Intramolekulares Gene Assembly

Diplomarbeit

Rainer Hausdorf
Matrikelnr.: 3081998

Betreuerin: Dr.-Ing. Monika Sturm
Betreuender Hochschullehrer: Prof. Franz Baader

Institut für Theoretische Informatik
Fakultät Informatik
Technische Universität Dresden

Beginn : 06.12.2008
Abgabe : 04.06.2009

Technische Universität Dresden
Lehrstuhl für Automatentheorie
Institut für Theoretische Informatik

05.12.2008

Aufgabenstellung Diplomarbeit

Intramolekulares Gene Assembly

Bearbeiter: Rainer Hausdorf (Informatik)

Das *Gene Assembly Modell* wurde zunächst in zwei unterschiedlichen Ansätzen wissenschaftlich bearbeitet und veröffentlicht. Basis der Unterscheidung bildete die Anzahl der miteinander agierenden Moleküle. Darauf aufbauend ging es in weiterführenden Arbeiten darum, zu untersuchen, inwieweit selbst in-vitro-Ansätze über diesen Vertreter der in-vivo-Modelle realisiert werden können. Im Zusammenhang mit diesen Untersuchungen setzten sich drei Modellierungen zum *Gene Assembly* durch, das universelle, das einfache und das elementare Gene Assembly Modell. In der Diplomarbeit sollen diese drei Modellierungen vorgestellt werden und anhand relevanter Eigenschaften untersucht, verglichen und falls notwendig in ihrer Definition ergänzt werden.

Folgende Punkte sollen dabei beachtet werden:

- Nachweis von Modelleigenschaften
  - Vollständigkeit
  - Konfluenz
  - Entscheidbarkeit von Gene Assembly
  - Eigenschaften von Genmustern
- Bewertung der Ergebnisse

Die Bearbeitung des Themas umfasst das Studium folgender Literaturstellen:

- Miika Langille, Ion Petre and Vladimir Rogojin. *Three models for gene assembly in ciliates: a comparison.* Technical report 878, TUCS, 2008.
- Ion Petre and Vladimir Rogojin. *Decision problems for shuffled genes.* Information and Computation, 206 (11), 1346-1352, Elsevier, 2008.
- Miika Langille and Ion Petre. *Simple gene assembly is deterministic.* Technical report 756, TUCS, 2007.
- Miika Langille and Ion Petre. *Sequential vs. parallel complexity in simple gene assembly.* Theoretical Computer Science, 395 (1), 24-30, Elsevier, 2008.

Organisation und Termine

Betreuerin: Dr. M. Sturm
Verantw. Hochschullehrer: Prof. F. Baader
Bearbeitungszeitraum: 06.12.2008 - 05.06.2009
Zwischenbericht: Ende 2/2009

Inhaltsverzeichnis

Kapitel 1

Einführung

Die vorliegende Arbeit *Intramolekulares Gene Assembly in Ciliaten* präsentiert eine Zusammenfassung relevanter Ergebnisse aus diesem Forschungsgebiet. Ausgehend vom Titel der Arbeit stellen sich drei Fragen: Wer oder was sind Ciliaten? Was ist Gene Assembly und was bedeutet in diesem Zusammenhang „intramolekular“?

Gene Assembly in Ciliaten

Ciliaten sind ein 7500 Arten umfassender Organismenstamm einzelliger Eukaryoten [Pre94]. Zwei Alleinstellungsmerkmale grenzen Ciliaten als eigenständigen Stamm ab: Der Besitz haarähnlicher Cilien, für Fortbewegung und Nahrungsaufnahme, sowie das Vorhandensein von zwei in Funktion und Größe unterschiedlicher Zellkerne. Der Größere wird als Makronucleus bezeichnet, der Kleinere entsprechend als Mikronucleus. Der Makronucleus ist der genetisch aktive Kern, in welchem die Prozesse zur Steuerung der Zelle stattfinden. Dagegen ist der Mikronucleus ein ruhender Speicher genetischen Materials. Beide Kerne enthalten die gleiche genetische Information, wobei es Unterschiede im Aufbau des Genoms gibt.

Während einer bestimmten Phase der Paarung zweier Ciliaten wird das mikronucleare Genom in das Genom eines Makronucleus umgewandelt. Dieser Transformationsprozess wird als *Gene Assembly* bezeichnet. Aufgrund des stark unterschiedlichen Genomaufbaus von Mikronucleus und Makronucleus gilt Gene Assembly als der aufwendigste, bekannte DNA Prozess in lebenden Organismen [EHP$^+$03]. Eine zugängliche Einführung zu Gene Assembly und dessen *Gene Assembly Operationen* bietet [HPR03]. Kapitel 2 der vorliegenden Arbeit vermittelt Grundlagen zu DNA, Ciliaten, sowie Gene Assembly und dessen Operationen. Ausführliche Betrachtungen zum Genomaufbau von Ciliaten finden sich beispielsweise in [Pre94].

Gene Assembly im DNA-Computing

Arbeiten von Laura F. Landweber und Lila Kari [LK98, LK99] zeigten die berechnungstechnische Natur von Gene Assembly, wodurch sich Gene Assembly zu einem Teilbereich des Forschungsgebiets *DNA-Computing* entwickelte.
DNA-Computing zählt neben Quantum Computing, Neural Computing und Evolutionary

Computing zum großen interdisziplinären Natural Computing, welches sich mit von der Natur inspirierten Rechenvorgängen und natürlichen Systemen beschäftigt. Hauptziel des Forschungsgebiets ist die Entwicklung alternativer Computingkonzepte mit entsprechender Hard- und Software. Diese Computingkonzepte sollen konventionelle Rechentechnik bei der Bearbeitung extrem rechenintensiver Aufgaben und insbesondere bei der Lösung kombinatorischer Suchprobleme (*NP-Probleme*) in den Punkten Rechengeschwindigkeit, Rechenparallelität, Speicherdichte und Effizienz weit übertreffen.
Die Idee des DNA-Computing besteht im Einsatz von DNA-Molekülen als Datenträger, auf welchen chemische und physikalische Prozesse die Ausführung von Operationen nachbilden und somit Rechenvorgänge realisieren. Begründer des DNA-Computing ist Leonard M. Adleman. Ihm gelang 1994 eine durch DNA repräsentierte Instanz des Hamiltonkreis, ein Vertreter der NP Komplexitätsklasse, mittels molekularbiologischer Techniken zu lösen. Er zeigte dadurch erstmalig, dass die praktische Nutzung von DNA für Berechnungen möglich ist [Adl94]. Ein zusammenfassender Überblick über die Entwicklung des DNA-Computing mit seinen verschiedenen Ansätzen findet sich in [HS04].

Gene Assembly Modelle

Für Gene Assembly existieren zwei Modellrichtungen, deren Unterschied in der Anzahl an Gene Assembly Operationen beteiligter Moleküle begründet ist. Das so genannte *intermolekulare Gene Assembly Modell* wurde in [LK98] formuliert. Intermolekular bedeutet, dass Operationen zwischen mehr als einem Molekül stattfinden können.
Der zweite Modellansatz, welcher Gegenstand der vorliegenden Arbeit ist, wird als *intramolekulares Gene Assembly Modell* bezeichnet und wurde in [PER01] eingeführt. Dieses Modell basiert auf intramolekularen Operationen, das heißt verschiedene Teile desselben Moleküls agieren miteinander. An Operationen dieses Modells ist demnach immer nur ein Molekül beteiligt. Forschungen zum intramolekularen Gene Assembly Modell konzentrieren sich in der Mehrzahl auf formale und biologische Eigenschaften des Gene Assembly Prozesses. Dabei stehen Themen wie mögliche Strategien von Gene Assembly oder mögliche aus Gene Assembly resultierende Gene im Vordergrund.
Neben dem ursprünglichen intramolekularen Gene Assembly Modell, welches heute als *universelles Modell* bezeichnet wird, entwickelten sich zwei weitere intramolekulare Modelle. Ein *elementares Modell* [HPRR05, HPRR08] und ein so genanntes *einfaches Modell* [LP06]. Beide Modelle beruhen auf einer Restriktion der Gene Assembly Operationen zu *vereinfachten Gene Assembly Operationen*. Die Unterteilung in das elementare und das einfache Modell ist durch eine differenzierte Auslegung der Definition von vereinfachten Gene Assembly Operationen begründet. Innerhalb von Kapitel 4 wird ausführlich auf die Thematik der unterschiedlichen Operationsdefinitionen eingegangen.
Wie eine Reihe wissenschaftlicher Arbeiten, darunter [LP06, LP08, LPR08, PR08] zeigen, haben die teils geringen Abweichungen zwischen den drei intramolekularen Gene Assembly Modellen starke Auswirkungen auf den Charakter der Modelle hinsichtlich ausgewählter Eigenschaften. Die Ergebnisse dieser Arbeiten werden in Kapitel 5 zusammengefasst, wobei ein Vergleich der drei Modelle vorgenommen wird.
Die Transformation eines mikronuclearen Gens mittels Gene Assembly zu einem makronuclearen Gen kann als Sortierung vorzeichenbehafteter Permutationen abstrahiert werden. Ein Gen wird dabei eindeutig durch eine vorzeichenbehaftete Permutation beschrieben. Diese Form der

Darstellung wird in Kapitel 3 und 4 vorgestellt. In Kapitel 5 wird die Frage aufgeworfen, ob es möglich ist, zu einem Gene Assembly Modell vorzeichenbehaftete Permutationen zu charakterisieren, welche mit Hilfe der Gene Assembly Operationen sortiert werden können. Mit anderen Worten, ist es ausgehend von bestimmten Merkmalen einer gegebenen vorzeichenbehafteten Permutation möglich, zu schließen, ob diese durch Gene Assembly sortiert werden können. Während für das elementare Modell in der Tat solch eine Charakteristik von sortierbaren Permutationen existiert [HPRR08], gilt dies im einfachen Modell als offenes Problem [LPR08, Seite 10, 12], [LP08, Seite 29]. In Kapitel 6 geben wir mit der Charakterisierung für einen Typ von Gene Assembly Operationen, der so genannten sd-Operation, einen ersten Schritt zur Lösung dieses Problems an.

Kapitel 2

Biologischer Hintergrund

Dieses Kapitel behandelt Gene Assembly von der biologischen Seite. Es erfolgt eine Vorstellung über den Aufbau von DNA. Wir gehen näher auf den Stamm der Ciliaten ein und beschreiben die Unterschiede zwischen mikronuclearen und makronuclearen Genom. Anschließend widmen wir uns dem Gene Assembly Prozess und dessen Operation. Wir betrachten den molekularbiologischen Prozess der *homologen Rekombination*, welcher die Grundlage der Gene Assembly Operationen darstellt. Ausführliche Betrachtungen zu Molekularbiologie finden sich beispielsweise in [Alb03, Mun01]; zu Ciliaten in [Pre94].

2.1 Desoxyribonucleinsäure - DNA

Die Desoxyribonucleinsäure ist die Trägerin des Erbguts aller Organismen, den *Bacteria*, *Archaea* und *Eukarya*. Da unser Augenmerk bei den eukaryotischen Ciliaten liegt, beschränken wir uns auf die Betrachtung von DNA in dieser Organismengruppe. Eukaryoten zeichnet aus, dass sie einen echten Zellkern in sich tragen. In diesem Zellkern befindet sich der Großteil der DNA. Weitere, geringere Vorkommen finden sich in den Mitochondrien und Plastiden der Zelle. Die DNA ist in Chromosomen zusammengefasst, welche die kompakte Form eines DNA-Fadens darstellen und in sehr großen Längen vorliegen können. Die Organisation der DNA in Chromosomen schützt die DNA bei mechanischen Belastungen und sorgt für eine geordnete Trennung des genetischen Materials beim Zellteilungsprozess. Die Gesamtheit der Chromosomen einer Zelle, ein Chromosomensatz, liegt üblicherweise doppelt vor und wird als *diploid* bezeichnet. Ein einfacher Chromosomensatz, wie er in den geschlechtlichen Samen- und Eizellen vorhanden ist, wird *haploid* genannt.

Vom Nucleotid zum DNA-Einzelstrang

Die DNA ist ein *Polynucleotid*, ein langkettiges Biomolekül, welches sich aus der Verbindung von *Nucleotiden* zusammensetzt. Ein einzelnes Nucleotid besteht aus einem Molekül Zucker, einer Phosphorsäure- bzw. Phosphatgruppe und einer organischen DNA-Base. Das Zuckermolekül ist eine aus fünf Kohlenstoffatomen bestehende Desoxyribose. Die fünf Bindungsstellen der Desoxyribose werden von 1´ bis 5´ durchnummeriert (eine je Kohlenstoffatom). An der 3´-Position ist eine OH-Gruppe gebunden. An der 5´-Position des Zuckers befindet sich die

Phosphatgruppe des Nucleotids. Die Phosphatgruppen geben der DNA eine gesamtnegative Ladung. Sie sind es auch, die die DNA chemisch gesehen zur Säure machen.
Während Phosphatgruppe und Desoxyribose bei jedem Nucleotid gleich sind, kommen bei dem Bestandteil Base mit *Adenin, Guanin, Thymin* oder *Cytosin* vier Varianten in Frage. Nucleotide werden daher, entsprechend der Abkürzung ihrer Base, mit den Buchstaben A, G, T, oder C bezeichnet. Die Base befindet sich an der 1´-Position der Desoxyribose.
Eine Verknüpfung zweier Nucleotide kann zwischen der Phosphatgruppe am 5´-Atom des einen Nucleotids und der an 3´-Position vorhandenen OH-Gruppe eines anderen Nucleotids entstehen. Solch eine Verbindung wird *Phosphodiesterbindung* genannt. Über weitere Phosphodiesterbindungen an der 3´- und 5´-Position können sich erneut Desoxyribosen anlagern, so dass ein *DNA-Einzelstrang* entsteht. Die Enden eines DNA-Einzelstrang-Moleküls lassen sich aufgrund ihres chemischen Aufbaus unterscheiden. DNA-Stränge sind somit gerichtet. Die für eine weitere Bindung bereitstehende Phosphatgruppe bildet das *5´-Ende* und die OH-Gruppe das *3´-Ende* eines DNA-Einzelstrangs.

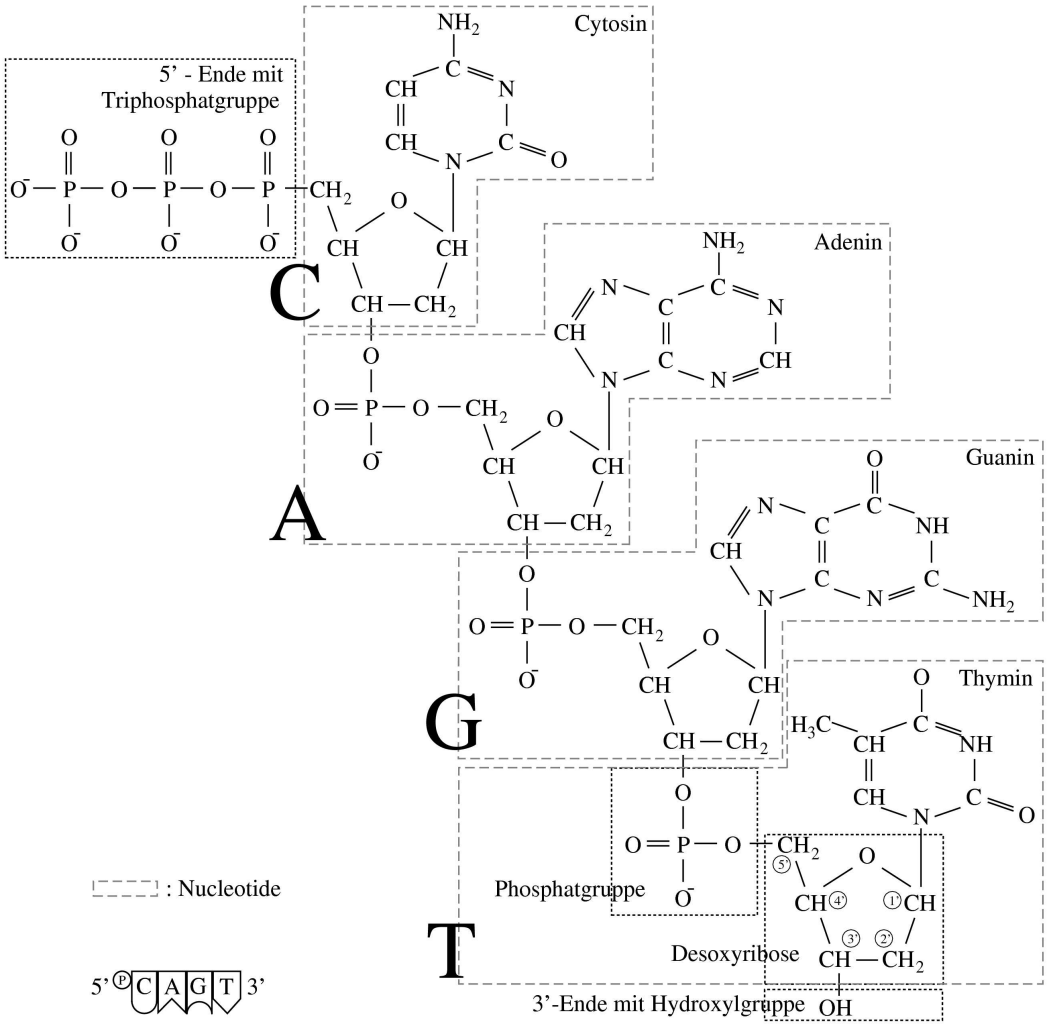


Abbildung 2.1: Primärstruktur des DNA-Einzelstranges 5'P-CAGT-3'

Ein DNA-Einzelstrang ist eine Sequenz der Nucleotide A, C, G und T. Die Enden der Nucleotidsequenz werden mit 5´ und 3´ angegeben [HS04]. Die 5´-3´-Richtung wird als *sense* und die 3´-5´-Richtung als *antisense* bezeichnet. Es ist allgemein üblich, Nucleotidsequenzen in 5´-3´-Leserichtung anzugeben.
Durch die Sequenz der Basen ist die genetische Information eines Organismus bestimmt. Daher ist ein Nucleotid die kleinste informationstragende Einheit in der DNA, ein monomerer Baustein. Durch die Prozesse der *Transkription* und *Translation* wird genetische Information in der Zelle verarbeitet. Eine ausführliche Schilderung von Transkription und Translation findet sich in [Alb03], soll aber hier nicht weiter betrachtet werden.

Vom Einzelstrang zum Doppelstrang

Unter bestimmten Voraussetzungen ist es möglich, dass sich zwei benachbarte DNA-Einzelstränge zu einem *DNA-Doppelstrang* verbinden. Über so genannte *Wasserstoffbrücken* können sich die Nucleotide A und T und die Nucleotide C und G aneinanderlagern. Diese Eigenschaft wird als *Komplementarität* von Nucleotiden bezeichnet. Eine einzelne Verbindung wird *Basenpaarung* genannt.
Basenpaarungen können nicht nur zwischen zwei Nucleotiden gebildet werden, sondern auch zwischen Nucleotidsequenzen. Bedingung hierfür ist, dass jede Base eines DNA-Einzelstranges zu der gegenüberliegenden Base eines entgegengesetzt ausgerichteten DNA-Einzelstranges komplementär ist. So entsteht eine doppelbändige Kette komplementärer Nucleotide, ein DNA-Doppelstrang. Ein Strang verläuft dabei in 5´-3´-Richtung, der andere umgekehrt in 3´-5´-Richtung. Dies wird als *antiparallel* bezeichnet wird.

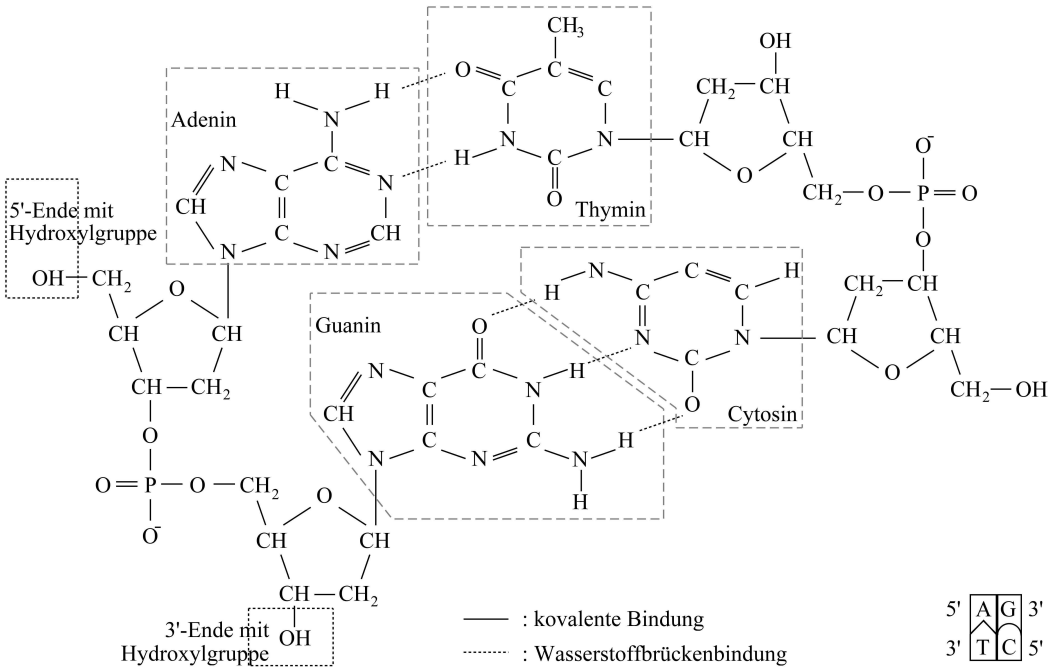


Abbildung 2.2: Sekundärstruktur des DNA-Doppelstranges $\frac{5'-AG-3'}{3'-TC-5'}$ mit Kennzeichnung der Wasserstoffbrückenbindungen.

