x) < \infty$$

Der optimale Lerneffekt $\text{learn}(x)$ lässt sich während der Generierung im Allgemeinen nicht bestimmen. Wir suchen aber nach einem möglichst guten (d.h. möglichst kleinen) *erkannten* Lerneffekt l. Je kleiner l ist, desto größer ist der Bereich an Kandidaten, die im lexikographischen Durchlauf übersprungen werden.

Letztendlich verbleibt es bei der Implementierung der einzelnen Tests, welche Lerneffekte erkannt werden. Ein klassisches Beispiel für einen Test mit erweitertem Lerneffekt ist der implementierte Kanonizitätstest nach ordnungstreuer Erzeugung, siehe Seite 92.

5.4 Das Programm `origen`

Das Programm `origen` („orientation generator“) ermöglicht die Generierung von Orientierungen zum uniformen Matroid. Per Kommandozeilenparameter können eine Reihe von Restriktionen an die Generierung gestellt werden, eine ausführliche Hilfeseite erhält man mit `origen -h`. Wir beschreiben hier auszugsweise die Benutzung.

Chirotope werden nach folgender Syntax ein- und ausgegeben: Das Chirotop χ wird durch den Rang k, gefolgt von den Funktionswerten $\chi(\vec{a}) \in \{+,-,0\}$, $\vec{a} \in \binom{n}{k}$, dargestellt, wobei die $\vec{a} \in \binom{n}{k}$ in reverser lexikographischer Ordnung durchlaufen werden. Die Anzahl n der Punkte kann man dann aus der Länge des Chirotops mittels des binären Logarithmus berechnen. So ist mit

```
4++----+-+++++-+
```

ein Chirotop χ vom Rang $k = 4$ angegeben. Wegen $\binom{6}{4} = 15$ ist dieses Chirotop über $n = 6$ Punkten definiert, und die Funktionswerte sind explizit:

$$\begin{array}{lll}
\chi(0,1,2,3) = + & \chi(0,1,2,4) = + & \chi(0,1,3,4) = - \\
\chi(0,2,3,4) = - & \chi(1,2,3,4) = - & \chi(0,1,2,5) = - \\
\chi(0,1,3,5) = + & \chi(0,2,3,5) = - & \chi(1,2,3,5) = + \\
\chi(0,1,4,5) = + & \chi(0,2,4,5) = + & \chi(1,2,4,5) = + \\
\chi(0,3,4,5) = + & \chi(1,3,4,5) = - & \chi(2,3,4,5) = +
\end{array}$$

Die Sortierung der k-Tupel in reverser lexikographischer Ordnung folgt dem Züricher Format für Chirotope. Optional können die $\vec{a} \in \binom{n}{k}$ auch in standard lexikographischer Ordnung sortiert werden. Konvertierungen der beiden Formate sind möglich. Das gleiche Chirotop in standard lexikographischer Reihenfolge lautet:

```
4++--++--++-++-+.
```

Ein Generierungsprozess zur Erzeugung aller Chirotope vom Rang k über n wird mit der Syntax

`origen` *k n*

gestartet. Die Wahl der Isomorphieklassen, d.h. ob bzgl. Umnummerierungen, Negation, Reorientierung oder Kombinationen davon kanonisiert werden soll, wird mit den kombinierbaren Optionen `-canneg`, `-canlab` und `-canori` eingestellt. Für Umnummerierungen kann zusätzlich per Option `-G` die operierende

Gruppe $G \leq S_n$ angegeben werden. Diese wird dann als Menge von Generatoren in Zykelschreibweise übergeben, z.B. `-G "<(0,1,2),(1,2,3)>"` (die Elemente beginnen mit 0), oder auch als vordefinierte Gruppe, etwa `-G "<<A4>>"`.

Als Restriktion kann eine Gruppe von Automorphismen mit der Option `-A` vorgeschrieben werden. Die Liste verbotener Kreise wird per Option `-circs` spezifiziert, und zwar in der Form `-circs "(0,-2,3,4),(1,4,-5,6)"`. Speziell zur Einschränkung auf azyklische Chirotope ist die Option `-acyclic` vorhanden.