In der Bezeichnung der äußeren Enden eines DNA-Doppelstranges wird unterschieden, ob ein Strangende einen *Einzelstrangüberhang* vorweist. Ein Einzelstrangüberhang ist eine zumeist sehr kurze, ungepaarte Nucleotidsequenz. DNA-Doppelstrangenden mit Einzelstrangüberhang werden als *sticky* bezeichnet. Es wird zwischen 3´- und 5´-Einzelstrangüberhängen unterschieden. Ohne Einzelstrangüberhang wird das DNA-Doppelstrangende *blunt* genannt.

Im strukturellen Aufbau von DNA wird zwischen *linearen* und *nicht linearen* DNA-Strängen unterschieden. Ein DNA-Strang heißt linear, wenn er ausschließlich aus einer Sequenz (linearen Abfolge) von Nucleotidpaaren und/oder Nucleotiden besteht, genau zwei äußere Enden besitzt und an keiner Stelle nicht komplementäre, antiparallel ausgerichtete Nucleotide enthält. Die Eigenschaft linear wird im DNA-Computing bevorzugt, da die Anwendung vieler molekularbiologischer Prozesse auf linearen DNA-Strängen besser abläuft, als auf nicht linearen Strängen. Ein weiterer Vorteil ist, dass sich lineare DNA-Doppelstränge im Vergleich zu nicht linearer DNA über eine sehr einfache Struktur auszeichnen und *linearisiert* notiert werden können. Dazu werden in einer zweizeiligen Darstellung die oberen Einzelstrangbereiche in 5´-3´-Richtung und die unteren in 3´-5´-Richtung angegeben. Über Basenpaarungen verbundene Nucleotide stehen dabei direkt untereinander. Nicht lineare DNA-Doppelstränge sind die Folge von Basenfehlpaarungen, so genannte Mismatches. Abbildung 2.3 zeigt Beispiele für lineare beziehungsweise nicht lineare DNA-Stränge. Auch wenn von „linearen“ DNA-Doppelstrang-Molekülen die Rede ist, sei darauf hingewiesen, dass der Strang die Form einer Doppelhelix annimmt, indem sich die beiden DNA-Einzelstränge umeinander winden. Sinnbildlich steht hierfür das Bild einer in sich verdrehten Strickleiter.

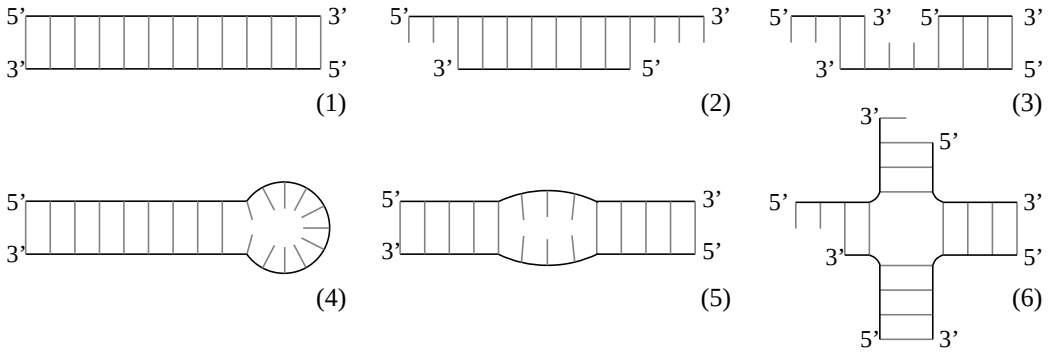


Abbildung 2.3: Beispiele für lineare DNA-Stränge. (1) linearer DNA-Strang mit zwei blunt-Enden. (2) linearer DNA-Strang mit 5´-sticky-Ende und 3´-sticky-Ende. (3) linearer DNA-Strang mit 5´-sticky-Ende, blunt-Ende und einem Einzelstrangüberhang. (4) nicht linearer DNA-Strang mit Hairpinstruktur. (5) nicht linearer DNA-Strang mit so genannter Bubble. (6) nicht linearer DNA-Strang mit 4-Way-Junction.

2.2 Kerndimorphismus in Ciliaten

Ciliaten (dt.: Wimpertierchen) sind einzellige Eukaryoten, welche eine Größe von circa 50-300 Mikrometer erreichen [Pre94]. Ciliaten kommen sowohl in Süß- als auch Salzwasser oder auch in feuchten Böden vor. Mit etwa 7500 bekannten Arten gelten die Ciliaten als ein wichtiger und vielfältiger Organismenstamm.

Zwei Alleinstellungsmerkmale zeichnen Ciliaten aus: Zum einen der bereits angesprochene Kerndimorphismus, zum anderen die namensgebenden Cilien. Diese sind haarähnliche Fortsätze, welche die Zelloberfläche teilweise oder vollständig bedecken. Durch gerichtetes Schlagen der Cilien dienen diese der Fortbewegung. Zusätzlich wird durch Strudelbewegungen der Cilien die Nahrungsaufnahme der Ciliaten unterstützt, indem Nahrungspartikel durch eine mundähnliche Öffnung in die Zelle getrieben werden.
Kerndimorphismus bezeichnet die Existenz zweier in Funktionalität und Größe unterschiedlicher Zellkerne innerhalb derselben Zelle. Der größere der beiden Kerne, der Makronucleus, ist der genetisch aktive Zellkern. In ihm laufen die nötigen Prozesse für die Umsetzung der genetischen Information ab. Der kleinere Mikronucleus ist genetisch inaktiv.
Beide Kerne enthalten dieselbe genetische Information. Dennoch sind die DNA-Sequenzen verschieden. Im Makronucleus trägt jedes enthaltene DNA-Molekül ein Gen, welches in einer fortlaufenden DNA-Sequenz vorliegt. Im Mikronucleus ist die Sequenz von jedem Gen in *MDSs (macronuclear destined sequences)* aufgeteilt. Die MDSs werden durch nicht kodierende Bereiche, so genannte *IESs (internally eliminated sequences)*, voneinander getrennt. Die MDSs können zudem in Reihenfolge und Ausrichtung unsortiert sein. Die einzelnen Gene selbst sind im Mikronucleus außerdem durch nicht kodierende DNA-Abschnitte weit voneinander getrennt. Diese Bereiche werden als *Spacer-DNA* bezeichnet. Weniger als 5 Prozent der mikronuclearen DNA kodiert genetische Information.
Besonders komplexe Strukturen sind in der Ciliatengruppe *Stichotrich* zu finden, weshalb der Forschungsschwerpunkt im Allgemeinen bei diesen Einzellern liegt.

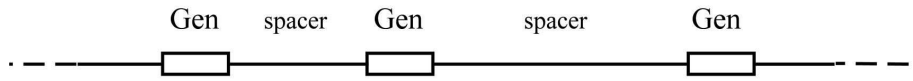


Abbildung 2.4: Gene entlang eines Segmentes chromosomaler DNA. Nach [HPR03].

2.2.1 Ciliatengruppe Stichotrichs

Es gibt wenige exzessiv molekulargenetisch untersuchte Ciliatengruppen. Eine davon ist die so genannte Gruppe der Stichotrichs. Die Gruppe ist von besonderer Bedeutung, da sich die DNA der Stichotrichs während ihrer evolutionären Entwicklung stark modifiziert hat. Diese Veränderungen erfordern erstaunliche Manipulationen der DNA in den Zellen der Stichotrichs. Stichotrichs vermehren sich bei guten Nahrungs- und Umgebungsverhältnissen über einfache Querteilung. Bei schlechten Bedingungen sichert ein Stichotrich das Überleben durch Zellpaarung mit einem zweiten Stichotrich gleicher Art (Abbildung 2.5). Während der Zellpaarung wird durch den Transformationsprozess Gene Assembly der Mikronucleus in einen Makronucleus umwandelt.

Stichotrichs haben je nach Art mindestens zwei oder mehr Mikronuclei und zwei oder mehr Makronuclei. Abbildung 2.5 veranschaulicht die Zellpaarung bei Stichotrichs.

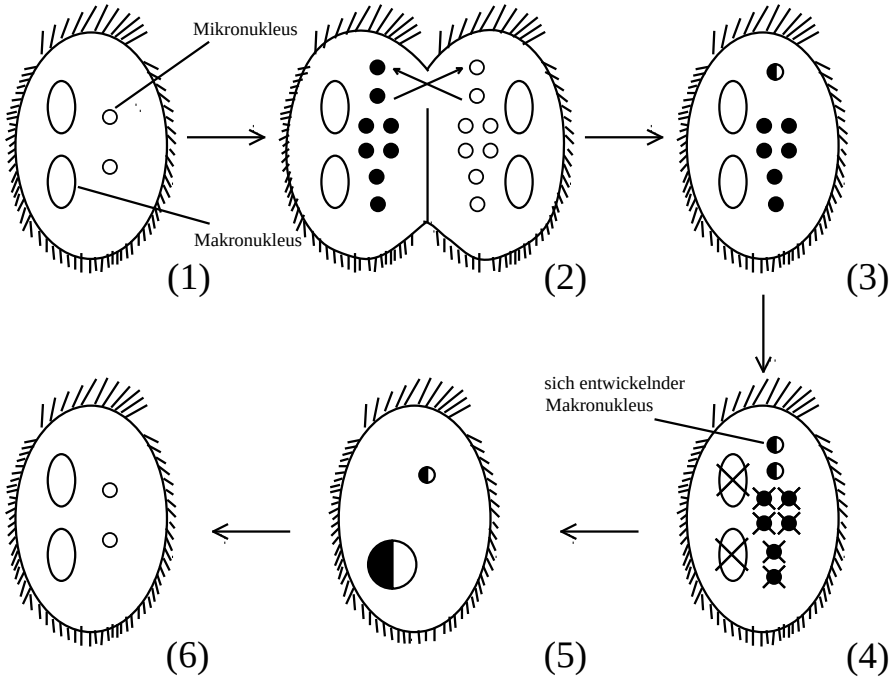


Abbildung 2.5: Zellpaarung bei Stichotrichs. Nach [EHP+03]

In Abbildung 2.5 zeigt (1) das Cilium vor der Paarung. Lagern sich zwei Stichotrichs wie es in (2) für die Paarung aneinander, bilden sie zwischen sich einen Verbindungskanal. Die Mikronuclei durchlaufen dann die Vorgänge der *Mitose* (Zellkernteilung) mit anschließender *Meiose* (Reifeteilung). Über den Verbindungskanal wandert je ein haploider Mikronucleus des einen Stichotrichs hinüber zur anderen Stichotrichzelle. Dort verschmelzen sie mit einem in der Zelle verbliebenen haploiden Mikronucleus zu je einem diploiden Mikronucleus. Die beiden Stichotrichs trennen sich anschließend wieder voneinander. Diesen Zustand zeigt 2.5 (3). In (4) hat sich der neue diploide Mikronucleus durch Mitose geteilt, während die zwei Makronuclei zusammen mit den restlichen haploiden Mikronuclei degeneriert sind und sich aufgelöst haben. Einer der beiden diploiden Mikronuclei wandelt sich durch Gene Assembly in einen Makronucleus. Dies ist in (5) zu sehen. Durch mitotische Teilungen von Mikro- und Makronucleus wird der Zyklus abgeschlossen und der Ausgangszustand erreicht (siehe 2.5 (6).

2.2.2 Umwandlung Mikronucleus zum Makronucleus

Das Genom des Mikronucleus ist in sehr langen Chromosomen organisiert [Pre94]. Gene auf diesen Chromosomen sind durch Spacer-DNA weit voneinander getrennt. Zusätzlich sind die Gene in sich unterbrochen durch nicht kodierende IESs (kurze Sequenzen von weniger als 100 Nucleotiden). Die kodierenden MDSs wiederum unterscheiden sich in ihrer Reihenfolge von ihrer Zielorganisation im Makronucleus. Zusätzlich können Invertierungen der MDSs vorliegen. Durch den Gene Assembly Prozess werden die Spacer-DNA und die IESs herausgetrennt, die MDSs in die makronucleare Zielordnung sortiert und Invertierungen aufgehoben.

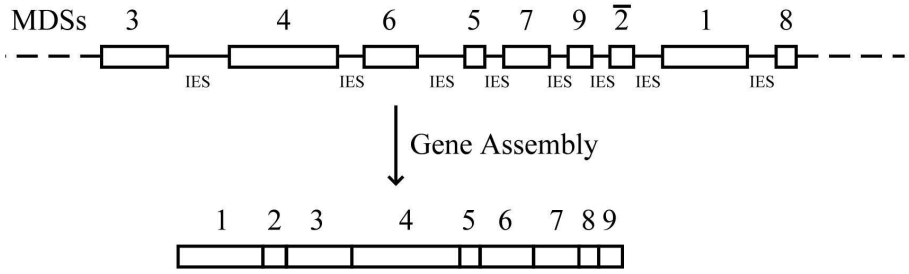


Abbildung 2.6: Diagramm des Gene Assembly Prozesses für das mikronucleare Gen, welches das Actin-Protein in Stichotrich *Sterkiella Nova* kodiert. Bei der makronuclearen Entwicklung werden die IESs herausgeschnitten und die MDSs miteinander verbunden. Nach [EHP+03]

Wie kann Gene Assembly funktionieren und nicht kodierende Bereiche von kodierenden unterscheiden? Wie werden MDSs in ihre korrekte Reihenfolge gebracht und wie ist eine Invertierung zu erkennen? Es zeigt sich, dass Ciliaten ihr Genom ähnlich einer „Linked List“ in der Informatik verwalten [Pre94]. Das Ende jeder MDS bildet eine kurze Sequenz (3-20 Nucleotide), welche identisch dem Beginn der sich in der makronuclearen Zielordnung direkt anschließenden MDS ist. Diese kurzen Sequenzen dienen somit der Genomorganisation in Ciliaten, ähnlich wie Pointer (Zeiger) der Speicheradressierung in der Informatik. Daher werden diese Sequenzen ebenfalls *Pointer* genannt. Es sind immer zwei Pointer zueinander identisch, weswegen von *Pointerpaaren* gesprochen wird. Des Weiteren beginnt die erste MDS eines Gens mit einem speziellen Beginnmarker, während die MDS am Ende eines Gens mit einem Abschlussmarker endet.
Diese Organisation ist die Grundlage dafür, dass Gene Assembly das mikronucleare Genom in die makronucleare Zielordnung bringt. Wie wir im nächsten Abschnitt sehen werden, ist dafür lediglich die (wiederholte) Ausführung dreier Operationen, namentlich *loop, hairpin* und *doubleloop*, nötig. Gene Assembly kann daher als begriffliche Zusammenfassung dieser Operationen aufgefasst werden.

2.3 Intramolekulare Gene Assembly Operationen

Den drei Gene Assembly Operationen loop, hairpin und doubleloop liegt der Vorgang der homologen Rekombination zugrunde. Bevor jede der Operationen einzeln betrachtet wird, soll zunächst geklärt werden, was sich hinter homologer Rekombination verbirgt und wie diese abläuft.

2.3.1 Homologe Rekombination

Rekombination ist ein molekularbiologischer Prozess, unter dem die Neukombination von DNA-Sequenzen verstanden wird [Alb03]. Es gibt zwei mögliche Anwendungen von Rekombination: Zum einen intermolekulare Rekombination, deren Voraussetzung das Vorhandensein zweier DNA-Moleküle ist. Zum anderen kann aber Rekombination auch auf verschiedenen Abschnitten desselben DNA-Moleküls stattfinden. Dabei handelt es sich um intramolekulare Rekombination.

Rekombination setzt die mit Hilfe von Enzymen ausgeführten Verfahren des Schnitts (*Cut*) und der Verknüpfung (*Ligation*) von DNA-Molekülen voraus.

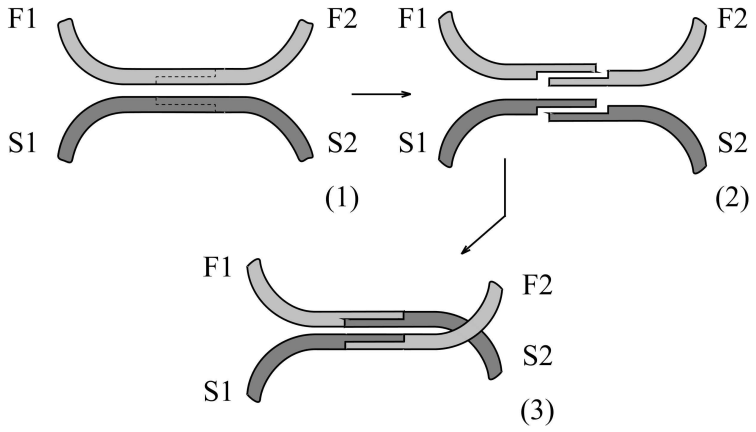


Abbildung 2.7: Homologe Rekombination. Nach [HPR03]

.

Für die homologe Rekombination sind Sequenzhomologien erforderlich. Das bedeutet, dass die an der Rekombination beteiligten DNA-Moleküle Bereiche mit identischen Basenpaarsequenzen enthalten müssen. Diese Sequenzen werden als *Schnittsequenzen* bezeichnet. Im ersten Schritt der Rekombination lagern sich diese identische Sequenzen aneinander (1). *Restriktionsenzyme* bewerkstelligen einen stufenartig versetzten Cut an der gleichen Position der jeweiligen Sequenz. Die Schnittposition wird im Weiteren als *Spaltstelle* bezeichnet werden (2). Die so entstandenen sticky-Enden der DNA-Moleküle wechseln gegenseitig die Position, so dass zwei neu kombinierte DNA-Stränge entstehen, wenn das Enzym *Ligase* im letzten Schritt der Rekombination die Cuts repariert (3).

2.3.2 Loop, Hairpin, Doubleloop

Die drei im folgenden Abschnitt vorgestellten Gene Assembly Operationen loop, hairpin und doubleloop, haben gemein, dass sie auf intramolekularer, homologer Rekombination basieren. Ihre Anwendung findet demnach auf demselben DNA-Strang statt. Die für Rekombination benötigten identischen Sequenzen sind die MDSs begrenzenden Pointer. Für die erfolgreiche Anwendung der Rekombination faltet sich das DNA-Molekül in sich selbst, so dass zwei identische Pointer, ein Pointerpaar, direkt aneinander liegen.

Loop Rekombination: ld

Die Anwendung der (*loop-, direct-repeat*)-*excision* Operation, kurz ld, bewirkt die Heraustrennung der im mikronuclearen Genom vorkommenden IESs. Der nicht kodierende Bereich, welcher bei der Ausführung von ld entfernt wird, muss von zwei identischen Pointern begrenzt sein. Durch Faltung des Strangs wird dieses Pointerpaar räumlich direkt nebeneinander gelegt.

Dabei entsteht die der Operation namensgebende Schleife (Loop) und die homologe Rekombination kann ablaufen. Die Pointer werden durch Enzyme an den Spaltstellen getrennt, die Bruchstellen vertauscht und Ligase-Enzyme verbinden die Stränge.

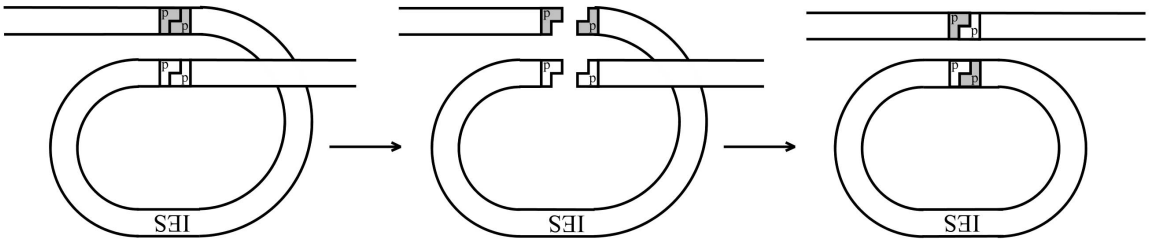


Abbildung 2.8: Loop-Rekombination

Durch die Operation entstehen zwei Moleküle, ein lineares und ein zirkuläres Molekül. Das zirkuläre Molekül ist die IES und somit Abfallprodukt. Es sei darauf hingewiesen, dass mittels ld ein beliebiger Bereich zwischen zwei identischen Pointern ausgeschnitten werden kann. Im intramolekularen Gene Assembly ist dieser Bereich automatisch Abfallprodukt, weil keine Möglichkeit besteht, das zirkuläre Molekül wieder mit dem Linearen zu vereinen. Daher darf im herausgetrennten Bereich kein MDS enthalten sein, weil eine korrekte Zusammensetzung der makronuclearen Ordnung nicht weiter möglich wäre.

Hairpin Rekombination: hi

Die Anwendung der (*hairpin, inverted-repeat*)-*excision* Operation, kurz hi, ist notwendig, wenn im mikronuclearen Molekül ein Pointerpaar vorliegt, bei dem ein Pointer invertiert zum Anderen ausgerichtet ist. Die zwei Pointer werden durch eine haarnadelähnliche Faltung des DNA-Moleküls, welches die Invertierung räumlich auflöst, direkt nebeneinander gelegt. So kann die homologe Rekombination stattfinden. Im Ergebnis liegt das DNA-Molekül so vor, dass der durch die Pointer begrenzte Abschnitt invertiert wurde.

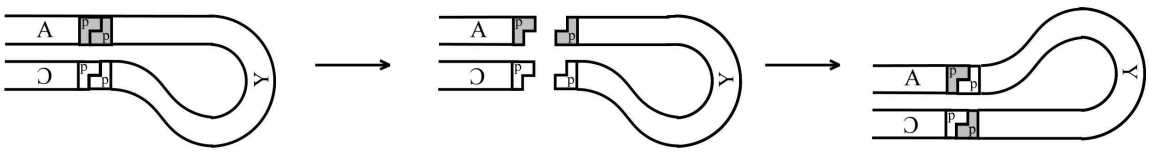


Abbildung 2.9: Hairpin-Rekombination

Doubleloop Rekombination: dlad

Die Ausgangsstellung für die Anwendung der Operation (*double loop, alternating direct repeat*)-*excision*, kurz dlad, sind zwei ineinander verschachtelte Pointerpaare (in der Abbildung p und q). Wie die Abbildung 2.10 zeigt, bilden die insgesamt vier Pointer miteinander drei Teilstränge.

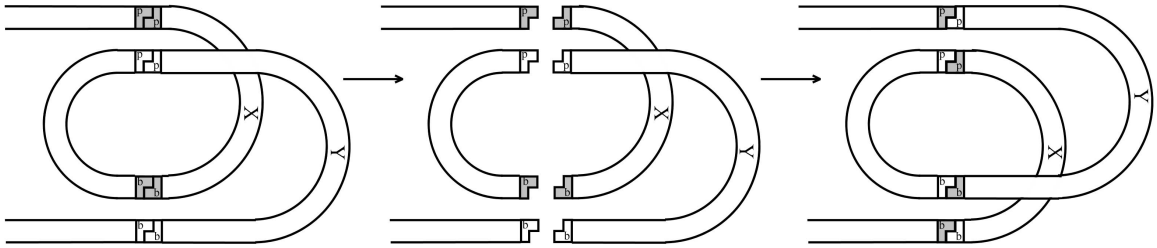


Abbildung 2.10: Doubleloop-Rekombination

Auch bei dieser Operation erfolgt, analog zu ld und hi, eine Faltung des mikronuclearen Moleküls. Es werden zwei Schleifen (double loop) gebildet, um die jeweiligen Pointer der zwei Pointerpaare räumlich direkt nebeneinander zu legen. An beiden Stellen findet homologe Rekombination statt. Im Ergebnis der dlad-Operation liegt das Ausgangsmolekül vor, wobei die zwei äußeren Teilstränge miteinander vertauscht wurden.

Kapitel 3

Mathematische Grundlagen

In diesem Kapitel werden Begriffe und Notationen für das Verständnis der vorliegenden Arbeit vorgestellt und erläutert. Eine kompakte Einführung in die Grundlagen der Theoretischen Informatik findet sich zum Beispiel in [Sch01].

3.1 Begriffe und Notationen

Für ein endliches Alphabet $\Sigma = \{a_1, \ldots, a_n\}$ definieren wir mit Σ^* das freie Monoid von Σ und nennen jedes Element von Σ^* ein *Wort*. Das leere Wort wird als $w = \varepsilon$ notiert. Für ein Wort $w \in \Sigma^*$ wird mit $|w|$ die Länge eines Wortes angegeben. So ist für $w = aabac \in \Sigma^*$ mit $\Sigma = \{a, b, c\}$ die Länge $|w| = 5$. Die Menge aller nicht leeren Wörter über Σ wird mit Σ^+ notiert. Es gilt $\Sigma^* = \Sigma^+ \cup \{\epsilon\}$. Mit $|\Sigma|$ wird die Anzahl der Symbole in Σ angegeben. Für zwei Wörter $u, v \in \Sigma^+$ sagen wir u ist ein *Teilwort* von v (notiert durch $u \leq v$), wenn $v = xuy$ für beliebige Wörter x, y. Das Wort u ist eine Subsequenz von v (notiert durch: $u \leq_s v$), wenn $u = a_1 a_2 \ldots a_m$, $a_i \in \Sigma$ und $v = v_0 a_1 v_1 a_2 v_2 \ldots a_m v_m$ für beliebige Wörter $v_i, 0 \leq i \leq m \in \Sigma^*$.

Beispiel 3.1 Gegeben sei das Alphabet $\Sigma = \{1, 2, \ldots, 9\}$. Sei $v \in \Sigma^*$ ein Wort mit $v =$ 1 2 3 4 5 6 7 8 9. Es gilt:

- $u_1 \leq_s v$, mit $u_1 = 259$, $u_1 \in \Sigma^*$;
- $u_2 \not\leq_s v$, mit $u_2 = 284$, $u_2 \in \Sigma^*$;
- $u_3 \leq_s v$, mit $u_3 = 345$, $u_3 \in \Sigma^*$. Wir sehen, dass u ebenso ein Teilwort von v ist.

Für einen gerichteten Graph $G = (V, E)$ gelten folgende Konventionen:

(i) Ein Knoten p heißt Wurzel von G, wenn kein $q \in V$ mit $(q, p) \in E$ existiert.

(ii) Der Graph $G = (V, E)$ ist zyklisch, wenn er eine Schleife enthält. Eine Schleife ist ein nicht leerer Pfad von q nach q für ein beliebiges $q \in V$.

(iii) Sind $p, q \in V$ wird mit $q \rightarrow_G^+ p$ ein nicht leerer Pfad von q nach p in G notiert. Wenn $(q, p) \in E$ schreiben wir $q \rightarrow_G p$.

(iv) Ist $\Theta \subseteq V$ und $p \in V$ notieren wir mit $\Theta \rightarrow_G^+ p$, wenn für alle $q \in \Theta$ gilt: $q \rightarrow_G^+ p$.

3.2 Permutationen

Sei $\overline{\Sigma} = \{\overline{a}_1, \ldots, \overline{a}_n\}$ mit $\Sigma \cap \overline{\Sigma} = \emptyset$. Für $p, q \in \Sigma \cup \overline{\Sigma}$ sagen wir, p und q besitzen dasselbe Vorzeichen, wenn entweder $p, q \in \Sigma$ oder $p, q \in \overline{\Sigma}$. Andernfalls sagen wir, p und q tragen unterschiedliche Vorzeichen. Ein Element e ist *nicht gekennzeichnet*, wenn $e \in \Sigma$. Wenn $e \in \overline{\Sigma}$ ist e *gekennzeichnet*.

Es sei $\Sigma^{\maltese} = (\Sigma \cup \overline{\Sigma})^*$. Für jedes $u \in \Sigma^{\maltese}$ mit $u = x_1 \ldots x_k$, $x_i \in (\Sigma \cup \overline{\Sigma})$ und $1 \leq i \leq k$, notieren wir $\|u\| = \|x_1\| \ldots \|x_k\|$, wobei $\|a\| = \|\overline{a}\| = a$, für alle $a \in \Sigma$. Wir notieren $\overline{u} = \overline{x_k} \ldots \overline{x_1}$, wobei $\overline{\overline{a}} = a$, für alle $a \in \Sigma$. Für zwei Alphabete Σ und Δ wird ein Mapping $f : \Sigma^{\maltese} \rightarrow \Delta^{\maltese}$ Morphismus genannt, wenn $f(uv) = f(u)f(v)$ und $f(\overline{u}) = \overline{f(u)}$.

Eine Permutation π über Σ ist eine Bijektion $\pi : \Sigma \rightarrow \Sigma$. Die Ordnungsrelation $(a_1, a_2, \ldots a_m)$ über Σ wird festgehalten, indem wir π als das Wort $\pi(a_1) \ldots \pi(a_m) \in \Sigma^*$ notieren. Eine *nicht vorzeichenbehaftete Permutation* π über Σ ist ein Wort $u \in \Sigma^*$. Eine *vorzeichenbehaftete Permutation* π über Σ ist ein Wort $u \in \Sigma^{\maltese}$, wobei $\|u\|$ eine Permutation über Σ ist. Eine vorzeichenbehaftete Permutation π ist sortiert, wenn sie in einer der beiden folgenden Formen vorliegt:

(i) $\pi = a_k a_{k+1} \ldots a_n a_1 \ldots a_{k-1}$, für $k \geq 1$. Wir sagen π ist eine *orthodox sortierte Permutation*.

(ii) $\pi = \overline{a_{k-1}} \ldots \overline{a_1 a_n} \ldots \overline{a_{k+1} a_k}$ für $k \geq 1$. In dem Fall sprechen wir davon, dass es sich bei π um eine *invertiert sortierte Permutation* handelt.

In beiden Varianten sagen wir für den Fall $k = 1$, dass π als *linear sortierte Permutation* vorliegt. In allen anderen Fällen ist π *zirkulär*.

Beispiel 3.2 Eine Permutation $\pi_1 = 12345$ ist eine orthodox sortierte Permutation und zusätzlich linear sortiert. Ebenso $\pi_2 = 34512$, welche die Eigenschaft zirkulär trägt. Die Permutationen $\pi_3 = \overline{54321}$ und $\pi_4 = \overline{15432}$ sind invertiert sortierte Permutationen, wobei π_3 auch linear sortiert ist, während π_4 zirkulär vorliegt. Die Permutation $\pi_5 = 12\overline{34}5$ ist weder orthodox noch invertiert sortiert. Permutationen die Elemente mit verschiedenen Vorzeichen enthalten sind niemals sortiert.

Ein sortierter Block in einer vorzeichenbehafteten Permutation π ist ein Teilwort zu π, von der Form $a_i a_{i+1} \ldots a_j$ oder der Form $\overline{a_j} \ldots \overline{a_{i+1} a_i}$, $1 \leq i \leq j \leq n$, wobei $a_{i-1}a_i$, $\overline{a_i a_{i-1}}$, $a_j a_{j+1}$, und $\overline{a_{j+1} a_j}$ keine Teilwörter von π sind. Einen sortierten Block notieren wir mit $[a_i, a_j]$, wobei a_i das linke und a_j das rechte Randelement des Blocks ist. Mit $S(\pi)$ wird die Anzahl sortierter Blöcke in π notiert.

3.3 Struktur einer Permutation

Um die Definition der Struktur einer Permutation anzugeben, wird zunächst der Morphismus $\xi_i : \Sigma^{\maltese} \rightarrow \Sigma^{\maltese}$ für alle $1 \leq i \leq |\Sigma|$ benötigt:

$$\xi_i(a_j) = \begin{cases} \varepsilon & \text{wenn } j = i; \\ a_j & \text{wenn } j < i; \\ a_{j-1} & \text{wenn } j > i; \end{cases}$$

mit $a_j \in (\Sigma \cup \overline{\Sigma})$.

Des Weiteren wird das Mapping $\sigma_i : \Sigma^{\maltese} \to \Sigma^{\maltese}$ benötigt, wobei für jedes Wort $u \in \Sigma^{\maltese}$, $\sigma_i(u)$ wie folgt gebildet wird:

(i) $\sigma_i(u) = u$, wenn $a_i a_{i+1} \not\leq u$, mit $a_i, a_{i+1} \in \Sigma$ oder,

(ii) $\sigma_i(u) = u$, wenn $a_{i+1} a_i \not\leq u$, mit $a_i, a_{i+1} \in \overline{\Sigma}$ oder,

(iii) $\sigma_i(u) = \xi_i(u)$ andernfalls.

Die *Struktur* einer vorzeichenbehafteten Permutation ist das Mapping $\sigma : \Sigma^{\maltese} \to \Sigma^{\maltese}$, so dass $\sigma(u) = (\sigma_1 \circ \sigma_2 \circ \ldots \circ \sigma_{|\Sigma|-1} \circ \sigma_{|\Sigma|})(u)$.

Beispiel 3.3 Gegeben sei die vorzeichenbehaftete Permutation $\pi = 1\overline{54}23 \in \Sigma^{\maltese}$ mit $\Sigma = \{1, 2, 3, 4, 5\}$, $|\Sigma| = 5$. Die Struktur $\sigma(\pi)$ wird folgendermaßen ermittelt:

$\pi_5 = \sigma_5(\pi) = \pi$	$\pi_2 = \sigma_2(\pi_3) = \xi_2(\pi_3) = 1\overline{3}2$
$\pi_4 = \sigma_4(\pi_5) = \xi_4(\pi_5) = 1\overline{4}23$	$\pi_1 = \sigma_1(\pi_2) = \pi_2$
$\pi_3 = \sigma_3(\pi_4) = \pi_4$	$\sigma(\pi) = \pi_1 = 1\overline{3}2$

Ein Wort $u \in \Sigma^{\maltese}$ ist *einfach* wenn $\sigma(u) = u$. Das bedeutet, dass weder das Teilwort $a_i a_{i+1}$ mit $a_i, a_{i+1} \in \Sigma$ noch das Teilwort $a_{i+1} a_i$ mit $a_i, a_{i+1} \in \overline{\Sigma}$ in u enthalten ist. Ein Wort u ist das *einfache, zu v assozierte Wort* wenn $\sigma(v) = u$ mit $v \in \Sigma^{\maltese}$.

Abschließend ist noch anzumerken, dass eine sortierte Permutation π lediglich die Strukturen $\sigma(\pi) = 1$, $\sigma(\pi) = 21$, $\sigma(\pi) = \overline{1}$ oder $\sigma(\pi) = \overline{12}$ annehmen kann.

Kapitel 4

Drei Gene Assembly Modelle

Aufbauend auf den Gene Assembly Operationen ld, hi und dlad wurde in [EPR99, EPR01] ein Modell aufgestellt, welches die Operationen formal beschreibt. Dieses erste Gene Assembly Modell wird im Folgenden als *universelles Modell* bezeichnet. In diesem Modell wurde eine MDS durch ein Paar (p, q) von Pointern beschrieben, wobei p den eingehenden und q den ausgehenden Pointer der MDS darstellt. Diese Art der Kennzeichnung wird als *MDS-Deskriptor* bezeichnet.

Wird eine der Operationen ld, hi und dlad auf ein Molekül M angewandt, so kann die Ausführung aufgrund der Struktur der Bereiche von M, welche von der Operation betroffen sind, sehr umständlich sein. Um eine Operation auszuführen, welche den Prozess der homologen Rekombination beinhaltet, sind umfangreiche Faltungen des Moleküls notwendig. Es stellt sich die Frage, ob es für eine realistischere Analyse des Gene Assembly Prozesses sinnvoll ist, die Operationen ld, hi und dlad so zu beschränken, dass die von einer Operation betroffenen Bereiche möglichst gering ausfallen [EHP$^+$03]. Die Einführung solcher *einfachen* Versionen von ld, hi und dlad wurde in [EPPR01b] vorgenommen. Mit diesen Operationen als Grundlage wurden zwei Gene Assembly Modelle aufgestellt, die beide den Modellnamen *simple gene assembly* nutzen. Um beide Modelle auseinander zu halten wurde in [LPR08] eine Namensaufteilung in *elementary model* und *simple model* vorgenommen. In dieser Arbeit verwenden wir entsprechende deutschsprachige Bezeichnungen. *Elementares Modell* statt elementary model und *einfaches Modell* statt simple model. Eingeführt wurde das einfache Modell in [LP06], das elementare Modell in [HPRR05, HPRR08]. Beiden Modelle ist gemein, dass der Gene Assembly Prozess als die Sortierung von vorzeichenbehafteten Permutationen betrachtet wird. Dafür ist die Darstellung eines Gens als Permutation notwendig.

Im *permutationsbasierten* Ansatz wird ein Gen durch die Sequenz seiner MDSs repräsentiert. Für ein aus n MDSs bestehendes Gen wird in dieser Notation das Alphabet $\Sigma_n = \{1, 2, \ldots, n\}$ verwendet. Dabei ist n eine beliebige aber feste natürliche Zahl. Jede MDS erhält demnach eine eindeutige Ziffer. Die Nummerierung der MDSs wird dabei so vorgenommen, dass die makronucleare Anordnung der MDSs das Wort $1\ 2 \ldots n$ bildet. Um zu kennzeichnen, dass eine MDS p, $p \in \Sigma_n$ in der mikronuclearen Anordnung invertiert vorliegt, wird das Element p mit dem entsprechenden Symbol $\overline{p} \in \overline{\Sigma}_n$ gekennzeichnet. Durch diese Schreibweise ist ein mikronucleares Gen als vorzeichenbehaftete Permutation über $\Sigma_n^{\pm}$ und ein makronucleares Gen als sortierte vorzeichenbehaftete Permutation formalisiert. Mit dieser Vorgabe kann der makronucleare Abstand zweier MDSs zueinander durch einfache Subtraktion mit Betragsbildung

ermittelt werden. So beträgt der Abstand der MDSs p und q, genau $|p - q|$.

Im Folgenden werden wir für jedes der drei Gene Assembly Modelle die formale Definition der jeweiligen Operationen angeben. Außerdem gehen wir näher darauf ein, unter welchen Umständen eine Operation als „einfach“ zu bezeichnen ist. Im sich anschließenden Kapitel 5 nehmen wir einen Vergleich der Modelle hinsichtlich ausgewählter Modelleigenschaften vor.

4.1 Universelles Modell

Im universellen Modell (engl.: *general model*) unterliegen die drei Operationen ld, hi und dlad keinerlei Beschränkungen, so dass jede MDS, welche an einer Operation beteiligt ist, an einer beliebigen Stelle auf dem zugehörigen Molekül platziert werden kann. Um den angesprochenen Vergleich der drei Modelle in Kapitel 5 vorzunehmen, ist es notwendig die Operationen des universellen Modells anstelle der üblichen Beschreibung von MDS-Deskriptoren, stattdessen in einer permutationsbasierten Darstellung zu definieren.

Definition 4.1 (Operationen des universellen Modells [LPR08]) Für $1 \leq p < n$ ist hi_p wie folgt definiert:

$$\begin{aligned}
\mathrm{hi}_p(xpy(\overline{p+1})z) &= xp(p+1)\overline{y}z,\\
\mathrm{hi}_p(x\overline{p}y(p+1)z) &= x\overline{y}p(p+1)z,\\
\mathrm{hi}_p(x(p+1)y\overline{p}z) &= x\overline{y}(\overline{p+1})\overline{p}z,\\
\mathrm{hi}_p(x(\overline{p+1})ypz) &= x(\overline{p+1})\overline{py}z,
\end{aligned}$$

wobei $x, y, z \in \Sigma_n^{\maltese}$. Mit $\mathrm{Hi} = \{\mathrm{hi}_i \,|1 \leq i < n\}$ wird die Menge aller universellen Hairpin-Operationen bezeichnet.

Für $1 \leq p, q < n$ mit $|p - q| > 1$ ist $\mathrm{dlad}_{p,q}$ wie folgt definiert:

$$\begin{aligned}
\mathrm{dlad}_{p,q}(xp''uq''vp'wq'z) &= xwq'q''vp'p''uz,\\
\mathrm{dlad}_{p,q}(xp''uq'vp'wq''z) &= xwvp'p''uq'q''z,\\
\mathrm{dlad}_{p,q}(xp'uq''vp''wq'z) &= xp'p''wq'q''vuz,\\
\mathrm{dlad}_{p,q}(xp'uq'vp''wq''z) &= xp'p''wvuq'q''z,
\end{aligned}$$

wobei $p' = p, p'' = p{+}1$ oder $p' = (\overline{p+1}), p'' = \overline{p}$ und $q' = q, q'' = q{+}1$ oder $q' = (\overline{p+1}), q'' = \overline{p}$ und $x, u, v, w, z \in \Sigma_n^{\maltese}$. In allen diesen Fällen bezeichnen wir $\mathrm{dlad}_{p,q} = \mathrm{dlad}_{q,p}$.

Für $1 \leq p < n$ definieren wir $\mathrm{dlad}_{p-1,p}$ und $\mathrm{dlad}_{p,p-1}$ wie folgt:

$$\begin{aligned}
\mathrm{dlad}_{p-1,p}(xp'''up''wp'z) &= xwp'p''p'''uz,\\
\mathrm{dlad}_{p-1,p}(xp''vp'wp'''z) &= xwvp'p''p'''z,\\
\mathrm{dlad}_{p-1,p}(xp'up'''vp''z) &= xp'p''p'''vuz,
\end{aligned}$$

wobei $p' = p - 1, p'' = p, p''' = p + 1$ oder $p''' = (\overline{p+1}), p'' = p, p' = (\overline{p-1})$ und $x, u, v, w, z \in \Sigma_n^{\maltese}$. Die Menge aller universellen Doubleloop-Operationen wird mit $\mathrm{Dlad} = \{\mathrm{dlad}_{i,j} \,|1 \leq i, j < n, i \neq j\}$ bezeichnet.

Für permutationsbasierte Modelle ist es charakteristisch, dass die ld-Operation nicht ausdrücklich modelliert wird. Stattdessen wird die Annahme getroffen, dass zwei in der makronuclearen Zielordnung benachbarte MDSs zu einem unbestimmten Zeitpunkt und unabhängig von anderen Operationen mittels der entsprechenden ld-Operation verbunden werden.

BEISPIEL 4.2 Gegeben sei die vorzeichenbehaftete Permutation π über Σ_n mit $n = 7$ und $\pi = 6\ \overline{3}\ 7\ \overline{4}\ 5\ 1\ 2$. Eine Sortierung mit Operationen des universellen Modells ist wie folgt möglich:

$$\begin{array}{ll} \mathrm{hi}_2(\pi) = \pi_1 = 6\ \overline{3}\ \overline{2}\ \overline{1}\ \overline{5}\ 4\ \overline{7} & \mathrm{hi}_6(\pi_1) = \pi_2 = 6\ 7\ \overline{4}\ 5\ 1\ 2\ 3 \\ \mathrm{hi}_4(\pi_2) = \pi_3 = 6\ 7\ 4\ 5\ 1\ 2\ 3 & \mathrm{dlad}_{5,3}(\pi_3) = 1\ 2\ 3\ 4\ 5\ 6\ 7 \end{array}$$

4.2 Elementares Modell

Damit die Operationen des elementaren und des einfachen Modells möglichst kleine Bereiche eines Gens beeinflussen, unterliegen sie der Restriktion, dass ausschließlich mikronucleare und somit keine zusammengesetzten MDSs an den molekularen Operationen ld, hi und dlad beteiligt sein dürfen. Demnach lassen sich zwei oder mehrere mikronucleare MDSs, nachdem sie zusammengebracht wurden, nicht mehr auf dem zugehörigen Molekül bewegen. Bevor wir die Definition für elementare Operationen angeben, werden wir zunächst diese Einschränkung erläutern und veranschaulichen. Zur Erinnerung sind in Abbildung 4.1 noch einmal die drei Operationen ld, hi und dlad abgebildet.