Die eingegebenen Parameter werden nicht auf Konsistenz geprüft. So sollte sich die Liste der verbotenen Zykel aus Bahnen unter der operierenden Gruppe zusammensetzen. Ansonsten ist das Ergebnis schwer interpretierbar, da evtl. ein kanonischer Repräsentant die Restriktion verletzt, während nicht kanonische Versionen zwar kompatibel zu den Restriktionen sein könnten, aber aufgrund des Kanonizitätstests verworfen werden. Im Fall der Generierung von Chirotopen zu chemischen Strukturformeln ist diese Einschränkungen jedoch nicht schwerwiegend, da die verbotenen Kreise aus dem molekularen Graphen abgelesen werden, während die operierende Gruppe die Graph-Automorphismengruppe ist. Im Graphen isomorphe $k+1$-Tupel von Knoten führen also zu isomorphen Bedingungen für die Kreise, und somit ist die entstehende Liste verbotener Kreise stets eine Vereinigung von Bahnen.

Weiter ist die Angabe einer Gruppe von Automorphismen A nur dann mit der Generierung von Isomorphieklassen unter Umnummerierung voll kompatibel, wenn A Normalteiler in der operierenden Gruppe G ist. Ansonsten können $x \in X$ mit $Ax = x$ existieren, deren kanonischer Repräsentant gx jedoch nicht fix unter A bleibt. Diese Isomorphieklassen werden dann nicht generiert. Als Ausweg kann man entweder nur Isomorphieklassen unter dem Normalisator $N^G(A)$ generieren, oder man startet für jedes gAg^{-1}, $g \in G$ eine separate Generierung, und erhält so aus jeder Isomorphieklasse mindestens einen Repräsentanten, unter Inkaufnahme von evtl. doppelt generierten Isomorphieklassen. Während die Kombination einer Gruppe von Automorphismen mit der Kanonisierung bzgl. Negation keinerlei Probleme bereitet, ist die Kombination mit Kanonisierung bzgl. Umorientierung nicht angedacht worden.

Vergleich

Unser Generator ist nur bedingt mit den Generatoren aus Darmstadt und Zürich zu vergleichen, da unterschiedliche Zielsetzungen verfolgt wurden. Die Tabellen 7.2 und 6.4 aus [Fin01] betreffen uniforme Matroide und konnten so mit unserem

Generator verifiziert werden. Es stellte sich auch heraus, dass wir hier an vergleichbare Grenzen stoßen wie die Züricher. Es macht aufgrund der enormen Anzahl an Chirotopen auch nur wenig Sinn, vollständige Tabellen von orientierten Matroiden für größere Parametersätze zu generieren. Hier sind gezielte Generierungen mit spezifischen Restriktionen gefragt.

6 Zusammenfassung und Ausblick

Es wurde ein Generator von Isomorphieklassen orientierter uniformer Matroide (genauer: von Isomorphieklassen uniformer Chirotope) vom Rang k über n, k,$n \in \mathbb{N}$, entwickelt und implementiert. Dabei sind als Restriktionen eine Liste verbotener Kreise sowie eine vorgegebene Gruppe von Automorphismen möglich. Insbesondere kann die Konstruktion auf azyklische Chirotope eingeschränkt werden. Die Isomorphieklassen können bzgl. Umnummerierung, Negation oder Reorientierung sowie bzgl. Kombinationen daraus gewählt sein. Für Umnummerierungen kann kann man sich auf eine Untergruppe $G \leq S_n$ beschränken, um eine zusätzliche Struktur auf n, wie einen molekularen Graphen, zu berücksichtigen. Die erwähnte Einschränkung auf uniforme Matroide ist nicht von prinzipieller Natur: Mit nur wenig Aufwand ist es möglich, alle Orientierungen zu einem beliebigen anderen vorgegebenen Matroid zu konstruieren.

Die Konstruktion der Chirotope an sich ist so schnell, dass fast der gesamte Aufwand zur Kanonisierung der gefundenen Kandidaten benötigt wird. Die Anzahl der zu kanonisierenden Kandidaten wird zwar mittels des Prinzips der ordnungstreuen Erzeugung relativ gering gehalten, mit dem in Kapitel 4 entwickelten Ansatz gemäß ordnungstreuer Erzeugung sind aber in naher Zukunft noch starke Effizienzsteigerungen zu erwarten.