Als Resultat der loop-Operation ergibt sich neben dem linearen noch ein zirkuläres Molekül. Da wir intramolekulare Modelle betrachten, kann dieses Molekül im weiteren Verlauf nicht mit dem linearen rekombinieren. Das zirkuläre Molekül ist ein Abfallprodukt, welches nur IESs enthalten darf. Eine weitere Einschränkung von ld ist daher nicht möglich und die ld-Operation somit immer „einfach“.

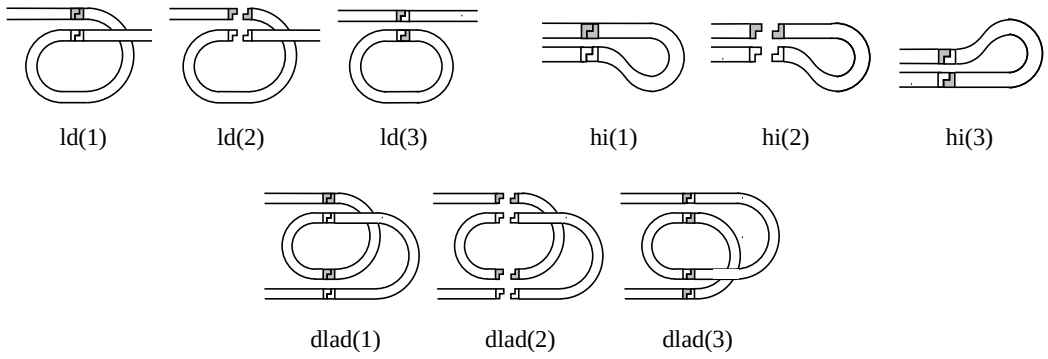


Abbildung 4.1: Die molekularen Operationen ld, hi und dlad im Überblick.

Bei der Anwendung von hi und dlad können die betroffenen Sequenzabschnitte wesentlich länger sein. Als Beispiel sehen wir uns noch einmal das Actin-Protein Gen aus Sterkiella Nova an (Abbildung 4.2).

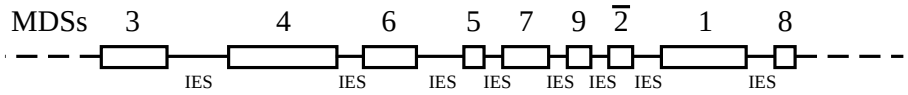


Abbildung 4.2: Struktur des Actin-Protein kodierenden Gen in Sterkiella Nova.

Der Pointer am Beginn der MDS M_3 tritt ein zweites Mal invertiert am Ende der MDS M_2 auf. Bei der Aneinanderlagerung der Pointer zur Ausführung einer hi-Operation würde sich eine Haarnadel bestehend aus den MDSs M_3, M_4, M_6, M_5, M_7 und M_9 ergeben.

DEFINITION 4.3 (VEREINFACHTE hi-OPERATION [HPR06]) Die Anwendung der Operation hi für einen Pointer p wird *vereinfacht* genannt, wenn die Sequenz, welche die zwei Vorkommen von p trennt, aus genau einer IES und genau einer MDS besteht.

DEFINITION 4.4 (VEREINFACHTE dlad-OPERATION [HPR06]) Die Anwendung der Operation dlad für zwei Pointer p und q wird als vereinfacht bezeichnet, sofern die Sequenz, welche von dem ersten Auftreten von p und q umschlossen ist und die Sequenz, welche von dem zweiten Vorkommen von p und q umspannt wird, entweder aus genau einer MDS oder genau einer IES besteht.

Die Abbildungen 4.3 und 4.4 zeigen die jeweiligen Genstrukturen für die eine einfache hi- beziehungsweise eine einfache dlad-Operation möglich ist.

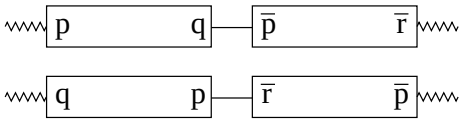


Abbildung 4.3: Genstrukturen, welche die Anwendung der vereinfachten hi-Operation für den Pointer p erlauben. Eine MDS ist durch ein Rechteck gekennzeichnet. Eine gerade Linie steht für genau eine IES. Eine Zickzack Linie steht für eine beliebig lange DNA-Sequenz aus MDSs und IESs. Nach [EHP+03]

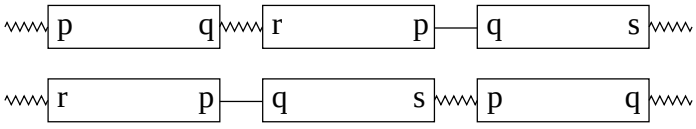


Abbildung 4.4: Genstrukturen, welche die Anwendung der vereinfachten dlad-Operation für die Pointer p und q erlauben. Eine MDS ist durch ein Rechteck gekennzeichnet. Eine gerade Linie steht für genau eine IES. Eine Zickzack Linie steht für eine beliebig lange DNA-Sequenz aus MDSs und IESs. Nach [EHP+03]

Beispiel 4.5 Gegeben sei die in Sterkiella Nova für das Actin-Protein kodierende Folge von MDSs (Vgl. Abbildung 4.2): $M_3M_4M_6M_5M_7M_9\overline{M_2}M_1M_8$. Notiert als vorzeichenbehaftete Permutation π ergibt sich $\pi = 346579\overline{2}18$.

(i) Die Anwendung von hi für den Pointer am Beginn von M_3 und am Ende von M_2 ist nicht einfach. Die Sequenz zwischen den beiden Vorkommen der Pointer besteht aus sechs MDSs: $M_3M_4M_6M_5M_7M_9$.

(ii) Mit p_6 wird der Pointer am Beginn von M_6 und am Ende von M_5 bezeichnet. Mit p_7 der Pointer am Ende von M_6 und am Beginn von M_7. Die Anwendung von dlad_{p_6,p_7} ist einfach. In der Tat sind die ersten Vorkommen von p_6 und p_7 nur durch M_6 voneinander getrennt, während die zweiten Vorkommen nur von der IES zwischen M_5 und M_7 unterbrochen sind.

Es folgt die Definition der Operationen des elementaren Modells [HPR06]:

Definition 4.6 (Operationen des elementaren Modells [HPR06]) Für $p \geq 1$ ist die Operation eh_p wie folgt definiert:

$$\begin{aligned}
\text{eh}_p(xp(\overline{p+1})z) &= xp(p+1)z,\\
\text{eh}_p(x\overline{p}(p+1)z) &= xp(p+1)z,\\
\text{eh}_p(x(p+1)\overline{p}z) &= x(\overline{p+1})\overline{p}z,\\
\text{eh}_p(x(\overline{p+1})pz) &= x(\overline{p+1})\overline{p}z,
\end{aligned}$$

wobei $x, z \in \Sigma_n^{\pm}$. Die Menge aller elementaren Hairpin-Operationen wird mit $\text{Eh} = \{\text{eh}_p \,|1 \leq p \leq n\}$ bezeichnet.

Für $p \geq 1$ ist die Operation ed_p wie folgt definiert:

$$\begin{aligned}
\text{ed}_p(xpy(p-1)(p+1)z) &= xy(p-1)p(p+1)z,\\
\text{ed}_p(x(p-1)(p+1)ypz) &= x(p-1)p(p+1)yz,\\
\text{ed}_p(x\overline{p}y(\overline{p+1})(\overline{p-1})z) &= xy(\overline{p+1})\overline{p}(\overline{p-1})z,\\
\text{ed}_p(x(\overline{p+1})(\overline{p-1})y\overline{p}z) &= x(\overline{p+1})\overline{p}(\overline{p-1})yz,
\end{aligned}$$

wobei $x, y, z \in \Sigma_n^{\pm}$. Die Menge aller elementaren Doubleloop-Operationen wird mit $\text{Ed} = \{\text{ed}_p \,|1 < p < n\}$ bezeichnet.

Beispiel 4.7 Gegeben sei die vorzeichenbehaftete Permutation $\pi = 153\overline{4}62$. Eine Sortierung mit Operationen des elementaren Modells ist wie folgt möglich.

$$\begin{aligned}
&\text{eh}_3 = 153462, && \text{ed}_2 \circ \text{ed}_5 \circ \text{eh}_3 = 123456,\\
&\text{ed}_5 \circ \text{eh}_3 = 134562.
\end{aligned}$$

4.3 Einfaches Modell

Das einfache Modell ist eine Weiterentwicklung des elementaren Modells. Es wurde in [LP06] aufgestellt. Der Unterschied zum elementaren Modell liegt in einer abweichenden Auslegung

der Bedeutung von „genau eine MDS“ in den Definitionen der vereinfachten Operationen aus 4.3 und 4.4. In diesem Zusammenhang spielt der Ausführungszeitpunkt der Operation ld die entscheidende Rolle. In bisherigen wissenschaftlichen Arbeiten findet sich die Aussage, dass mögliche ld-Operationen zu einem unbestimmten Zeitpunkt im Verlauf der Sortierung erfolgen. In dieser Arbeit sei der Ausführung einer ld-Operation im einfachen Modell eine höhere Priorität zugeordnet, als der Anwendung von hairpin oder doubleloop. Mit anderen Worten: Eine ld-Operation wird in dem Moment angewendet, sobald sie möglich ist.

Mit diesem Hintergrundgedanken sind die folgenden Schlüsse aus [HPR06] besser nachvollziehbar: Sofern zwei MDSs M_i und M_{i+1} (getrennt durch eine IES) nebeneinander liegen, werden diese durch ld_{i+1} verbunden, wobei der nicht kodierende Bereich zwischen ihnen entfernt wird (vgl.: Abbildung 2.8). Im Resultat erhalten wir einen rein kodierenden Bereich aus genau diesen zwei MDSs. Die angesprochene abweichende Auslegung zum elementaren Modell besteht darin, dass dieser Bereich im einfachen Modell als „genau eine MDS“, wie in Definition 4.3 und 4.4 verlangt, betrachtet wird. Wie zu sehen ist, bleiben wir durch Einführung der Priorisierung von ld den Definitionen gerecht.
Zusammenfassend lässt sich die Aussage treffen, dass das einfache Modell erlaubt, sowohl mikronucleare MDSs, als auch solche, welche bereits mit anderen verbunden wurden, durch die Operationen ld, hi und dlad zu bewegen.
Die sich daraus ergebenden Änderungen der Formalisierung zeigen sich in der Definition der Operationen des einfachen Modells.

Definition 4.8 (Operationen des einfachen Modells. Nach [LP06]) Für $1 \leq p < n$ ist sh_p wie folgt definiert:

$$\begin{aligned}
\mathrm{sh}_{(p,p+i)}(xp\dots(p+i)(\overline{p+k})\dots(\overline{p+i+1})y) &= xp\dots(p+i)(p+i+1)\dots(p+k)y,\\
\mathrm{sh}_{(p,p+i)}(x(\overline{p+i})\dots\overline{p}(p+i+1)\dots(p+k)y) &= xp\dots(p+i)(p+i+1)\dots(p+k)y,\\
\mathrm{sh}_{(p,p+i)}(x(p+i+1)\dots(p+k)(\overline{p+i})\dots\overline{p}y) &= x(\overline{p+k})\dots(\overline{p+i+1})(\overline{p+i})\dots\overline{p}y,\\
\mathrm{sh}_{(p,p+i)}(x(\overline{p+k})\dots(\overline{p+i+1})p\dots(p+i)y) &= x(\overline{p+k})\dots(\overline{p+i+1})(\overline{p+i})\dots\overline{p}y,
\end{aligned}$$

wobei $k > i \geq 0$ und $x, y \in \Sigma_n^{\maltese}$. Die Menge aller einfachen Hairpin-Operationen wird mit $\mathrm{Sh} = \{\mathrm{sh}_{(p,p+i)} \,|1 \leq i < n\}$ bezeichnet.

Für $2 \leq p < n$ ist sd_p wie folgt definiert:

$$\begin{aligned}
\mathrm{sd}_{(p,p+i)}(xp\dots(p+i)y(p-1)(p+i+1)z) &= xy(p-1)p\dots(p+i)(p+i+1)z,\\
\mathrm{sd}_{(p,p+i)}(x(p-1)(p+i+1)yp\dots(p+i)z) &= x(p-1)p\dots(p+i)(p+i+1)yz,\\
\mathrm{sd}_{(\overline{p+i},\overline{p})}(x(\overline{p+i+1})(\overline{p-1})y(\overline{p+i})\dots\overline{p}z) &= x(\overline{p+i+1})(\overline{p+i})\dots\overline{p}(\overline{p-1})yz,\\
\mathrm{sd}_{(\overline{p+i},\overline{p})}(x(\overline{p+i})\dots\overline{p}y(\overline{p+i+1})(\overline{p-1})z) &= xy(\overline{p+i+1})(\overline{p+i})\dots\overline{p}(\overline{p-1})z,
\end{aligned}$$

wobei $i \geq 0$ und $x, y, z \in \Sigma_n^{\maltese}$. Die Menge aller einfachen Doubleloop-Operationen wird mit $\mathrm{Sd} = \{\mathrm{sd}_{(p,p+i)}, \mathrm{sd}_{(\overline{p+i},\overline{p})} \mid 2 \leq p < n, 0 \leq i \leq (n-1)-p\}$ bezeichnet.

Ein offensichtlicher Vorteil dieser Erweiterung besteht darin, dass das einfache Modell in vielen Fällen Permutationen erfolgreich sortieren kann, an denen das elementare Modell scheitert

(vgl. Beispiel 4.9). Weitere Vorteile gegenüber dem elementaren Modell zeigen sich im Vergleich der drei Modelle, welchen wir im folgenden Kapitel aufstellen werden.

BEISPIEL 4.9 Gegeben sei die vorzeichenbehaftete Permutation $\pi = 213\overline{6}745\overline{5}$. Wir betrachten die Sortierung für das elementare Modell im Vergleich mit dem einfachen Modell.

elementares Modell	**einfaches Modell**
$\mathrm{eh}_6(\pi) = 2136745\overline{5}$	$\mathrm{sh}_{(6,6)}(\pi) = 2136745\overline{5}$
$\mathrm{ed}_2 \circ \mathrm{eh}_6(\pi) = 123674\overline{5}$	$\mathrm{sd}_{(2,2)} \circ \mathrm{sh}_{(6,6)}(\pi) = 123674\overline{5}$
$\mathrm{eh}_4 \circ \mathrm{ed}_2 \circ \mathrm{eh}_6(\pi) = 1236745$	$\mathrm{sh}_{(4,4)} \circ \mathrm{sd}_{(2,2)} \circ \mathrm{sh}_{(6,6)}(\pi) = 1236745$
blockiert	$\mathrm{sd}_{(4,5)} \circ \mathrm{sh}_{(4,4)} \circ \mathrm{sd}_{(2,2)} \circ \mathrm{sh}_{(6,6)}(\pi) = 1234567$

4.4 Sortierende Strategien

Eine Komposition von Operationen $\Phi = \phi_k \circ \phi_{k-1} \circ \ldots \phi_2 \circ \phi_1$, wobei alle Operationen ϕ_i mit $1 \leq i \leq k$ entweder aus der Menge $\mathrm{Hi} \cup \mathrm{Dlad}$, $\mathrm{Sh} \cup \mathrm{Sd}$, oder $\mathrm{Eh} \cup \mathrm{Ed}$ stammen, wird *Strategie* genannt. Eine Komposition von Operationen $\Phi = \phi_k \circ \phi_{k-1} \circ \ldots \phi_2 \circ \phi_1$ wird *sortierende Strategie* für π genannt, wenn $\Phi(\pi)$ eine sortierte Permutation ist. Wenn $\phi_i \in (\mathrm{Hi} \cup \mathrm{Dlad})$, für alle $1 \leq i \leq k$, dann ist Φ eine *universell sortierende Strategie*. Wenn $\phi_i \in (\mathrm{Sh} \cup \mathrm{Sd})$, für alle $1 \leq i \leq k$, dann ist Φ eine *einfach sortierende Strategie*. Wenn $\phi_i \in (\mathrm{Eh} \cup \mathrm{Ed})$, für alle $1 \leq i \leq k$, dann ist Φ eine *elementar sortierende Strategie*.

Eine unsortierte vorzeichenbehaftete Permutation π ist *blockiert*, wenn keine einfache beziehungsweise elementare Operation auf π anwendbar ist. Eine Operationsfolge Φ wird *nicht sortierend* genannt, wenn $\Phi(\pi)$ eine blockierte Permutation ist. Wenn keine sortierende Strategie für π existiert wird π *nicht zu sortierende Permutation* genannt. Zu einer Strategie Φ notieren wir mit $|\Phi|$ die Anzahl der Operationen einer Strategie.

Bemerkung: Im weiteren Verlauf dieser Arbeit sprechen wir von einer Strategie, wenn deren Anwendung entweder zu einer sortierten oder blockierten Permutation führt. Andernfalls sprechen wir von einer Teilstrategie.

Kapitel 5

Vergleich der drei Modelle

Wenngleich die Unterschiede in den Operationsdefinitionen der Gene Assembly Modelle nur gering erscheinen mögen, insbesondere zwischen dem elementaren und dem einfachen Modell, so bewirken diese jedoch starke Eigentümlichkeiten der Modelle hinsichtlich ausgewählter Eigenschaften. In diesem Kapitel werden wir Resultate aus bisher geleisteten Modelluntersuchen zusammenfassen und dabei die drei Modelle untereinander vergleichen. Die Auswahl der betrachteten Modelleigenschaften ist [LPR08] entnommen. Die folgende Tabelle zeigt um welche Merkmale es sich konkret handelt und erläutert knapp deren Bedeutung.

Eigenschaft	**Bedeutung**
Vollständigkeit	Jedes Genmuster (jede beliebige vorzeichenbehafte Permutation π über Σ_n) ist sortierbar.
Konfluenz	(a) Verschiedene sortierende Strategien zu einer vorzeichenbehafteten Permutation führen zur gleichen Permutation. (b) Für kein Genmuster gibt es sowohl sortierende als auch nicht sortierende Strategien. (c) Verschiedene sortierende Strategien zu einer vorzeichenbehafteten Permutation führen zur gleichen Struktur der Permutation.
Konstante sequentielle Komplexität	Jede sortierende Strategie zu einer gegebenen vorzeichenbehafteten Permutation benötigt die gleiche Anzahl intramolekularer Operationen.
Entscheidbarkeit des Sortierproblems	Ist es für eine gegebene vorzeichenbehaftete Permutation effektiv möglich zu entscheiden, ob diese sortiert werden kann.
Charakterisierung von sortierbaren Permutationen	Ausgehend von charakteristischen Merkmalen einer gegebenen vorzeichenbehafteten Permutation ist es möglich zu schließen, dass diese sortiert werden kann oder nicht.

Die Tabelle gibt gleichzeitig die Reihenfolge vor, in welcher wir die Eigenschaften innerhalb der Modelle überprüfen. Für jedes Merkmal betrachten wir zunächst das universelle Gene Assembly Modell, gefolgt vom elementaren und dem einfachen Modell.

5.1 Vollständigkeit

Wir geben zunächst die Definition für die Vollständigkeit eines Gene Assembly Modells an. Es sei darauf hingewiesen, dass die Bezeichnung „beliebiges Genmuster“ ausdrücklich auch solche Genmuster umfasst, welche real nicht existieren.

Definition 5.1 (Vollständigkeit eines Gene Assembly Modells) Ein Gene Assembly Modell ist vollständig, wenn für jedes beliebige Genmuster, also für jede beliebige vorzeichenbehaftete Permutation π über einem Alphabet Σ_n, eine sortierende Strategie existiert.

In [EPPR01b] wurde gezeigt, dass das universelle Modell vollständig ist. Das Resultat wurde nicht für die permutationsbasierte Definition des Modells bewiesen, sondern mit der formalen Darstellung durch MDS-Deskriptoren. Um den Beweis für die Darstellung vorzeichenbehafteter Permutationen zu führen, gibt es die Möglichkeit zu zeigen, dass die Menge vorzeichenbehafteter Permutationen und die Menge von MDS-Deskriptoren zueinander korrespondieren. Wir wollen hier jedoch den direkten Beweis liefern, indem der Beweis für MDS-Deskriptoren aus [EPPR01b] Schritt für Schritt im Fall vorzeichenbehafteter Permutationen nachgezeichnet wird. Für diesen Fall ist der entscheidende Punkt, dass für jedes $\phi \in \mathrm{Hi} \cup \mathrm{Dlad}$ und jede vorzeichenbehaftete Permutation π die Anzahl der sortierten Blöcke in $\phi(\pi)$ kleiner als in π ist. Eine vorzeichenbehaftete Permutation π ist genau dann sortiert, wenn $S(\pi) \leq 2$ gilt und π einheitliche Vorzeichen trägt.

Satz 5.2 Jede beliebige vorzeichenbehaftete Permutation ist sortierbar unter $\mathrm{Hi} \cup \mathrm{Dlad}$.

Beweis Zunächst zeigen wir, dass immer eine der beiden Operationen hi oder dlad auf jede beliebige vorzeichenbehaftete Permutation π der Länge n angewendet werden kann, wobei n nicht festgelegt aber endlich ist. Sind in π Elemente mit unterschiedlichen Vorzeichen enthalten, so kann hi angewendet werden. Damit hi nicht ausführbar wird, stellen wir uns vor, die Elemente von π wären einheitlich nicht als invertiert gekennzeichnet. Außerdem enthalte π lediglich sortierte Blöcke der Länge 1. Daraus ergibt sich, dass auf kein Element p direkt das Element $p+1$ folgt. Sei p ein Element in π $(1 \leq p < n)$, so dass die Anzahl von Elementen im Intervall zu $p+1$ minimal ist (kein anderes Element in π hat weniger Elemente in seinem Intervall).

Fall 1: mindestens ein Element:
Seien die Elemente q und $q+1$ mit p und $p+1$ verschränkt $(1 \leq p, q < n)$. Das heißt, dass entweder das Element q oder $q+1$ im Intervall zwischen p und $p+1$ liegt. Entweder q oder $q+1$ muss außerhalb des Intervalls p und $p+1$ liegen, da sonst dessen Minimalität verletzt ist.

(i) $|p-q| > 1$:
Somit ist $\mathrm{dlad}_{p,q}$ auf π anwendbar mit einem der Muster: $(xp+1uqvpwq+1z), (xp+1uq+1vpwqz), (xpuqvp+1wq+1z), (xpuq+1vp+1wqz)$, wobei x, u, v, w, z Wörter über Π_n sind.

(ii) $|p-q| = 1$:
Demnach handelt es sich bei q um das Element $p-1$. Somit ist $\mathrm{dlad}_{p-1,p}$ auf π anwendbar mit dem Muster: $(xpvp-1wp+1z)$, wobei x, u, v, w, z Wörter über Π_n sind.

Fall 2: kein Element:
Dies bedeutet, dass das Element $p+1$ direkt vor dem Element p liegt. Somit ist $\mathrm{dlad}_{p-1,p}$ auf π anwendbar mit dem Muster: $(xp+1upwp-1z), (xp-1up+1vpz)$, wobei x, u, v, w, z Wörter über Π_n sind.

Jede der Operationen verringert die Anzahl sortierter Blöcke und nach jeder Operationsanwendung erhalten wir erneut eine vorzeichenbehaftete Permutation. Wir wissen, dass eine vorzeichenbehaftete Permutation π sortiert ist, genau dann wenn $S(\pi) \leq 2$ gilt und π einheitliche Vorzeichen trägt. □

Das elementare und das einfache Modell sind im Gegensatz zum universellen Gene Assembly Modell nicht vollständig, wie das folgende Beispiel zeigt.

Beispiel 5.3 Gegeben sei die Permutation $\pi = 321$. Aufgrund der einheitlichen Vorzeichen ist leicht zu sehen, dass weder eine Hairpin-Operation sh aus dem einfachen Modell noch eine Operation eh aus dem elementaren Modell angewendet werden kann. Auch gibt es keine Möglichkeit die Doubleloop-Operationen sd und ed zu verwenden. Somit kann die Permutation unter Anwendung dieser beiden Modelle nicht sortiert werden. Dies gelingt nur mit Hilfe des universellen Modells: $\mathrm{dlad}_{1,2}(\pi) = 123$.

Eine Datenbank bekannter mikro- und makronuclearer Genmuster in Ciliaten bietet [CCL05]. Aufgrund der Vollständigkeit des universellen Modells ist es offensichtlich, dass für jede Sequenz dieser Datenbank eine sortierende Strategie im universellen Modell existiert. Wie sich herausstellt, ist das elementare Modell nicht fähig jedes Genmuster der Datenbank zu sortieren [CCL05]. Das jedoch gelingt mit dem einfachen Modell, wodurch das Modell als biologisch Vollständig betrachtet werden kann. Selbstverständlich ist bis zur Vervollständigung der Datenbank nicht auszuschließen, dass ein Genmuster in Ciliaten existiert, welches nicht mit den Operationen des einfachen Modells sortiert werden kann.

Beispiel 5.4 Das Actin-Protein in Sterkiella Novakodierende wird durch die vorzeichenbehaftete Permutation $\pi = 346579\overline{2}18$ repräsentiert (vgl. Abbildung 4.2). Mit Hilfe des elementaren Modells lässt sich die Invertierung von MDS 2 nicht auflösen und so kann es im elementaren Modell keine sortierende Strategie zu π geben. Mit der einfach sortierenden Strategie

$$\mathrm{sh}\,(1,1) \circ \mathrm{sh}_{(2,2)} \circ \mathrm{sd}_{(8,8)} \circ \mathrm{sd}\,(5,5)(\pi) = \overline{9}\ \overline{8}\ \overline{7}\ \overline{6}\ \overline{5}\ \overline{4}\ \overline{3}\ \overline{2}\ \overline{1}$$

ist π im einfachen Modell sortierbar.

5.2 Konfluenz

Mit einem mikronuclearen Genmuster als Eingabe liefert jede Operation in Gene Assembly ein Genmuster als Ausgabe. Zusätzlich lassen sich solange Operationen anwenden, bis das Genmuster sortiert oder keine Operation mehr anwendbar ist. Es ist somit leicht zu erkennen, dass Gene Assembly ein Reduktionssystem ist. Dieses lässt sich selbstverständlich auf konfluentes Verhalten hin untersuchen.
Konfluenz ist die Eigenschaft eines Reduktionssystems, dass zwei Terme, welche ausgehend von einem Ausgangsterm durch unterschiedliche Ableitungen erhalten wurden, durch weitere Ableitung stets wieder zu einem Term zusammengeführt werden können.

Definition 5.5 (Konfluenz eines Reduktionssystems) Ein Reduktionssystem R ist konfluent, wenn für jede beliebige Eingabe x gilt,

$$\forall \Phi_1, \Psi_1 \text{ mit } \Phi_1(x) = u, \Psi_2(x) = v \; \exists \Phi_2, \Psi_2 \text{ mit } \Phi_2(u) = \Psi_2(v) = z.$$

In [LPR08] wurde unter drei Gesichtspunkten konfluentes Verhalten verschiedener Strategien analysiert. Dabei wurde als Endterm zum einen das resultierende Genmuster, zum anderen der Sortiererfolg und schließlich die resultierende Genstruktur untersucht.

5.2.1 Konfluenz bezüglich resultierenden Genmuster

Definition 5.6 (Konfluenz bezüglich resultierenden Genmuster. Nach [LPR08]) Ein Gene Assembly Modell ist bezüglich resultierenden Genmuster konfluent, wenn für jede beliebige vorzeichenbehaftete Permutation π gilt:

$$\forall \Phi_1, \Psi_1 \text{ mit } \Phi_1(\pi) = u, \Psi_1(\pi) = v \; \exists \Phi_2, \Psi_2 \text{ mit } \Phi_2(u) = \Psi_2(v) = z,$$

wobei $\Phi_2 \circ \Phi_1$ und $\Psi_2 \circ \Psi_1$ Strategien zu π sind. [LPR08].

Ein Gene Assembly Modell ist unter diesem Ansatz also konfluent, wenn jede Strategie zu einer gegebenen vorzeichenbehafteten Permutation im selben Genmuster resultiert.
Anhand eines Beispiels ist zu sehen, dass keines der Modelle unter dieser Definition konfluent ist.

Beispiel 5.7 Gegeben sei die Permutation $\pi = 2413$. Mit $\mathrm{dlad}_{2,1}(\pi) = \mathrm{sd}_{(2,2)}(\pi) = \mathrm{ed}_2(\pi) = 4123$ und $\mathrm{dlad}_{2,3} = sd_{3,3)}(\pi) = \mathrm{ed}_3(\pi) = 2341$ erhalten wir zwei unterschiedliche Resultate bei gleicher Eingabe.

5.2.2 Konfluenz bezüglich Erfolg

Das vorangegangene Beispiel zeigt, dass alle drei Modelle nicht deterministisch sind in dem Sinn, dass verschiedene Strategien zu unterschiedlichen Resultaten führen können. Dies führt zu der Frage, ob zu einer Permutation anwendbare Strategien in einer Resultatsmenge enden, die sowohl sortierte als auch geblockte Permutationen enthält.

Definition 5.8 (Konfluenz bezüglich Erfolg. Nach [LPR08]) Ein Gene Assembly Modell ist bezüglich Erfolg konfluent, wenn für jede beliebige vorzeichenbehaftete Permutation π gilt:

$$\forall \Phi, \Psi \text{ mit } \Phi(\pi) = u, \Psi(\pi) = v \text{ und } u, v \text{ sind sortiert oder } u, v \text{ sind geblockt},$$

wobei Φ und Ψ Strategien zu π sind. [LPR08].

Ein Gene Assembly Modell ist demnach konfluent, wenn es keine vorzeichenbehaftete Permutation gibt, welche sowohl sortierende als auch nicht sortierende Strategien besitzt.
Da es im universellen Modell keine nicht sortierenden Strategien gibt (Satz 5.2), ist das Modell bei dieser Betrachtung konfluent. Das elementare Modell hingegen ist unter dieser Definition nicht konfluent. Einen Widerspruch für konfluentes Verhalten zeigt Beispiel 5.9.

Beispiel 5.9 Gegeben sei die Permutation $\pi = 24135$. Eine geblockte Permutation erhalten wir mit $ed_3(\pi) = 23415$, während $ed_2 \circ ed_4(\pi) = 12345$ ein erfolgreiches Resultat liefert.

In [LP06] wurde gezeigt, dass das einfache Modell konfluent bezüglich Erfolg ist. Der entscheidende Aspekt ist dabei, dass für zwei auf eine Permutation π angewandte Operationen ϕ und ψ aus $\mathrm{Sh} \cup \mathrm{Sd}$, entweder $\sigma(\phi(\pi)) = \sigma(\psi(\pi))$ oder für die Hintereinanderausführung $\sigma(\phi \circ \psi(\pi)) = \sigma(\psi \circ \phi(\pi))$ gilt. Dadurch kann es im einfachen Modell zu einer Permutation niemals sortierende und nicht sortierende Strategien geben. Dieser *schwache Determinismus* [LP06] ist gleichfalls die Begründung dafür, dass das einfache Modell auch dem dritten Konfluenzansatz standhält, welcher Konfluenz bezüglich der Struktur verlangt.

5.2.3 Konfluenz bezüglich Struktur

In Abschnitt 3.3 wurde die Struktur von vorzeichenbehafteten Permutationen vorgestellt. Diese ist im dritten Konfluenzansatz von Interesse.

Definition 5.10 (Konfluenz bezüglich Struktur. Nach [LPR08]) Ein Gene Assembly Modell ist bezüglich Struktur konfluent, wenn für jede beliebige vorzeichenbehaftete Permutation π gilt:

$\forall \Phi_1, \Psi_1$ mit $\Phi_1(\pi) = u_1, \Psi_1(\pi) = v_1$ $\exists \Phi_2, \Psi_2$ mit $\Phi_2(\pi) = u_2, \Psi_2(\pi) = v_2$ und $\sigma(u_2) = \sigma(v_2)$,

wobei $\Phi_2 \circ \Phi_1$ und $\Psi_2 \circ \Psi_1$ Strategien zu π sind. [LPR08].

Nach dieser Definition ist ein Gene Assembly Modell konfluent, wenn jede Strategie zu einer Permutation zur selben Struktur führt.
Im universellen Modell gibt es ausschließlich erfolgreiche Strategien. Daher sind die möglichen Strukturen einer vorzeichenbehafteten Permutation π entweder $\sigma(\pi) = 1$ (linear) oder $\sigma(\pi) = \bar{1}$ (zirkulär). In [EPPR01a, Pet06] wurde bewiesen, dass entweder alle sortierenden Strategien für eine Permutation zu einem linearen Molekül führen oder sie liefern ein zirkuläres Molekül. Dies gilt ebenso für die erfolgreichen Sortierungen im einfachen und elementaren Modell. Während das universelle Modell im Struktursinn also konfluent ist, müssen wir für die zwei anderen Modelle noch die unerfolgreichen Strategien betrachten.
Das elementare Modell kann insofern dem Konfluenansatz nicht standhalten, da es für Permutationen sowohl sortierende als auch nicht sortierende Strategien gibt. Für das einfache Modell haben wir mit dem schwachen Determinismus bereits den Hauptaspekt genannt, um zu zeigen, dass das Modell bezüglich der Struktur konfluent ist. Der ausführliche Beweis findet sich in [LP06]. Das folgende Beispiel zeigt das konfluente Verhalten im einfachen Modell.

Beispiel 5.11 Gegeben sei die sortierbare Permutation $\pi_1 = \overline{2}\ 3\ 5\ \overline{6}\ 7\ 1\ 4$. Es gibt verschiedene Strategien, welche unterschiedliche Resultate erzeugen, aber in ihrer Struktur übereinstimmen.

$$\begin{aligned}
\pi_{1_1} &= \mathrm{sh}_{(6,6)} \circ \mathrm{sd}_{(2,3)} \circ \mathrm{sh}_{(2,2)}(\pi_1) = 5\ 6\ 7\ 1\ 2\ 3\ 4 \quad \sigma(\pi_{1_1}) = 2\ 1 \\
\pi_{1_2} &= \mathrm{sh}_{(2,5)} \circ \mathrm{sh}_{(2,2)} \circ \mathrm{sd}_{(4,4)}(\pi_1) = 2\ 3\ 4\ 5\ 6\ 7\ 1 \quad \sigma(\pi_{1_2}) = 2\ 1 \\
\pi_{1_3} &= \mathrm{sh}_{(2,2)} \circ \mathrm{sd}_{(4,4)} \circ \mathrm{sh}_{(5,5)}(\pi_1) = 2\ 3\ 4\ 5\ 6\ 7\ 1 \quad \sigma(\pi_{1_3}) = 2\ 1 \\
\pi_{1_4} &= \mathrm{sd}_{(2,3)} \circ \mathrm{sh}_{(5,5)} \circ \mathrm{sh}_{(2,2)}(\pi_1) = 5\ 6\ 7\ 1\ 2\ 3\ 4 \quad \sigma(\pi_{1_4}) = 2\ 1
\end{aligned}$$

Beispiel 5.12 Gegeben sei die nicht zu sortierende Permutation $\pi_2 = 5\ 1\ 3\ 6\ 8\ 2\ 4\ 7\ 9$. Es gibt verschiedene Strategien, welche unterschiedliche Resultate erzeugen, aber in ihrer Struktur übereinstimmen.

$$\begin{array}{ll}
\pi_{2_1} = \mathrm{sd}_{(2,2)} \circ \mathrm{sd}_{(7,7)}(\pi_2) = 5\ 1\ 2\ 3\ 6\ 7\ 8\ 4\ 9 & \sigma(\pi_{2_1}) = 3\ 1\ 4\ 2\ 5 \\
\pi_{2_2} = \mathrm{sd}_{(2,2)} \circ \mathrm{sd}_{(8,8)}(\pi_2) = 5\ 1\ 2\ 3\ 6\ 4\ 7\ 8\ 9 & \sigma(\pi_{2_2}) = 3\ 1\ 4\ 2\ 5 \\
\pi_{2_3} = \mathrm{sd}_{(3,3)} \circ \mathrm{sd}_{(7,7)}(\pi_2) = 5\ 1\ 6\ 7\ 8\ 2\ 3\ 4\ 9 & \sigma(\pi_{2_3}) = 3\ 1\ 4\ 2\ 5 \\
\pi_{2_4} = \mathrm{sd}_{(3,3)} \circ \mathrm{sd}_{(8,8)}(\pi_2) = 5\ 1\ 6\ 2\ 3\ 4\ 7\ 8\ 9 & \sigma(\pi_{2_4}) = 3\ 1\ 4\ 2\ 5
\end{array}$$

5.3 Sequentielle Komplexität

In diesem Abschnitt steht die Frage im Vordergrund, ob aus der Menge von Strategien zu einer vorzeichenbehafteten Permutation bestimmte Strategien zu bevorzugen sind. Da wir Permutationen sortieren möchten, ziehen wir natürlich sortierende Strategien denen vor, welche zu geblockten Permutationen führen. Des Weiteren lassen sich Bestimmungen vornehmen, durch die wir Strategien bewerten können. In [LPR08] wurde die Anzahl von Operationen einer Strategie betrachtet, welche benötigt wird, um eine gegebene vorzeichenbehaftete Permutation zu sortieren. Die Länge einer sortierenden Strategie wurde dabei als sequentielle Komplexität bezeichnet. Konstante sequentielle Komplexität eines Gene Assembly Modells bedeutet demnach, dass jede sortierende Strategien zu einer vorzeichenbehafteten Permutation die gleiche Anzahl intramolekularer Operationen benötigt.

Definition 5.13 (Konstante sequentielle Komplexität. Nach [LPR08]) Die sequentielle Komplexität eines Gene Assembly Modells ist konstant, wenn für jede sortierende Strategie Φ und Ψ zu einer gegebenen vorzeichenbehafteten Permutation π gilt: $|\Phi| = |\Psi|$.

Anhand des Beispiels 5.14 zeigt sich, dass es im universellen Modell Unterschiede in der Länge von sortierenden Strategien gibt. Zusätzlich kann festgestellt werden, dass Permutationen mit Hilfe verschiedener Operationskombinationen sortiert werden können.

Beispiel 5.14 Gegeben sei die Permutation $\pi = 15\overline{24}36$. Zwei sortierende Strategien unterschiedlicher Länge wären folgende:

Strategie 1	**Strategie 2**
$\mathrm{hi}_1(\pi) = 12\overline{54}36$	$\mathrm{hi}_2 = 152346$
$\mathrm{hi}_3 \circ \mathrm{hi}_1(\pi) = 12\overline{543}6$	$\mathrm{dlad}_{1,5} \circ \mathrm{hi}_2 = 123456$
$\mathrm{hi}_2 \circ \mathrm{hi}_3 \circ \mathrm{hi}_1(\pi) = 123456$	

Wie in [LP08] gezeigt wurde, unterscheidet sich sowohl das elementare, als auch das einfache Modell in dieser Hinsicht vom universellen Modell. In jener Arbeit wurde ausgehend vom Resultat des schwachen Determinismus bewiesen, dass alle sortierenden Strategien für eine gegebene vorzeichenbehaftete Permutation dieselbe Länge vorweisen. Des Weiteren sind auch die Operationskombinationen dieselben. Das heißt, die Sortierfolgen zu einer gegebenen vorzeichenbehafteten Permutation haben dieselbe Anzahl sh- und sd-Operationen. Mit

Hilfe von Gewichten oder Kosten für Operationen könnte für das universelle Modell eine Unterscheidungsmöglichkeit eingeführt werden, um bestimmte Strategien Anderen vorzuziehen. [HLPR07] bietet eine ausführliche Betrachtung von Komplexitätsfragen zu Gene Assembly.

5.4 Entscheidbarkeit des Sortierproblems

Mit *Sortierproblem* wird die Fragestellung bezeichnet, ob es effektiv entscheidbar ist, dass eine gegebene vorzeichenbehaftete Permutation in einem bestimmten Modell sortieren lässt. Das universelle Modell muss aufgrund seiner Vollständigkeit nicht betrachtet werden, da jede Permutation mit Hilfe dieses Modells sortierbar ist, wodurch die Entscheidung des Sortierproblems immer positiv ausfällt.
Für das einfache Modell ist die Problemlösung ebenfalls nicht problematisch. Grundlage dafür ist die Konfluenzeigenschaft, dass entweder alle Strategien erfolgreich oder alle Strategien nicht erfolgreich sind (vgl. Abschnitt 5.2.2). Demnach kann das Sortierproblem entschieden werden, in dem eine beliebige Strategie gefunden wird. Je nach dem, ob es sich dabei um eine sortierende oder eine nicht sortierende Strategie handelt, wird das Sortierproblem entsprechend entschieden. Um eine Strategie zu einer Permutation π zu finden, wird aus möglichen ausführbaren Operationen eine ausgewählt und auf π angewendet. Sind im Anschluss weitere Operationen möglich, wird solange erneut eine ausgewählt, bis π sortiert oder blockiert ist. Dieser Vorgang benötigt quadratischen Zeitaufwand [LP06].

Im elementaren Modell ist das Sortierproblem weitaus komplizierter. Sinnvoll erscheint es, die Problematik aufzuteilen und zunächst zu charakterisieren welche vorzeichenbehaftete Permutationen mit Hilfe der Operationen aus Eh und welche mit den Operationen aus Ed sortierbar sind. Die Eh-Sortierbarkeit vorzeichenbehafteter Permutationen lässt sich vergleichsweise leicht angeben:

Satz 5.15 [HPRR08]. Eine vorzeichenbehaftete Permutation π ist Eh-sortierbar, genau dann, wenn

(i) $||\pi|| = p(p+1)\dots n1\dots(p-1)$, mit $1 \leq p \leq n$ und es existieren r, t mit $1 \leq r \leq p-1$ und $p \leq t \leq n$, so dass r und t nicht gekennzeichnete Elemente sind, oder

(ii) $||\pi|| = (p-1)\dots 1n\dots(p+1)p$, mit $1 \leq p \leq n$ und es existieren r, t mit $1 \leq r \leq p-1$ und $p \leq t \leq n$, so dass r und t gekennzeichnete Elemente sind.

In ersterem Fall wird π zu $p(p+1)\dots n1\dots(p-1)$ sortiert, während im zweiten Fall π zu $\overline{(p-1)}\dots\overline{1}\overline{n}\dots\overline{(p+1)}\overline{p}$ sortiert wird.

Beispiel 5.16 Die Permutation $\pi_1 = 432\overline{1}87\overline{6}5$ ist Eh-sortierbar. Eine Strategie für π_1 ist $(\mathrm{eh}_7 \circ \mathrm{eh}_6 \circ \mathrm{eh}_5 \circ \mathrm{eh}_3 \circ \mathrm{eh}_2 \circ \mathrm{eh}_1)(\pi_1) = \overline{4}\ \overline{3}\ \overline{2}\ \overline{1}\ \overline{8}\ \overline{7}\ \overline{6}\ \overline{5}$. Es sei darauf hingewiesen, dass eh_2 erst nach eh_1 angewendet werden kann, genau wie sh_3 erst nach eh_2 und ebenso eh_7 nach eh_6.
Die Permutation $\pi_2 = 43218\overline{7}\overline{6}5$ ist nicht Eh-sortierbar, da die Elemente $1, 2, 3, 4$ nicht gekennzeichnet werden können.

Die Beschreibung von Ed-sortierbaren vorzeichenbehafteten Permutationen ist aufwendig. Aufgrund dessen, dass Operationen aus Ed die Kennzeichnung der Elemente einer Permutation nicht verändern, ist es offensichtlich, dass Permutationen, welche Elemente verschiedener

Vorzeichen enthalten, unter Ed nicht zu sortieren sind. Damit reduziert sich die Betrachtung auf die beiden Fälle, dass die Elemente einer Permutation entweder alle gekennzeichnet sind oder alle Elemente sind es nicht. Da diese beiden Fälle im Hinblick auf ihre Invertierung symmetrisch sind, ist die Betrachtung von nur einem der Fälle ausreichend. Wir betrachten in der Folge lediglich Permutationen deren Elemente alle nicht gekennzeichnet sind. Für die Angabe Ed-sortierbarer Permutationen soll zunächst der Abhängigkeitsgraph einer Permutation eingeführt werden.