Neben der Entwicklung eines Computerprogramms wurde Klarheit in die Zusammenhänge zwischen dem praktisch sehr effizienten Algorithmus der Kanonisierung von Graphen mittels iterierter Verfeinerung (`nauty`) und dem Homomorphieprinzip gebracht. Daraus ergibt sich die Möglichkeit, die Idee der iterierten Verfeinerung auf andere Klassen diskreter Strukturen anzuwenden. Ein entsprechender Algorithmus wurde in Pseudocode verfasst.

Ebenso wurde McKay's Ansatz zur Konstruktion diskreter Strukturen mit dem Homomorphieprinzip und dem Leiterspiel nach B. Schmalz in Verbindung gebracht, bzw. die bereits vermuteten Zusammenhänge wurden durchleuchtet. Auch

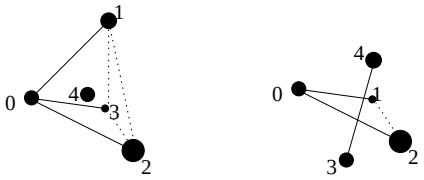


Abbildung 6.1: Zwei Situationen im molekularen Graphen, die verbotene Kreise $(0\,1\,2\,3\,\bar{4})$ bzw. $(0\,1\,2\,\bar{3}\,\bar{4})$ induzieren

hier wurde ein Algorithmus in Pseudocode angegeben, der sich direkt für Chirotope umsetzen lässt.

Anwendung in der Chemie

Der Generator wird in Zukunft vor allem Verwendung finden, um einen Überblick über mögliche Konformationen von Molekülen zu geben. Er ist somit ein erster Schritt in Richtung auf einen Generator von Stereoisomeren bzw. von Konformeren, als Ergänzung zum Molekül-Konfigurationsgenerator MOLGEN.

Zu gegebenen molekularen Graphen ist insbesondere die Konstruktion von Isomorphieklassen unter Umnummerierung bzgl. der Graph-Automorphismengruppe gefragt. Das bedeutet, es werden nicht alle Umnummerierungen von Chirotopen als äquivalent angesehen, sondern nur Automorphismen des molekularen Graphen. Ist man auch an chiralen Molekülen interessiert, so sollte man *nicht* bzgl. Negation kanonisieren, und ebenso sind Umorientierungen für affine Punktkonfigurationen schwer interpretierbar.

Zusätzlich kann man aus dem molekularen Graphen mittels chemischer Gesetzmäßigkeiten Restriktionen für die Generierung in Form von verbotenen Kreisen ablesen: Als Beispiel sind in Abbildung 6.1 zwei Situationen skizziert, die bei gewöhnlichen Umgebungsbedingungen als chemische Konformation ausgeschlossen werden können. Die linke Abbildung beschreibt ein Atom samt drei Nachbarn, wobei sich in dem davon aufgespannten Tetraeder ein weiteres Atom befindet. Die rechte Abbildung bezeichnet eine Situation, wo durch das von einem Atom und zwei Nachbarn aufgespannte Dreieck eine weitere Bindung geht. Beide Situationen können bei der Generierung durch Angabe auszuschließender Kreise vermieden werden. Weiter kann eine Gruppe von geometrischen Automorphismen vorgeschrieben werden, wie sie beispielsweise aus NMR-Daten ablesbar ist.

Zu jeder gefundenen Punktkonfiguration kann schließlich mittels Energieminimierung unter Nebenbedingungen eine Konformation gesucht werden. (Zuvor können aus dem molekularen Graphen nach [CH88] Schranken für die Distanzmatrix abgelesen werden. Es wird dann lediglich nach einer Realisierung der Chirotope innerhalb dieser Schranken gesucht. Aus den Schranken für die Distanzmatrix kann man auch direkt Kriterien für die Konstruktion der Chirotope ablesen, welche in einer zukünftigen Version des Generators mit berücksichtigt werden.) Insgesamt erhält man einen Satz an Konformationen, welcher einen guten Überblick über den Konformationsraum des Moleküls gibt.