5.4.1 Der Abhängigkeitsgraph

Der Abhängigkeitsgraph beschreibt für eine nicht vorzeichenbehaftete Permutation π die Reihenfolge von Ed-Operationen, welche in einer auf π anwendbaren Strategie benutzt werden können. Um den Aufbau des Graphen besser verstehen zu können, seien zunächst einige Beobachtungen aus [HPRR08] wiedergegeben, welche direkt aus der Definition von ed folgen.

LEMMA 5.17 Sei π eine vorzeichenbehaftete Permutation über Σ_n und $p \in \Sigma_n$.

(i) Wenn $p(p+1) \leq \pi$, dann gilt für jede Strategie Φ von auf π anwendbaren Ed-Operationen: $p(p+1) \leq \Phi(\pi)$.

(ii) Wenn $p(p+1) \leq \pi$, dann können die Operationen ed_p und ed_{p+1} in keiner Strategie auf π genutzt werden.

(iii) ed_1 und ed_n können in keiner Strategie angewendet werden.

Daraus ergibt sich zusätzlich:

(i) ed_{p-1} und ed_p können nicht in derselben Strategie für π angewendet werden.

(ii) ed_p kann höchstens einmal in einer Strategie für π angewendet werden.

Bei dem Abhängigkeitsgraphen handelt es sich um einen gerichteten Graphen mit reflexiven Schlingen.

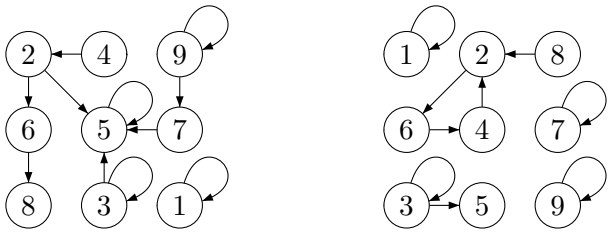
DEFINITION 5.18 (ABHÄNGIGKEITSGRAPH [HPRR08]) Für eine Permutation $\pi \in \Sigma_n$ sei ihr Abhängigkeitsgraph als der gerichtete Graph $\Gamma_\pi = (V = \Sigma_n, E)$ definiert, wobei

$$\begin{aligned} E = \; & \{(p,q) \mid (q-1)p(q+1) \leq_s \pi, \text{ mit } 1 \leq p \leq n, 2 \leq q \leq n-1\} \cup \\ & \{(q,q) \mid (q+1)(q-1) \leq_s \pi \text{ oder } q = 1 \text{ oder } q = n\}. \end{aligned}$$

Eine Kante (p,q) im Abhängigkeitsgraphen sagt aus, dass die Operation ed_q erst angewendet werden kann, wenn ed_p ausgeführt wurde. Dies lässt sich mit Hilfe der Definition der ed-Operation nachvollziehen. Demnach kann ed_q nur angewandt werden, sofern $(q-1)(q+1)$ ein Teilwort von π ist. Um dieses Teilwort zu erhalten, muss das Element p aus der Subsequenz $(q-1)p(q+1)$ entfernt werden. Dies ist nur über die Anwendung von ed_p möglich. Ein vollständiger Induktionsbeweis findet sich in [HPRR08].

Eine reflexive Schlinge (q, q) bedeutet, dass die Operation ed_q niemals in einer Strategie für π ausgeführt werden kann. Es sei darauf hingewiesen, dass eine reflexive Schlinge (q, q) auch durch $(q-1)q(q+1) \leq_s \pi$ in G_π enthalten sein kann. Für den Beweis dieser Beobachtung sei auf [HPRR08] verwiesen. Der zweite Teil der Definition des Graphen besagt, dass bei Vorhandensein der Subsequenz $(q+1)(q-1) \leq_s \pi$ die Operation ed_q in einer Strategie zu π nicht angewendet werden kann. Die Anwendung ist aus folgendem Grund nicht möglich:

Um ed_q anzuwenden, müssten wir zunächst das Teilwort $(q-1)(q+1)$ erhalten. Dafür müsste aber entweder ed_{q-1} oder ed_{q+1} angewendet werden. Durch die Vorgaben von Lemma 5.17 könnte im Anschluss ed_q nicht mehr benutzt werden.



(a) (b)

Abbildung 5.1: Abhängigkeitsgraph: (a) zu Permutation $\pi_1 = 143527698$ und (b) Permutation $\pi_2 = 184365279$

Beispiel 5.19 Gegeben sei die Permutation $\pi_1 = 143527698$. Ihr Abhängigkeitsgraph ist in Abbildung 5.1 (a) zu sehen. Die Knoten 1 und 9 besitzen definitionsgemäß je eine reflexive Schlinge. Ebenso die Knoten 3 (aufgrund der Subsequenz $(q+1)(q-1) = (3+1)(3-1) = 42 \leq_s \pi$) und 5 (aufgrund der Subsequenz $(q-1)p(q+1) = (5-1)5(5+1) = 456 \leq_s \pi$). Die Operationen $\mathrm{ed}_1, \mathrm{ed}_3, \mathrm{ed}_5$, und ed_9 sind in einer Strategie für π_1 nicht anwendbar. Dies ist korrekt, da wir von Lemma 5.17 wissen, dass ed_1 und ed_n niemals angewendet werden dürfen. Um ed_3 und ed_5 anwenden zu können, müssten wir zunächst eine Strategie anwenden, durch welche wir die Teilwörter 24 und 46 erhalten. Um das Teilwort 24 zu bekommen, müssten wir aber zumindest eine der Operationen ed_2 oder ed_4 anwenden. Doch Lemma 5.17 besagt, dass wir niemals ed_2 und ed_3 beziehungsweise ed_3 und ed_4 anwenden dürfen. Aus diesem Grund können wir in keiner Strategie zu π_1 die Operation ed_3 anwenden. Durch die gleichen Folgerungen trifft dies ebenso auf ed_5 zu. Im Abhängigkeitsgraphen ist die Kante (9,7) enthalten. Diese besagt, dass die Operation ed_7 erst nach Ausführung von ed_9 anzuwenden ist. Allerdings ist ed_9 niemals anzuwenden und somit ist auch ed_7 in keiner Strategie zu π_1 ausführbar. Im Abhängigkeitsgraph sind außerdem die Kanten $(4, 2), (2, 6)$ und $(6, 8)$ enthalten. Der Graph gibt uns damit eine Strategie für die Sortierung von π_1 vor: Zunächst ed_4, dann ed_2, dann ed_6 und schließlich ed_8. Wie wir sehen sortiert diese Strategie die Permutation π_1: $\mathrm{ed}_8 \circ \mathrm{ed}_6 \circ \mathrm{ed}_2 \circ \mathrm{ed}_4(\pi_1) = 123456789$.

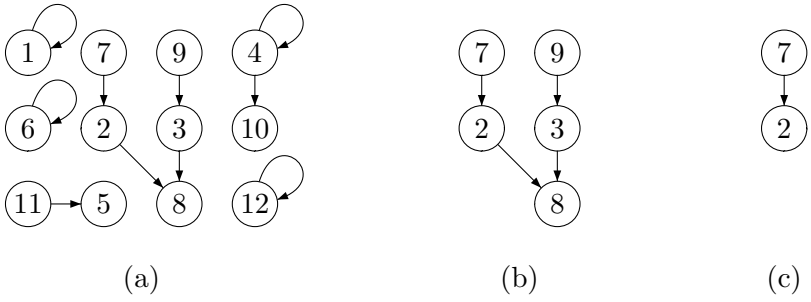
Beispiel 5.20 Gegeben sei die Permutation $\pi_2 = 184365279$. Der zugehörige Abhängigkeitsgraph ist in Abbildung 5.1 zu sehen. Die Knoten $1, 3, 7, 9$ besitzen reflexive Schlingen und die entsprechenden Operationen ed_1, ed_3, ed_7 und ed_9 können daher niemals in einer Strategie für π_2 angewendet werden. Die Kante $(3, 5)$ besagt, dass ed_5 nach Ausführung von ed_3 angewendet werden kann. Da aber ed_3 in keiner Strategie für π_2 enthalten sein darf, kann ed_5 ebenso niemals auftreten.

Fortsetzung Beispiel 5.20 Die Knoten $2, 4, 6$ bilden zusammen eine Schleife. Dies bedeutet, dass ed_2 erst nach ed_4 ausgeführt werden darf, ed_4 erst nach ed_6 und ed_6 erst nach ed_2. Lemma 5.17 enthält aber die Forderung, dass jede ed-Operation höchstens einmal ausgeführt werden darf. Somit können die Operationen ed_2, ed_4 und ed_6 in keiner Strategie für π_2 angewendet werden. Es existiert keine eingehende Kante zum Knoten 8. Entsprechend kann die Operation ed_8 auf π_2 angewendet werden. In der Tat ist dies die einzig mögliche Operation. Wir erhalten $\mathrm{ed}_8(\pi_2) = 143652789$. Weitere Operationen sind nicht anwendbar, daher ist π_2 im Endergebnis nicht zu sortieren.

5.4.2 Verbotene und bewegliche Elemente

Wie wir gesehen haben, ist es durch den Abhängigkeitsgraphen möglich Elemente zu erkennen, die in keiner Strategie, unabhängig ob sortierend oder nicht, zu einer Permutation π verwendet werden dürfen. Diese Elemente werden *verbotene Elemente* genannt. Ist ein Element nicht verboten, so wird es *beweglich* genannt. Mit $F(\pi)$ sei nachfolgend die Menge der verbotenen Elemente notiert, mit $M(\pi)$ die der beweglichen [PR08].
Eine Permutation π, in der ihre verbotenen Elemente nicht sortiert vorliegen, kann offensichtlich nicht sortiert werden, da diese in ihrer Reihenfolge feststehen. Genau genommen konnte bewiesen werden, dass eine Permutation π sortierbar ist, genau dann wenn $\pi|_{F(\pi)}$ sortiert vorliegt [PR08]. Um das Ed-Sortierproblem entscheiden zu können, ist es also wichtig effektiv entscheiden zu können welche Elemente einer Permutation verboten sind.
Um zu erkennen, ob ein Element p einer Permutation π der Menge $F(\pi)$ zugeordnet werden muss, ist die Betrachtung des durch p aufgespannten Teilgraphen $\Gamma_{\pi,p} = (T_{\pi,p}, E_{\pi,p})$ zu dem Abhängigkeitsgraph $\Gamma_\pi = (V_\pi, E_\pi)$ notwendig. Es folgt die Definition des Teilgraphen und ein entsprechendes Beispiel. Im Anschluss daran wird ein Ergebnis aus [PR08] angegeben, welches die Bedingungen zeigt, unter denen ein Element p der Menge $F(\pi)$ zugeordnet wird.

Definition 5.21 (Teilgraph $\Gamma_{\pi,p}$. [PR08]) Für eine Permutation π sei $\Gamma_\pi = (V_\pi, E_\pi)$ ihr Abhängigkeitsgraph. Für ein Element $p \in V_\pi$ sei $\Gamma_{\pi,p} = (T_{\pi,p}, E_{\pi,p})$ der Teilgraph zu p, mit $T_{\pi,p} = \left\{r \in V_\pi \mid r \rightarrow^+_{\Gamma_\pi} p\right\} \cup \{p\}$ und $E_{\pi,p} = \{(i,j) \in E_\pi \mid i, j \in T_{\pi,p}\}$.



(a) (b) (c)

Abbildung 5.2: Abhängigkeitsgraph zur Permutation $\pi = 1\ 7\ 3\ 2\ 9\ 4\ 11\ 6\ 5\ 8\ 10\ 12$ (a). Teilgraph $\Gamma_{\pi,8} = (T_{\pi,8}, E_{\pi,8})$ für den Knoten 8 (b). Teilgraph $\Gamma_{\pi,2} = (T_{\pi,2}, E_{\pi,2})$ für den Knoten 2 (c).

SATZ 5.22 [PR08] Sei π eine Permutation über Σ_n und $\Gamma_\pi = (V_\pi, E_\pi)$ ihr Abhängigkeitsgraph. Ein Element $p \in \Sigma_n$ ist verboten in π, genau dann, wenn der Teilgraph $\Gamma_{\pi,p} = (T_{\pi,p}, E_{\pi,p})$ zyklisch ist oder $(q-1), q \in T_{\pi,p}$ für ein Element q.

Die erste Bedingung des Satzes sagt aus, dass p verboten ist, wenn p Teil einer Schleife im Graphen ist oder direkt von einer Schleife abhängt. Der zweite Teil besagt, dass p verboten ist, wenn im Teilgraph zwei Pfade $(q-1) \rightarrow^+_{\Gamma_{\pi,p}} p$ und $q \rightarrow^+_{\Gamma_{\pi,p}} p$ existieren. Das Verbot liegt darin begründet, dass um die Operation ed_p zu verwenden, zunächst die Operationen ed_{q-1} und ed_q angewendet werden müssten. Dies ist aber laut Lemma 5.17 nicht erlaubt. Für den Beweis des Satzes sei auf [PR08] verwiesen.

BEISPIEL 5.23 Gegeben sei die Permutation $\pi = 1\ 7\ 3\ 2\ 9\ 4\ 11\ 6\ 5\ 8\ 10\ 12$. Der zugehörige Abhängigkeitsgraph ist in Abbildung 5.2 (a) zu sehen. Die Knoten $1, 4, 6$ und 12 besitzen Eigenschleifen. Das heißt, die zugehörigen Operationen ed_1, ed_4, ed_6 und ed_{12} sind in keiner Strategie für π anzuwenden. Die Kante $(4, 10)$ sagt aus, dass ed_{10} erst nach ed_4 ausgeführt werden kann. Aufgrund dessen, dass ed_4 in keiner Strategie zur Anwendung kommen kann, ist auch die Ausführung von ed_{10} nicht möglich. Die Kante $(11, 5)$ bedeutet, dass ed_{11} vor ed_5 anzuwenden ist. Um zu kontrollieren ob die Operation ed_8 anwendbar ist, schauen wir uns den Teilgraphen zu Knoten 8 an. Dieser ist in Abbildung 5.2 (b) sehen. Der Graph enthält keine Schleifen, demnach bleibt die erste Bedingung aus Satz 5.22 unerfüllt. Der Teilgraph enthält jedoch die beiden Knoten 2 und 3, wodurch die zweite Bedingung des Satzes 5.22 nach zwei aufeinander folgenden Elementen $(q-1), q \in T_{\pi,2}$ erfüllt wird. Somit gehört der Knoten 8 zu den verbotenen Elementen, gleichbedeutend damit, dass ed_8 in keiner Strategie zu *pi* angewendet werden kann. In Abbildung 5.2 (c) ist der Teilgraph zu Knoten 2 abgebildet. Es ist leicht zu erkennen, dass Element 2 nicht zu den verbotenen Elementen zählt. Allerdings sehen wir, dass ed_2 erst nach der Anwendung von ed_7 ausgeführt werden kann.

5.4.3 Entscheidungsalgorithmus

Mit Hilfe des Auffindens der verbotenen Elemente lässt sich ein Entscheidungsalgorithmus angeben, welcher zur Eingabe einer nicht vorzeichenbehafteten Permutation π ausgibt, ob diese ed-sortierbar ist oder nicht. Der Algorithmus ist [PR08] entnommen.

- *Eingabe:* Permutation π über $\Sigma_n = \{1, \ldots, n\}$;
- *Schritt 1:* Konstruktion des Abhängigkeitsgraphen $G_\pi = (V = \Sigma_n, E)$
- *Schritt 2.1:* Aufstellen einer Menge U_1 von Knoten, welche im Graphen in Schleifen enthalten sind, einschließlich reflexiver Schlingen.
- *Schritt 2.2:* Aufstellen einer Menge $U_2 = \left\{p \in V \mid \{r-1, r\} \rightarrow^+_G p\right\}$
- *Schritt 2.3:* Die Menge der verbotenen Elemente zu π wird mit $F(\pi) = U_1 \cup U_2 \cup (p \in V \mid U_1 \rightarrow^+_G p)$ repräsentiert.
- *Ausgabe:* „Ja“, wenn $F|_\pi$ sortiert ist. Andernfalls „Nein“.

Beispiel 5.24 Gegeben sei die Permutation $\pi_1 = 1\ 7\ 3\ 2\ 9\ 4\ 11\ 6\ 5\ 8\ 10\ 12$.

- *Schritt 1:* Konstruktion des Abhängigkeitsgraphen: Abbildung 5.2 (a)
- *Schritt 2.1:* $U_1 = \{1, 4, 6, 12\}$
- *Schritt 2.2:* $U_2 = \{8\}$ vgl. 5.2 (b)
- *Schritt 2.3:* $F(\pi_1) = U_1 \cup U_2 \cup (p \in V \mid U_1 \rightarrow_G^+ p) = \{1, 4, 6, 12\} \cup \{8\} \cup \{10\}$
- *Ausgabe:* Die Menge $F(\pi_1)$ der verbotenen Elemente in der Permutation π_1 liegt sortiert vor: $\pi|_{F(\pi_1)} = 1\ 4\ 6\ 8\ 10\ 12 \leq_s \pi$. Ausgabe: „Ja“

In der Tat lässt sich π_1 sortieren. Eine mögliche Strategie wäre dafür zum Beispiel: $\mathrm{ed}_3 \circ \mathrm{ed}_7 \circ \mathrm{ed}_5 \circ \mathrm{ed}_9 \circ \mathrm{ed}_{11}(\pi_1) = 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10\ 11\ 12$.

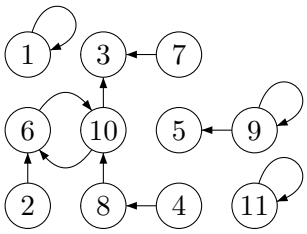


Abbildung 5.3: Abhängigkeitsgraph zu Permutation $\pi_2 = 1\ 3\ 5\ 2\ 10\ 7\ 4\ 9\ 6\ 8\ 11$.

Beispiel 5.25 Gegeben sei die Permutation $\pi_2 = 1\ 3\ 5\ 2\ 10\ 7\ 4\ 9\ 6\ 8\ 11$.

- *Schritt 1:* Konstruktion des Abhängigkeitsgraphen: Abbildung 5.3.
- *Schritt 2.1:* $U_1 = \{1, 6, 9, 10, 11\}$
- *Schritt 2.2:* $U_2 = \{3\}$
- *Schritt 2.3:* $F(\pi_2) = U_1 \cup U_2 \cup (p \in V \mid U_1 \rightarrow_G^+ p) = \{1, 6, 9, 10, 11\} \cup \{3\} \cup \{5\}$
- *Ausgabe:* Die Menge $F(\pi_1)$ der verbotenen Elemente in der Permutation π_2 ist nicht sortiert: $\pi|_{F(\pi_2)} = 1\ 3\ 5\ 10\ 9\ 6\ 11$. Ausgabe: „Nein“

5.5 Charakterisierung ed-sortierbarer Permutationen

In folgendem, in [HPRR08] bewiesenen, Satz werden nicht vorzeichenbehaftete Permutation charakterisiert, welche unter Ed sortierbar sind. Eine ähnliche, aber weitaus komplexere Charakterisierung, die ebenfalls in [HPRR08] zu finden ist, beschreibt die Sortierbarkeit unter $\{\mathrm{Eh} \cup \mathrm{Ed}\}$ für vorzeichenbehaftete Permutationen.

Satz 5.26 [HPRR08]. Eine nicht gekennzeichnete Permutation π ist Ed-sortierbar, genau dann, wenn eine Partitionierung $\{1, 2, \ldots, n\} = D \cup U$ existiert, so dass folgende Bedingungen erfüllt sind:

(i) $\pi|_U$ ist sortiert;

(ii) Der durch D induzierte Subgraph in G_π ist nicht zyklisch;

(iii) Wenn $(p, q) \in G_\pi$ mit $q \in D$, dann ist auch $p \in D$;

(iv) Für jedes $p \in D$ gilt, $(p-1)(p+1) \leq_s \pi$;

(v) Für jedes $p \in D$ gilt, $(p-1), (p+1) \in U$.

Für das einfache Modell existiert bisher noch keine solche Charakterisierung sortierbarer Permutationen. Im sich anschließenden Abschnitt 6 möchten wir aber eine solche Charakterisierung aufstellen. Für das universelle Modell ist eine Charakterisierung nicht notwendig, da in diesem Modell alle Permutationen erfolgreich sortiert werden können. Es folgt ein Beispiel zur Charakterisierung von Ed-sortierbaren Permutationen.

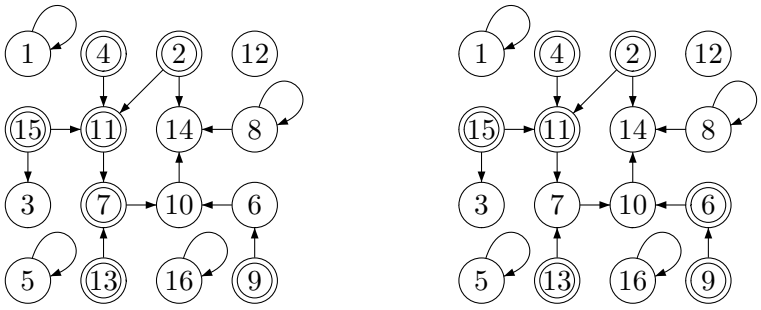


Abbildung 5.4: Abhängigkeitsgraph zu Permutation $\pi = 1\ 3\ 5\ 9\ 7\ 6\ 11\ 13\ 8\ 10\ 2\ 15\ 4\ 12\ 14\ 16$.

Beispiel 5.27 Gegeben sei die Permutation $\pi = 1\ 3\ 5\ 9\ 7\ 6\ 11\ 13\ 8\ 10\ 2\ 15\ 4\ 12\ 14\ 16$. Abbildung 5.4 zeigt den zugehörigen Abhängigkeitsgraphen. Mit Hilfe dieses Graphen und Satz 5.26 leiten wir eine sortierende Strategie für π her. Das heißt wir suchen eine Partitionierung $\{1, 2, \ldots, 16\} = D \cup U$.
Aus Eigenschaft (ii) folgt, dass $1, 5, 8, 16, 14 \in U$. Sei $3 \in D$, dann folgt aus (iii), dass auch $15 \in D$ und aus (v), dass die Elemente $2, 4 \in U$. Diese Verteilung widerspricht aber Eigenschaft (i), da $\pi|_U = 1\ 5\ 8\ 2\ 4\ 14\ 16$ nicht sortiert ist. Demnach ist $3 \in U$ und $2, 4 \in D$. Nehmen wir an, dass $10 \in D$, dann müssten neben den Elementen $15, 11, 13$ und 9 auch $6, 7 \in D$ sein. Dies verstößt aber gegen Bedingung (v). Somit muss $10 \in U$ gelten. Sei $7 \in D$, dann muss auch $11, 13, 15 \in D$ und $6 \in U$ gelten. Aus $11 \in D$ folgt, dass $12 \in U$ ist.
Wir nehmen noch das Element 9 in die Menge D auf. Damit haben wir eine vollständige Partitionierung für G_π, welche in 5.4 (a) dargestellt ist, wobei Elemente aus D mit einem doppelte Kreis markiert sind:

$$D = \{2, 4, 7, 9, 11, 13, 15\}, U = \{1, 3, 5, 6, 8, 10, 12, 14, 16\}$$

FORTSETZUNG BEISPIEL 5.27 Die Aufteilung erfüllt die Bedingungen $(i) - (v)$. Die Permutation kann demnach durch eine Strategie mit Hilfe der Operationen ed_p mit $p \in D$ sortiert werden. Der Abhängigkeitsgraph legt folgende Reihenfolge der Operationen fest: $\mathrm{ed}_{15}, \mathrm{ed}_2, \mathrm{ed}_4$ vor ed_{11} und $\mathrm{ed}_{11}, \mathrm{ed}_{13}$ vor ed_7. Die sonstige Abfolge kann beliebig sein. Eine Möglichkeit π zu ordnen wäre:

$$\mathrm{ed}_9 \circ \mathrm{ed}_7 \circ \mathrm{ed}_{11} \circ \mathrm{ed}_{15} \circ \mathrm{ed}_2 \circ \mathrm{ed}_4 \circ \mathrm{ed}_{13}(\pi) = 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10\ 11\ 12\ 13\ 14\ 15\ 16$$

Die Aufteilung der Mengen D und U kann variabel sein. So könnte in unserem Beispiel statt $7 \in D,\ 6 \in U$ auch $6 \in D,\ 7 \in U$ gewählt werden. Der zugehörige Graph ist in Abbildung 5.4 (b) zu sehen. Eine entsprechende sortierende Strategie wäre folgende:

$$\mathrm{ed}_6 \circ \mathrm{ed}_{11} \circ \mathrm{ed}_2 \circ \mathrm{ed}_4 \circ \mathrm{ed}_{15} \circ \mathrm{ed}_{13} \circ \mathrm{ed}_9(\pi) = 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10\ 11\ 12\ 13\ 14\ 15\ 16$$

Kapitel 6

Charakterisierung sd-sortierbarer Permutationen

Während wir im vorangegangenen Abschnitt eine Charakterisierung für ed-sortierbare Permutationen kennen gelernt haben, gibt es bisher keine solche Charakterisierung für das einfache Modell [LPR08, Seite 10]. Die Aufstellung einer Charakterisierung sd-sortierbarer Permutationen gilt somit als offenes Problem [LPR08, Seite 12], [LP08, Seite 29]. Diesem Problem wenden wir uns im Folgenden zu und geben eine Charakterisierung sd-sortierbarer Permutationen an.

Das Vorhandensein einer Charakteristik sd-sortierbarer Permutationen kann der erste Schritt sein, um weitere noch offene Probleme zu bewältigen (vgl. Abschnitt 7.2). Eine natürliche Vorgehensweise wäre die nachfolgend angegebene Charakterisierung um sh-Operationen zu erweitern, womit dann das gesamte einfache Modell erfasst sein würde.

Die Vorgehensweise der Charakterisierung sd-sortierbarer Permutationen stimmt in Ihrer Grundidee mit der des elementaren Modells überein. Auch wir verwenden verbotene Elemente, verzichten jedoch auf den Abhängigkeitsgraphen. Stattdessen bestimmen wir über strukturelle Merkmale eine Menge von sd-Operationen, welche genau die anwendbaren Operationen zu einer gegebenen Permutation π enthält. Auf Grundlage dieser Menge erfolgt dann die eigentliche Charakterisierung der sd-sortierbaren Permutationen.

6.1 Ausgangsmenge

Im Unterschied zum Abhängigkeitsgraphen im elementaren Modell, in welchem auch Operationen auftauchen deren Voraussetzungen nicht erfüllbar und somit nie ausführbar sind, möchten wir für das einfache Modell genau die Operationen ermitteln die auch tatsächlich angewendet werden können. Um diese zu bestimmen, werden wir uns zunächst überlegen, welche Operationen zu einer Permutation π der Länge n theoretisch anwendbar sind. Anschließend reduzieren wir diese Menge von Operationen, so dass nur die tatsächlich ausführbaren übrig bleiben.

Eine Permutation $\pi \in \Sigma_n$ mit $\Sigma = \{1, \ldots, n\}$ enthält n Elemente. Jedes Elementpaar (p, q) mit $p, q \in \Sigma$ und $q - p \geq 2$ bietet die Möglichkeit der Anwendung einer sd-Operation $\mathrm{sd}_{(p+1,q-1)}$.

Für das Element $(n-2)$ gibt es somit genau ein Paar: $(n-2, n)$. Für $(n-3)$ gibt es zwei Paare, für $(n-4)$ drei Paare und so weiter. Formal ausgedrückt gibt es

$$\frac{(n-2) * ((n-2)+1)}{2}$$

Paare. Die Menge der zugehörigen Operation zu diesen Paaren bildet die Ausgangsmenge theoretisch anwendbarer sd-Operationen.

$$A = \{(p, q) \mid p, q \in \Sigma, q - p \geq 2\}$$

Die Menge A werden wir im Folgenden Schritt für Schritt auf die Paare reduzieren, deren zugehörige Operationen tatsächlich zu einer gegebenen Permutation π anwendbar sind. Die Elemente $(p+1)$ und $(q-1)$ zu einem Paar (p, q) nennen wir *Endelemente*.

6.2 Sequenzintervalle und Endelemente

Den Hintergrund für den Ausschluss von Paaren aus der Menge der theoretisch anwendbaren Operationen, bildet das Lemma 6.1. Es schließt sich Lemma 6.2 an, dessen Ergebnis im weiteren Verlauf immer wieder von Bedeutung sein wird.

LEMMA 6.1 Für eine beliebige nicht vorzeichenbehaftete Permutation π gilt, dass für die Anwendung einer Operation $\mathrm{sd}_{(p+1,q-1)}$ die folgenden Bedingungen erfüllt sein müssen:

(i) $pq \leq \pi$

(ii) $[(p+1), (q-1)] \leq \pi$

Beide Voraussetzungen sind direkt aus der Definition 4.8 von sd-Operationen des einfachen Modells ableitbar.

LEMMA 6.2 Für eine beliebige nicht vorzeichenbehaftete Permutation $\pi \in \Sigma_n$ gilt: Die Anwendung einer zu einem Elementpaar $(p, q) \in A$ zugehörigen Operation $\mathrm{sd}_{(p+1,q-1)}$ ist nur möglich, sofern die Elemente p und q, sowie die Endelemente $(p+1)$ und $(q-1)$, zuvor nicht Teil einer sd-Operation in einer Komposition $\Phi(\pi)$ waren.

Beweis Wir wissen das gilt: $1 \leq p < (p+1) \leq (q-1) < q \leq n$. Wir zeigen für jedes der vier Elemente, dass eine Ausführung von $\mathrm{sd}_{(p+1,q-1)}$ unmöglich wird, wenn die Elemente zuvor durch eine sd-Operation bewegt werden.

- Bewegung von p: Sei p beweglich. Dafür benötigen wir nach 6.1 einen sortierten Block $[u, p]$ mit $1 < u \leq p$ und ein entsprechendes Teilwort $(u-1)\ (p+1) \leq \pi$. Bei Ausführung von $\mathrm{sd}_{(u,p)}$ bildet sich der sortierte Block $[(u-1), (p+1)]$. Darin ist auch das Teilwort $p\ (p+1)$ enthalten. So kann das Teilwort $pq \leq \pi$ nie erhalten werden, was einen Widerspruch zu Lemma 6.1 darstellt.

- Bewegung von q: Sei q beweglich, durch $[q, v]$ mit $q \leq v < n$ und $(q-1)\ (v+1) \leq \pi$. Bei Ausführung von $\mathrm{sd}_{(q,v)}$ entsteht der sortierte Block $[(q-1), (v+1)]$ und das Teilwort $pq \leq \pi$ kann nie erhalten werden, was einen Widerspruch zu Lemma 6.1 darstellt.

- Bewegung von $(p+1)$: Sei $(p+1)$ beweglich. Dafür ist ein sortierter Block $[(p+1), t]$ mit $(p+1) \leq t < q$ und das Teilwort $p\ (t+1) \leq \pi$ erforderlich. Bei Ausführung von $\mathrm{sd}_{((p+1),t)}$ bildet sich aber der sortierte Block $[p, (t+1)]$ und das Teilwort $pq \leq \pi$ kann nie erhalten werden, was einen Widerspruch zu Lemma 6.1 darstellt.

- Bewegung von $(q-1)$: Sei $(q-1)$ beweglich. Dafür ist ein sortierter Block $[s, (q-1)]$ mit $p < s \leq (q-1)$ und das Teilwort $(s-1)\ q \leq \pi$ erforderlich. Bei Ausführung von $\mathrm{sd}_{(s,(q-1))}$ bildet sich aber der sortierte Block $[(s-1), q]$ und das Teilwort $pq \leq \pi$ kann nie erhalten werden, was einen Widerspruch zu Lemma 6.1 darstellt. □

Wir ermitteln nun Schritt für Schritt welche Paare (p, q) die Bedingungen aus Lemma 6.1 erfüllen können. Dabei betrachten wir Merkmale, die einer Erfüllung der Voraussetzungen widersprechen und streichen dementsprechend Sequenzintervalle aus der Ausgangsmenge der theoretisch anwendbaren Operationen. Es folgen die drei Lemmata 6.3, 6.4 und 6.5, die solche Merkmale repräsentieren. Lemma 6.3 besagt, dass Bedingung (i) aus Lemma 6.1 nur dann erfüllt sein kann, sofern für ein Paar (p, q) in einer Permutation π die Subsequenz pq enthalten ist. Die beiden sich anschließenden Lemmata zeigen, dass die Elemente 1 und n, sowie die Endelemente zu (p, q) nicht Bestandteil der Subsequenz $pq \leq_s \pi$ sein dürfen.

LEMMA 6.3 Für eine beliebige nicht vorzeichenbehaftete Permutation π der Länge n ist die Operation $\mathrm{sd}_{(p+1,q-1)}$ nur dann anwendbar, sofern $pq \leq_s \pi$ ist.

Beweis Sei $pq \not\leq_s \pi$. Dies widerspricht aber der Bedingung (i) aus Lemma 6.1. Somit müssen wir zumindest eine sd-Operation anwenden, um p oder q zu bewegen. Doch dies ist ein Widerspruch zu Lemma 6.2. □

Im Folgenden werden wir die Paare, welche Lemma 6.3 erfüllen, als *Sequenzintervalle* bezeichnen.

LEMMA 6.4 Für eine beliebige nicht vorzeichenbehaftete Permutation π der Länge n ist die Operation $\mathrm{sd}_{(p+1,q-1)}$ niemals anwendbar, sofern $p\ 1\ q \leq_s \pi$ oder $p\ n\ q \leq_s \pi$ ist.

Beweis Sei $p\ 1\ q \leq_s \pi$ (respektiv $p\ n\ q \leq_s \pi$). Um das in Lemma 6.1 geforderte Teilwort pq zu erhalten, müssen wir zumindest eine sd-Operation anwenden. Es gibt nach Definition der Operationen des einfachen Modells keine sd-Operation, welche das Element 1 (respektiv n) bewegt. Demnach müssen wir p oder q bewegen. Dies ist aber ein Widerspruch zu Lemma 6.2. □

LEMMA 6.5 Für eine beliebige nicht vorzeichenbehaftete Permutation π der Länge n ist die Operation $\mathrm{sd}_{(p+1,q-1)}$ niemals anwendbar, sofern $p(p+1)q \leq_s \pi$ oder $p(q-1)q \leq_s \pi$ ist.

Beweis Sei $p(p+1)q \leq_s \pi$. Nach Bedingung (i) aus Lemma 6.1, darf das Element $(p+1)$ bei einer Anwendung von $\mathrm{sd}_{(p+1,q-1)}$ nicht in der Subsequenz pq eines Paares (p,q) enthalten sein. Um den Zustand $p(p+1)q \leq_s \pi$ zu ändern, müssen wir daher zumindest eine sd-Operation anwenden, welche $p, (p+1)$ oder q bewegt. Doch dies steht im Widerspruch zu Lemma 6.2.

Der Beweis für $p(q-1)q \leq_s \pi$ verläuft gleichartig. □

Fassen wir die bisherigen Beobachtungen zusammen, lässt sich die Ausgangsmenge, der zu einer Permutation π theoretisch anwendbaren sd-Operationen, auf eine Menge S von Paaren (p,q) reduzieren:

$$S = \{(p,q) \mid pq \leq_s \pi, q-p \geq 2, p\ x\ q \not\leq_s \pi \text{ mit } x \in \{1,(p+1),(q-1),n\}\}$$

Diese Menge enthält lediglich Sequenzintervalle (Paare (p,q) mit $pq \leq_s \pi$), deren Endelemente $(p+1)$ und $(q-1)$ nicht im Sequenzintervall (p,q) liegen.

BEISPIEL 6.6 Gegeben sei die Permutation $\pi = 3\ 9\ 1\ 5\ 7\ 11\ 4\ 6\ 8\ 2\ 10\ 12$. Folgende Sequenzintervalle ergeben sich:

$$\begin{aligned} S = \{&(1,5),(1,7),(1,11),(1,4),(2,10),(2,12),(3,9),(4,6),(4,8),(4,10),\\ &(4,12),(5,7),(5,11),(6,8),(6,10),(6,12),(7,11),(8,10),(8,12),(10,12)\} \end{aligned}$$

Beispielsweise ist das Paar $(2,4)$ nicht enthalten, da $24 \not\leq_s \pi$, was einen Widerspruch zu Lemma 6.3 darstellt. Das Sequenzintervall $(1,6)$ ist nicht enthalten, da dessen Subsequenz mit Element 5 das Endelement zu 6 enthält, was nach Lemma 6.5 untersagt ist. Die Sequenzintervalle $(3,5)$ und $(9,11)$ zeigen exemplarisch einen Verstoß gegen Lemma 6.4, da deren Subsequenz das Startelement 1 der Permutation enthält.

6.3 Blockintervalle

Bis hierhin haben wir lediglich die erste Voraussetzung aus Lemma 6.1 als Begründung für den Ausschluss von Paaren betrachtet. Wir wollen uns daher jetzt der zweiten Bedingung widmen, welche besagt, dass wir für die Ausführung einer Operation $\mathrm{sd}_{(p+1,q-1)}$ auf eine Permutation π, einen sortierten Block $[(p+1),(q-1)]$ benötigen. Es gibt zwei grundsätzliche Beobachtungen, welche die Bildung eines solchen Blockes unmöglich machen.

LEMMA 6.7 Für eine beliebige nicht vorzeichenbehaftete Permutation π der Länge n ist die Operation $\mathrm{sd}_{(p+1,q-1)}$ nur dann anwendbar, sofern $(p+1)(q-1) \leq_s \pi$.

Beweis Sei $(p+1)(q-1) \not\leq_s \pi$. Um einen sortierten Block $[(p+1),(q-1)]$, wie in Lemma 6.1 gefordert, zu erhalten müssen wir zumindest eine sd-Operation anwenden, die das Element $(p+1)$ oder $(q-1)$ bewegt. Dies verbietet uns aber Lemma 6.2. □

Endelemente zu einem Sequenzintervall (p,q), welche die Bedingung 6.7 erfüllen, bezeichnen wir im Folgenden als *Blockintervalle*. Der Fall, in welchem $(p+1) = (q-1)$ gilt, bedeutet, dass

der geforderte sortierte Block $[(p+1),(q-1)]$ lediglich aus einem Element besteht, wodurch die zweite Bedingung aus Lemma 6.1 auf triviale Weise erfüllt ist.

Wie wir aus Lemma 6.5 bereits wissen, darf keines der Endelemente $(p+1)$ und $(q-1)$ zu einem Sequenzintervall (p,q) Bestandteil der Subsequenz $pq \leq_s$ sein. In Lemma 6.8 sehen wir, dass auch die Umkehrung - die Elemente p und q sind Teil der Subsequenz $(p+1)(q-1) \leq_s \pi$ - die Operationsausführung von $\mathrm{sd}_{(p+1,q-1)}$ unmöglich werden lässt.

LEMMA 6.8 Für eine beliebige nicht vorzeichenbehaftete Permutation π der Länge n ist die Operation $\mathrm{sd}_{(p+1,q-1)}$ niemals anwendbar, sofern $(p+1)pq(q-1) \leq_s \pi$.

Beweis Sei $(p+1)pq(q-1) \leq_s \pi$. Um den aus der zweiten Bedingung von Lemma 6.1 geforderten sortierten Block $[(p+1),(q-1)]$ zu erhalten, ist zumindest eine sd-Operation nötig, die das Element $(p+1)$ oder $(q-1)$ bewegt. Auch die Ausführung mindestens zweier sd-Operationen welche die Elemente p und q bewegen wäre möglich. Alle diese Varianten sind uns aber durch Lemma 6.2 untersagt. □

Wie bei den Sequenzintervallen darf auch die von den Elementen eines Blockintervalls aufgespannte Subsequenz nicht das Start- beziehungsweise das Endelement zu einer Permutation π enthalten.

LEMMA 6.9 Für eine beliebige nicht vorzeichenbehaftete Permutation π der Länge n ist die Operation $\mathrm{sd}_{(p+1,q-1)}$ niemals anwendbar, sofern $(p+1)\ 1\ (q-1) \leq_s \pi$ oder $(p+1)\ n\ (q-1) \leq_s \pi$.

Beweis Sei $(p+1)\ 1\ (q-1) \leq_s \pi$ (respektiv $(p+1)\ n\ (q-1) \leq_s \pi$). Um den in Lemma 6.1 geforderten, sortierten Block $[(p+1),(q-1)]$ zu erhalten, müssen wir zumindest eine sd-Operation anwenden. Nach Definition der Operationen des einfachen Modells gibt es keine sd-Operation, welche das Element 1 (respektiv n) bewegt. Demnach müssen wir $(p+1)$ oder $(q-1)$ bewegen. Dies liefert aber in Hinblick auf Lemma 6.2 einen Widerspruch. □

Die Ergebnisse von Lemma 6.7, 6.8 und 6.9 lassen eine Eliminierung von Sequenzintervallen aus der Menge S zu. Die reduzierte Menge bezeichnen wir mit mit S_{Op}.

$$S_{Op} = \{(p,q) \mid (p,q) \in S,\ (p+1)(q-1) \leq_s \pi,\ (p+1)pq(q-1) \not\leq_s \pi,\\ (p+1)\ 1\ (q-1) \not\leq_s \pi,\ (p+1)\ n\ (q-1) \not\leq_s \pi\}$$

Die Menge S_{Op} ist eine, um die Bedingungen aus Lemma 6.7 und 6.8 reduzierte, Teilmenge von S.

Beispiel 6.10 Wir bearbeiten weiter die Permutation $\pi = 3\ 9\ 1\ 5\ 7\ 11\ 4\ 6\ 8\ 2\ 10\ 12$. Aus dem vorangegangen Beispiel kennen wir die Menge S.

$$S = \{(1,5),(1,7),(1,11),(1,4),(2,10),(2,12),(3,9),(4,6),(4,8),(4,10),\\ (4,12),(5,7),(5,11),(6,8),(6,10),(6,12),(7,11),(8,10),(8,12),(10,12)\}$$

Die Menge S_{Op} sieht folgendermaßen aus:

$$S_{Op} = \{(1,11),(2,10),(3,9),(4,6),(4,8),(4,12),\\ (5,7),(5,11),(6,8),(6,12),(7,11),(8,10),(10,12)\}.$$

Wie zu erkennen ist, gibt es für ein Sequenzintervall (p,q) mit Abstand $q-p=2$ und somit $(p+1)=(q-1)$, immer das zugehörige Blockintervall, da diese Grundelemente der Permutation sind. Im Beispiel triff dies auf die Sequenzintervalle $(4,6)$, $(5,7)$, $(6,8)$, $(8,10)$, und $(10,12)$ zu. Wir erkennen, dass die zugehörigen Blockintervalle genau die Operationen sind, welche auch im elementaren Modell angewendet werden könnten.
Zu dem Sequenzintervall $(1,11)$ ist das zugehörige Blockintervall $(2,10)$. Wir sehen, dass $2\ 10 \leq_s \pi$ gilt und auch, dass $(2,10)$ weder sein Sequenzintervall $(1,11)$ umspannt, noch das Start- oder Endelement der Permutation in der Subsequenz $2\ 10 \leq_s \pi$ enthalten ist. Somit sind alle Bedingungen erfüllt und das Sequenzintervall $(1,11)$ befindet sich zu Recht in der Menge S_{Op}.
Die zu den Sequenzintervallen $(1,5),(1,7),(1,4),(4,10)$ und $(6,10)$ zugehörigen Operationspaare erfüllen nicht die notwendige Eigenschaft aus Lemma 6.7, dass sie als Subsequenz in π vorliegen müssen. Beispielsweise sehen wir bei Sequenzintervall $(1,5)$, dass für das zugehörige Operationspaar $(2,4)$ gilt: $2\ 4 \not\leq_s \pi$.
Das Sequenzintervall $(2,12)$ ist nicht Teil von S_{Op}, weil dessen zugehöriges Blockintervall $(3,11)$ in der Subsequenz $3\ 11 \leq_s \pi$ das Startelement 1 enthält. Dies verbietet uns aber Lemma 6.9. Das gleiche trifft auf das zum Sequenzintervall $(8,12)$ gehörige Blockintervall $(9,11)$ zu.