Für diesen hier kurz skizzierten Weg zu einem Isomer- und Konformer-Generator wurden von mir bereits einzelne Teile implementiert. Ein kleines Hilfsprogramm berechnet so aus einem gegebenen molekularen Graphen die Liste der erwähnten verbotenen Kreise. Eine Berechnung von Schranken für die Distanzmatrix samt Verschärfung mittels Dreiecksungleichungen nach [CH88] wurde implementiert. Das Programmpaket SQP V1.1 zur nichtlinearen restringierten Optimierung von M. Gerdts [Ger04] wurde mit der Energiefunktion aus MOLGEN gekoppelt, sodass Molekülplatzierungen unter Nebenbedingungen möglich sind. Als Nebenbedingungen gehen neben den Schranken der Distanzmatrix auch die Orientierungen eines Chirotops ein.

Exemplarisch wurden so Konformationen zu Cyclohexan bestimmt, indem alle Isomorphieklassen von azyklischen uniformen Chirotopen vom Rang 4 über 6 Punkten unter Umnummerierungen bzgl. der Diedergruppe D_6 erzeugt wurden. Dabei wurden Kreise von den in Abbildung 6.1 gezeigten Typen ausgeschlossen. Von den gefundenen Chirotopen wurden nur die Orientierungen derjenigen Quadrupel berücksichtigt, die im molekularen Graphen zusammenhängende Teilgraphen bilden und so ergaben sich 13 theoretische Konfigurationen für Cyclohexan. Es stellte sich weiter heraus, dass mittels Energieoptimierung lediglich zu vier dieser Konfigurationen ein zulässiges energetisches Gleichgewicht gefunden werden konnte, und die so gefundenen Konformationen entsprechen genau den bekannten Twistformen (Bild und Spiegelbild), der Sesselform und der Wannenform des Cyclohexan.

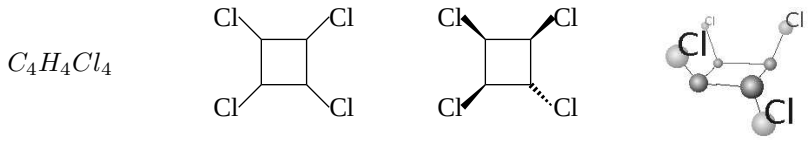
Zur Visualisierung wurde ein Molekülviewer und Konformationseditor entwickelt, der u.a. zur aktuellen Platzierung das zugehörige Chirotop darstellt.

Existenziell für den weiteren Erfolg des Projekts ist jedoch die Verfügbarkeit eines effizienten Generators für Chirotope unter Nebenbedingungen, wie er in dieser Arbeit beschrieben und zum Teil implementiert wurde.

Abschließend sei angemerkt, dass die Diskretisierung auf affine Punktkonfigura-

tionen und deren Darstellung durch ein Chirotop genau der *sterischen Beschreibung* der Konfiguration von Molekülen entspricht, wie sie unabhängig voneinander sowohl von der Gruppe um A. Dreiding in Zürich ([DW80, DDH82]) als auch im Umfeld von N. Zefirof und S. Tratch in Moskau ([TZ87, KTZ90, TZ96]) vorgeschlagen wurde. Konzeptionell wird dabei das klassische 3-Ebenen-Modell der molekularen Beschreibung um eine weitere sterische Ebene zur Beschreibung der Konfiguration durch das Chirotop erweitert, und wir erhalten ein *4-Ebenen-Modell*:

1. *Komposition* 2. *Konstitution* 3. *Konfiguration* 4. *Konformation*



In diesem Sinne kann man den hier vorgestellten Generator `origen` auch als Konfigurations-Generator für Moleküle verstehen. Die Chirotope erlauben eine genauere Unterscheidung von Konfigurationen als die klassische Stereochemie mit Stereozentren. Somit geht der hier vorgestellte Ansatz auch über die klassischen Stereogeneratoren, etwa von J. G. Nourse et al. [Nou79, NCSD79, NSCD80], von L. A. Zlatina [Zla91] oder von T. Wieland [Wie94, WKL96] hinaus.

Literaturverzeichnis

[AF96] D. Avis and K. Fukuda, *Reverse search for enumeration*, Discrete Appl. Math. **6** (1996), 21–46.

[BGdO00] J. Bokowski and A. Guedes de Oliveira, *On the generation of oriented matroids*, Discrete Comput. Geom. **24** (2000), 197–208.

[BGH+95] C. Benecke, R. Grund, R. Hohberger, R. Laue, A. Kerber, and T. Wieland, *Molgen+, a generator of connectivity isomers and stereoisomers for molecular structure elucidation*, Anal. Chim. Acta **314** (1995), 141–147.

[BGK+97] C. Benecke, T. Grüner, A. Kerber, R. Laue, and T. Wieland, *Molecular structure generation with molgen, new features and future developments*, Fresenius J. Anal. Chem. **358** (1997), 23–32.

[BLP89] C. A. Brown, Finkelstein L., and P. W. Jr Purdom, *A new base change algorithm for permutation groups*, SIAM J. Comput **18** (1989), no. 5, 1037–1047.

[BLVS+93] A. Björner, M. Las Vergnas, B. Sturmfels, N. White, and G. M. Ziegler, *Oriented matroids*, Cambridge University Press, Cambridge, 1993.

[Bok92] J. Bokowski, *Oriented matroids*, Handbook of Convex Geometry (P. Gruber and J. M. Wills, eds.), Elsevier, Amsterdam, 1992.

[CF91] G. Cooperman and Larry Finkelstein, *A strong generating test and short presentation for permutation groups*, J. Symb. Comp. **12** (1991), 475–497.

[CH88] G. M. Crippen and T. F. Havel, *Distance geometry and molecular conformation*, Georg Thieme Verlag, Taunton, 1988.

[DDH82] A. Dreiding, A. Dress, and H. Haegi, *Classification of mobile molecules by category theory*, Studies in Phys. and Theor. Chem. **8** (1982), 341–352.

[DK70] A. Dress and M. Küchler, *Vorlesungsskript: Zur Darstellungstheorie endlicher Gruppen I*, Universität Bielefeld, 1970.

[DW80] A. Dreiding and K. Wirth, *The multiplex. a classification of finite ordered point sets in oriented d-dimensional space*, MATCH — Commun. Math. Comput. Chem. **8** (1980), 341–352.

[Far78a] I. A. Faradzhev, *Constructive enumeration of combinatorial objects*, Problèmes Combinatoires et Théorie des Graphes **260** (1978), 131–135, Colloq. Internat. CNRS, University of Orsay, Orsay 1976.

[Far78b] ———, *Generation of nonisomorphic graphs with a given degree sequence*, Algorithmic Studies in Combinatorics, pp. 11–19, NAUKA, Moskau, 1978, In Russisch.

[Fin01] L. Finschi, *A graph theoretical approach for reconstruction and generation of oriented matroids*, Ph.D. thesis, ETH Zürich, 2001.

[Ger04] M. Gerdts, *SQP V1.1 – Ein Fortran77-Programmpaket zur Lösung von nichtlinearen, restringierten Optimierungsproblemen*, Universität Bayreuth, Bayreuth, 2004.

[Gin93] M. L. Ginsberg, *Dynamic backtracking*, J. Art. Int. Res. **1** (1993), 25–46.

[GKL96] R. Grund, A. Kerber, and R. Laue, *Construction of discrete structures, especially isomers.*, Discrete Appl. Math **67** (1996), 115–126.

[GKL+00] R. Gugisch, A. Kerber, R. Laue, M. Meringer, and J. Weidinger, *Molgen– comb, a software package for combinatorial chemistry*, MATCH — Commun. Math. Comput. Chem. **41** (2000), 189–203.

[Gru95] R. Grund, *Konstruktion molekularer Graphen mit gegebenen Hybridisierungeben und überlappungsfreien Fragmenten*, Bayreuther Mathematische Schriften **49** (1995), 1–113.

[Grü95] T. Grüner, *Ein neuer Ansatz zur rekursiven Erzeugung von schlichten Graphen*, Master's thesis, Universität Bayreuth, 1995.

[HEO05] D. F. Holt, B. Eick, and E. A. O'Brien, *Handbook of computational group theory*, Discrete Mathematics and its Applications, Chapman & Hall/Crc, 2005.

[Jer86] M. Jerrum, *A compact representation for permutation groups*, J. Algorithms **7** (1986), 60–78.

[Kas] P. Kaski, *Isomorph-free exhaustive generation of designs with prescribed groups of automorphisms*, SIAM Journal on Discrete Mathematics, (to appear).

[Ker99] A. Kerber, *Applied finite group actions*, 2. ed., Springer, Berlin, Heidelberg, New York, 1999.

[Knu79] D. E. Knuth, *Lexicographic permutations with restrictions*, Discrete Appl. Math **1** (1979), 117–125.

[KTZ90] M. H. Klin, S. S. Tratch, and N. S. Zefirov, *2d-configurations and clique-cyclic orientations of the graphs* $l(k_p)$, reports in mol. theory (1990), 149–163.

[KÖTZ] P. Kaski, P. R. J. Östergård, S. Topalova, and R. Zlatarski, *Steiner triple systems of order 19 and 21 with subsystems of order 7*, Discrete Mathematics, (to appear).

[Lan92] E. Lang, *Datenstrukturen und Algorithmen für Permutationsgruppen*, Diplomarbeit, Universität Bayreuth, 1992.

[Lau93] R. Laue, *Construction of combinatorial objects — a tutorial*, Bayreuther Mathematische Schriften **43** (1993), 53–96.

[McK81] B. D. McKay, *Practical graph isomorphism*, Congressus Numerantium **30** (1981), 45–87.

[McK90] ———, *nauty user's guide (version 1.5)*, Tech. report, Computer Science Department, Australian National University, 1990.

[McK98] ———, *Isomorph-free exhaustive generation*, J. Algorithms **26** (1998), 306–324.

[NCSD79] J. G. Nourse, R. E. Carhart, D. H. Smith, and C. Djerassi, *Exhaustive generation of stereoisomers for structure elucidation*, J. Am. Chem. Soc. **101** (1979), 1216–1223.

[Nil03] J. N. Nilsson, *Artificial intelligence, a new synthesis*, Morgan Kaufmann Publishers, Inc., San Francisco, 2003.

[Nou79] J. G. Nourse, *The configuration symmetry group and its application to stereoisomer generation, specification, and enumeration*, J. Am. Chem. Soc. **101** (1979), 1210–1215.

[NSCD80] J. G. Nourse, D. H. Smith, R. E. Carhart, and C. Djerassi, *Computer-assisted elucidation of molecular structure with stereochemistry*, J. Am. Chem. Soc. **102** (1980), 6289–6295.

[Rea78] R. C. Read, *Everyone a winner*, Annals of Discrete Mathematics **2** (1978), 107–120.

[RG96] J. Richter-Gebert, *Two interesting oriented matroids*, Documenta Mathematica **1** (1996), 137–148.

[Sch93] B. Schmalz, *Verwendung von Untergruppenleitern zur Bestimmung von Doppelnebenklassen*, Bayreuther Math. Schr. **31** (1993), 109–143.

[Sim71] C. C. Sims, *Computation with permutation groups*, Proceedings of the Second Symposium on Symbolic and Algebraic Manipulation (New York) (S. R. Petrick, ed.), 1971, pp. 23–28.

[TZ87] S. S. Tratch and N. S. Zefirov, *Combinatorial models and algorithms in chemistry. the ladder of combinatorial objects and its application to the formalization of structural problems of organic chemistry*, Principles of Symmetry and Systemology in Chemistry (N.F. Stepanov, ed.), Moscow, Moscow State University Publ., 1987, (in Russian), pp. 54–86.

[TZ96] ———, *Algebraic chirality criteria and their application to chirality classification in rigid molecular systems*, J. Chem. Inf. Comput. Sci. **36** (1996), 448–464.

[Wan] M. Wang, *Canonical forms of discrete objects for databases and internet data exchange*, Ph.D. thesis, University of Bayreuth, (to appear).

[Wie64] H. Wielandt, *Finite permutation groups*, Academic Press, New York, 1964.

[Wie94] T. Wieland, *Erzeugung, Abzählung und Konstruktion von Stereoisomeren*, MATCH — Commun. Math. Comput. Chem. **31** (1994), 153–203.

[Wil85] S. Gill Williamson, *Combinatorics for computer science*, Computer Science Press, Rockville, 1985.

[WKL96] T. Wieland, A. Kerber, and R. Laue, *Principles of the generation of constitutional and configurational isomers.*, J. Chem. Inf. Comput. Sci. **36** (1996), 413–419.

[Zie96] G. M. Ziegler, *Oriented matroids today*, Electronic J. of Comb. **Dynamic Survey 4** (1996), 1–39.

[Zla91] L. A. Zlatina, *Mathematical models of generation of stereoisomers and building space models of molecules*, Ph.D. thesis, All-Union Research Institute of Organic Synthesis, 1991.

Index