6.4 Operationsvoraussetzungen

Wir haben die grundsätzlichen Umstände untersucht, welche gegeben sein müssen, damit wir zu einem Sequenzintervall (p,q) die Operation $\mathrm{sd}_{(p+1,q-1)}$ anwenden können. Dabei haben wir festgehalten, wie die Elemente p und q in einer Permutation π angeordnet sein müssen, damit das in 6.1 geforderte Teilwort $pq \leq \pi$, sowie der sortierte Block $[(p+1)(q-1)]$, erhalten werden können. Jetzt interessieren wir uns im Detail dafür, welche Operationen notwendig sind, um diese Grundvoraussetzungen zu erfüllen. Für den Erhalt des Teilworts $pq \leq \pi$ muss festgestellt werden, welche Elemente innerhalb der von p und q aufgespannten Subsequenz liegen und durch welche Operationen diese entfernt werden können. Diese notwendig auszuführenden Operationen sind *Sequenzvoraussetzungen* zu (p,q).
Um einen sortierten Block $[(p+1)(q-1)]$ zu erhalten, müssen nicht zugehörige Elemente aus der Subsequenz $(p+1)(q-1) \leq_s$ entfernt werden. Zusätzlich müssen die Operationen bestimmt werden, durch welche sich letztlich der Block bildet. Diese Operationen sind *Blockvoraussetzungen* zu (p,q).

Wie im elementaren Modell gibt es auch bei der Verwendung von einfachen Operationen

direkt aus den Operationsdefinitionen abgeleitete Beobachtungen, die es zu beachten gilt.

LEMMA 6.11 Sei π eine nicht vorzeichenbehaftete Permutation über Σ_n und $p \in \Sigma_n$.

- $\mathrm{sd}_{(1,r)}$ und $\mathrm{sd}_{(s,n)}$ mit $1 \leq r, s \leq n$ können in keiner Strategie angewendet werden;
- Wenn $p(p+1) \leq \pi$, dann gilt für jede Strategie Φ von auf π anwendbaren Sd-Operationen: $p(p+1) \leq \Phi(\pi)$;
- Wenn $p(p+1) \leq \pi$, dann können die Operationen $\mathrm{sd}_{(t,p)}$ und $\mathrm{sd}_{(p+1,u)}$ mit $1 < t \leq p$ und $(p+1) \leq u < n$ in keiner Strategie auf π genutzt werden.

Daraus ergibt sich zusätzlich:

- $\mathrm{sd}_{(v,p-1)}$ und $\mathrm{sd}_{(}\, p, w)$ mit $1 < v \leq (p-1)$ und $p \leq w < n$ können nicht in derselben Strategie für π angewendet werden;
- $\mathrm{sd}_{(p,q)}$ mit $1 < p \leq q < n$ kann höchstens einmal in einer Strategie für π angewendet werden.

Für die Sequenzvoraussetzungen liegt ein besonderer und gleichzeitig trivialer Fall vor, sofern kein Element in der Subsequenz $pq \leq_s \pi$ enthalten ist und somit $pq \leq \pi$ gilt. Damit ist die erste Bedingung aus Lemma 6.1 eingehalten. Es gibt demnach keine Sequenzvoraussetzungen zu erfüllen.
Nicht immer ist es möglich, das entsprechende Teilwort pq zu einem Sequenzintervall (p, q) zu erhalten. Dies gelingt nur dann, wenn jedes Element innerhalb der von p und q aufgespannten Subsequenz durch entsprechende Operationen bewegt werden kann, wobei wir deren Operationsvoraussetzungen ebenfalls beachten müssen. Dabei müssen wir zusätzlich Lemma 6.2 berücksichtigen, welches die Beteiligung der Elemente $p, q, (p+1)$ und $(q-1)$ in einer sd-Operation vor Ausführung von $\mathrm{sd}_{(p+1,q-1)}$ verbietet.

Bei den Blockvoraussetzungen gibt es ebenfalls einen einfachen Fall. Dieser liegt vor, wenn für ein Sequenzintervall (p, q) gilt, dass $(p+1) = (q-1)$. Dann besteht der sortierte Block $[(p+1), (q-1)]$ nur aus einem einzige Element und wir sprechen von einem *elementaren* Blockintervall. Für elementare Blockintervalle gibt es offensichtlich keine Blockvoraussetzungen zu erfüllen. Wenn $(p+1) \neq (q-1)$ ist, dann ist es notwendig zu untersuchen, ob Elemente in der von $(p+1)$ und $(q-1)$ aufgespannten Subsequenz $(p+1)(q-1) \leq_s \pi$ vorkommen und wenn ja, welche. Anders als bei den Sequenzintervallen ist es bei den Blockintervallen nicht notwendig jedes Element aus der Subsequenz $(p+1)(q-1) \leq_s \pi$ zu entfernen. Das trifft nur auf Elemente zu, welche später nicht zum sortierten Block $[(p+1), (q-1)]$ gehören. Dies sind genau die Elemente e, für die entweder $e < (p+1)$ oder $e > (q-1)$ gilt. Elemente, welche später zum sortierten Block $[(p+1), (q-1)]$ gehören, müssen nicht zwangsläufig bewegt werden. Dieser Umstand wird später ausführlich behandelt.

Wie wir anhand der Blockintervalle bemerken, scheint es sinnvoll zu sein, Elemente innerhalb der Subsequenz $(p+1)(q-1) \leq_s \pi$ zu kategorisieren. Auch für die Sequenzintervalle und deren Elemente in der Subsequenz $pq \leq_s \pi$ ist es hilfreich, eine Kategorisierung vorzunehmen.

Zu einem beliebigen Sequenzintervall (p, q) einer Permutation π mit $(p, q) \in S_{Op}$ lassen sich folgende Mengen bestimmen:

- $K_{(p,q)} = \{i \mid i < p, p\ i\ q \leq_s \pi\}$
- $M_{(p,q)} = \{i \mid p < i < q, p\ i\ q \leq_s \pi\}$
- $G_{(p,q)} = \{i \mid i > q, p\ i\ q \leq_s \pi\}$
- $L_{(p,q)} = \{i \mid i < p, (p+1)\ i\ (q-1) \leq_s \pi\}$
- $H_{(p,q)} = \{i \mid i > q, (p+1)\ i\ (q-1) \leq_s \pi\}$
- $N_{(p,q)} = \{i \mid (p+1) < i < (q-1), (p+1)\ i\ (q-1) \leq_s \pi\} \cup \{(p+1), (q-1)\}$

Elemente in den Mengen $K_{(p,q)}$, $M_{(p,q)}$ und $G_{(p,q)}$ befinden sich innerhalb der Subsequenz $pq \leq_s \pi$ und müssen entfernt werden, damit das Teilwort pq erhalten wird.
Elemente in den Mengen $L_{(p,q)}$ und $H_{(p,q)}$ gehören nicht zum sortierten Block $[(p+1), (q-1)]$. Dementsprechend müssen diese aus der Subsequenz $(p+1)(q-1) \leq_s \pi$ entfernt werden.
Elemente der Menge $N_{(p,q)}$ gehören in den sortierten Block $[(p+1), (q-1)]$. Diese Elemente müssen dementsprechend nicht aus der Subsequenz $(p+1)(q-1) \leq_s \pi$ entfernt werden.

Beispiel 6.12 Wir betrachten weiterhin die Permutation $\pi = 3\ 9\ 1\ 5\ 7\ 11\ 4\ 6\ 8\ 2\ 10\ 12$ mit

$$S_{Op} = \{(1,11), (2,10), (3,9), (4,6), (4,8), (4,12), \\ (5,7), (5,11), (6,8), (6,12), (7,11), (8,10), (10,12)\}.$$

Die Tabelle gibt die Mengen $K_{(p,q)}$, $M_{(p,q)}$, $G_{(p,q)}$ und $L_{(p,q)}$, $H_{(p,q)}$, $N_{(p,q)}$ für die Sequenzintervalle aus S_{Op} an.

(p,q)	$K_{(p,q)}$	$M_{(p,q)}$	$G_{(p,q)}$	$L_{(p,q)}$	$H_{(p,q)}$	$N_{(p,q)}$
$(1,11)$	$\emptyset$	$\{5,7\}$	$\emptyset$	$\emptyset$	$\emptyset$	$\{2,10\}$
$(2,10)$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\{3,9\}$
$(3,9)$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\{4,6,8\}$
$(4,6)$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\{5\}$
$(4,8)$	$\emptyset$	$\{6\}$	$\emptyset$	$\emptyset$	$\emptyset$	$\{5,7\}$
$(4,12)$	$\{2\}$	$\{6,8,10\}$	$\emptyset$	$\emptyset$	$\emptyset$	$\{5,7,11\}$
$(5,7)$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\{6\}$
$(5,11)$	$\emptyset$	$\{7\}$	$\emptyset$	$\{2\}$	$\emptyset$	$\{6,8,10\}$
$(6,8)$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\{7\}$
$(6,12)$	$\{2\}$	$\{8,10\}$	$\emptyset$	$\emptyset$	$\emptyset$	$\{7,11\}$
$(7,11)$	$\emptyset$	$\emptyset$	$\emptyset$	$\{2\}$	$\emptyset$	$\{8,10\}$
$(8,10)$	$\{2\}$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\{9\}$
$(10,12)$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\{11\}$

Die Elemente aus $K_{(p,q)}$ können nur durch, von den Elementen aus $M_{(p,q)}$ und $G_{(p,q)}$ separierten, sd-Operationen bewegt werden. Gleichfalls können Elemente aus $M_{(p,q)}$ nur getrennt von den Elementen aus $K_{(p,q)}$ und $G_{(p,q)}$ und Elemente aus $G_{(p,q)}$ getrennt von Elementen aus $K_{(p,q)}$ und $M_{(p,q)}$ bewegt werden.
Die Trennung ist ersichtlich, wenn wir uns einen Verstoß vorstellen. Elemente, die durch eine sd-Operation $\mathrm{sd}_{(u,v)}$ bewegt werden, müssen als sortierter Block $[u, v]$ vorliegen. Besteht der

Block $[s,t]$ aber aus Elementen zweier verschiedener Mengen - bspw. $K_{(p,q)}$ und $M_{(p,q)}$ - so kann $[s,t]$ kein sortierter Block sein, denn das Element p kann darin nicht enthalten sein. Aus diesem Grund können auch Elemente aus $L_{(p,q)}$ und $H_{(p,q)}$ nur getrennt voneinander bewegt werden.

Mit S_X bezeichnen wir die Menge der Sequenzintervalle, die zur Bewegung von Elementen aus einer Menge X möglich sind. Jede der Elementmengen $K_{(p,q)}$, $M_{(p,q)}$, $G_{(p,q)}$, $L_{(p,q)}$ und $H_{(p,q)}$ enthält ein minimales Element min und ein maximales Element max. Um ein Element einer dieser Mengen zu bewegen, kommen lediglich Operationen $\mathrm{sd}_{(u,v)}$ mit Sequenzintervall $(u-1, v+1) \in S_{Op}$ in Betracht, für welche $(min-1) \leq (u-1) < (v+1) \leq (max+1)$ gilt.
Zusätzlich müssen wir Lemma 6.2 beachten. Das bedeutet, dass ein Sequenzintervall $(u-1, v+1)$ keines der Elemente p, q und $(p+1), (q-1)$ umschließen darf. Ist dies der Fall, können wir das Teilwort $(u-1)(v+1)$ nicht erhalten, ohne eins der Elemente $p, q, (p+1), (q-1)$ oder $(u-1), (v+1)$ zu bewegen. Dies wäre aber jeweils ein Verstoß gegen Lemma 6.2.
Die Mengen $S_{K_{(p,q)}}$, $S_{M_{(p,q)}}$, $S_{G_{(p,q)}}$ sowie $S_{L_{(p,q)}}$ und $S_{H_{(p,q)}}$ werden wie folgt bestimmt:

$$\begin{aligned} S_{X_{(p,q)}} = \{(u-1, v+1) \mid\ & (u-1, v+1) \in S_{Op},\ u, v \in X_{(p,q)}, \\ & (u-1)\ p\ (v+1) \not\leq_s \pi, p\ (v+1)\ q \not\leq_s \pi, \\ & (p+1)\ (u-1)\ (q-1) \not\leq_s \pi, (p+1)\ (v+1)\ (q-1) \not\leq_s \pi\} \end{aligned}$$

mit $X \in \{K, M, G, L, H\}$.

Jede dieser Mengen von Sequenzintervallen gibt uns genau die Sequenzintervalle (u_1, u_2) an, deren zugehörige Operationen $\mathrm{sd}_{(u_1-1, u_2+1)}$ zur Bewegung der Elemente aus $K_{(p,q)}$, $M_{(p,q)}$, $G_{(p,q)}$, $L_{(p,q)}$ und $H_{(p,q)}$ verwendet werden könnten.

Betrachten wir nun die Elemente der Menge $N_{(p,q)}$. Diese gehören in den sortierten Block $[(p+1), (q-1)]$ und müssen dementsprechend nicht aus der Subsequenz $(p+1)(q-1) \leq_s \pi$ entfernt werden. Dies heißt aber nicht, dass sie nicht Bestandteil einer sd-Operation sein dürfen, bevor $\mathrm{sd}_{(p+1, q-1)}$ anzuwenden ist. Wir suchen also die Operationen, welche die Elemente in die Subsequenz $(p+1)(q-1)$ hineinbewegen, die für die Bildung des sortierten Blockes $[(p+1), (q-1)]$ benötigt werden. Zusätzlich suchen wir die Operationen, welche die Elemente der Menge $N_{(p,q)}$ so ordnen, dass der Block $[(p+1), (q-1)]$ entsteht. Die Menge $S_{N_{(p,q)}}$ bilden wir wie folgt:

$$S_{N_{(p,q)}} = \{(u, v) \mid\ (u, v) \in S_{Op},\ u, v \in N_{(p,q)}\}$$

Beispiel 6.13 Wir betrachten weiterhin die Permutation $\pi = 3\ 9\ 1\ 5\ 7\ 11\ 4\ 6\ 8\ 2\ 10\ 12$ mit

$$\begin{aligned} S_{Op} = \{&(1,11), (2,10), (3,9), (4,6), (4,8), (4,12), \\ &(5,7), (5,11), (6,8), (6,12), (7,11), (8,10), (10,12)\}. \end{aligned}$$

Die Tabelle gibt die Mengen $S_{K_{(p,q)}}$, $S_{M_{(p,q)}}$, $S_{G_{(p,q)}}$ und $S_{L_{(p,q)}}$, $S_{H_{(p,q)}}$, $S_{N_{(p,q)}}$ für die Sequenzintervalle aus S_{Op} an.

(p,q)	$S_{K_{(p,q)}}$	$S_{M_{(p,q)}}$	$S_{G_{(p,q)}}$	$S_{L_{(p,q)}}$	$S_{H_{(p,q)}}$	$S_{N_{(p,q)}}$
$(1,11)$	$\emptyset$	$\{(4,6), (4,8), (6,8)\}$	$\emptyset$	$\emptyset$	$\emptyset$	$\{(2,10)\}$
$(2,10)$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\{(3,9)\}$
$(3,9)$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\{(4,6), (4,8), (6,8)\}$

Fortsetzung Beispiel 6.13

(p,q)	$S_{K_{(p,q)}}$	$S_{M_{(p,q)}}$	$S_{G_{(p,q)}}$	$S_{L_{(p,q)}}$	$S_{H_{(p,q)}}$	$S_{N_{(p,q)}}$
$(4,6)$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$(4,8)$	$\emptyset$	$\{(5,7)\}$	$\emptyset$	$\emptyset$	$\emptyset$	$\{(5,7)\}$
$(4,12)$	$\emptyset$	$\{(5,7),(5,11),(7,11)\}$	$\emptyset$	$\emptyset$	$\emptyset$	$\{(5,7),(5,11),(7,11)\}$
$(5,7)$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$(5,11)$	$\emptyset$	$\{(6,8)\}$	$\emptyset$	$\emptyset$	$\emptyset$	$\{(6,8),(8,10)\}$
$(6,8)$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$(6,12)$	$\emptyset$	$\{(7,11)\}$	$\emptyset$	$\emptyset$	$\emptyset$	$\{(7,11)\}$
$(7,11)$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\{(8,10)\}$
$(8,10)$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$(10,12)$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

Betrachten wir das Sequenzintervall $(4,8)$. In dessen Subsequenz $4\ 8 \leq_s \pi$ liegt das Element 6, welches wir in $M_{(4,8)}$ aufgenommen haben. Es ist leicht zu erkennen, dass wir dieses Element zunächst nur mit der sd-Operation $\mathrm{sd}_{(6,6)}$ bewegen können. Das zugehörige Sequenzintervall $(5,7)$ findet sich dementsprechend in der Menge $S_{M_{(4,8)}} = \{(5,7)\}$. Die Operation $\mathrm{sd}_{(6,6)}$ ist anwendbar, sofern wir deren Voraussetzungen erfüllen können. Das zugehörige Sequenzintervall $(5,7)$ liegt in der Permutation als Teilwort vor und 6 ist ein elementares Blockintervall. So gibt es an dieser Stelle keine Voraussetzungen zu erfüllen. Die Menge $S_{N_{(4,8)}} = \{(5,7)\}$ zeigt uns auf, dass wir die zum Sequenzintervall $(5,7)$ zugehörige Operation ausführen müssen, um den sortierten Block $[5,7]$ zu bilden. Wie wir gerade festgestellt haben, ist die Operation $\mathrm{sd}_{(6,6)}$ ohne Voraussetzungen ausführbar. Daraus ergibt sich, dass auch $\mathrm{sd}_{(5,7)}$ mit Sequenzintervall $(4,8)$ auf $\mathrm{sd}_{(6,6)}(\pi)$ anwendbar ist.
Betrachten wir nun das Sequenzintervall $(8,10)$, unterbrochen durch Element 2 und somit $K_{(8,10)} = \{2\}$. Bei der Suche einer Operation, welche das Element 2 bewegt, sagt uns die Menge $S_{K_{(8,10)}} = \emptyset$, dass es keine solche Operation gibt. In der Tat ist es nicht möglich das Sequenzintervall $(1,3)$ zu erhalten, da $3\ 1 \leq_s \pi$ ist. Es gibt somit keine Möglichkeit die Operation $\mathrm{sd}_(2,2)$ anzuwenden und somit ist es auch nicht möglich, das Element 2 zu bewegen. Dadurch erhalten wir aber nie das Teilwort $8\ 10 \leq \pi$ und können daher niemals die Operation $\mathrm{sd}_{(9,9)}$ anwenden. Das heißt, dass das Sequenzintervall $(8,10)$ nicht in die finale Menge von auf π anwendbaren Operationen aufgenommen wird.

Mit S_{Op}, den aufgestellten Elementmengen $K_{(p,q)}$, $M_{(p,q)}$, $G_{(p,q)}$, $L_{(p,q)}$, $H_{(p,q)}$ und $N_{(p,q)}$, sowie deren zugehörige Mengen von Sequenzintervallen $S_{K_{(p,q)}}$, $S_{M_{(p,q)}}$, $S_{G_{(p,q)}}$, $S_{L_{(p,q)}}$, $S_{H_{(p,q)}}$ und $S_{N_{(p,q)}}$ können wir nun für ein Sequenzintervall (p,q) bestimmen, ob die zugehörige sd-Operation in einer Strategie zu einer Permutation π angewendet werden kann.

SATZ 6.14 Sei π eine nicht vorzeichenbehaftete Permutation und $(p,q) \in S_{Op}$ ein Sequenzintervall mit den zugehörigen Mengen $S_{X_{(p,q)}}$, $X \in A = \{K, M, G, L, H, N\}$. Die Operation $\mathrm{sd}_{(p+1,q-1)}$ ist genau dann auf π anwendbar, notiert mit $\beta(p,q) = true$, wenn für jede der Mengen $S_{X_{(p,q)}}$ eine Partition von Sequenzintervallen $S_{X_{(p,q)}} = T_{X_{(p,q)}} \cup F_{X_{(p,q)}}$ und eine Partition von Elementen $\{(p+1), \ldots, (q-1)\} = D \cup U$ existiert, so dass folgende Bedingungen erfüllt sind:

(i) $\forall (u,v) \in T_{(p,q)}$ ist $(t, z_1) \notin T_{(p,q)}$ mit $t \leq u < z_1 < v$ und $(z_2, w) \notin T_{(p,q)}$ mit $u < z_2 < v \leq w$, wobei
$$T_{(p,q)} = \{(p,q)\} \cup \left(\bigcup_{(r,s) \in T_{X_{(p,q)}}} T_{(r,s)} \right), \text{ mit } (p,q) \notin \left(\bigcup_{(r,s) \in T_{X_{(p,q)}}} T_{(r,s)} \right) \text{ und } X \in A;$$

(ii) $\forall (r,s) \in T_{X_{(p,q)}}$ gilt $\beta(r,s) = true,\ X \in A$;

(iii) $\forall e \in X_{(p,q)}$ gilt $\exists (r,s) \in T_{X_{(p,q)}},\ r < e < s,\ X \in A \backslash N$;

(iv) $\forall (r,s) \in T_{X_{(p,q)}}$ gilt $\mathrm{sd}_{(p+1,q-1)} \circ \Phi(\pi),\ X \in A$, wobei Φ eine Teilstrategie zu π ist und $\mathrm{sd}_{(r+1,s-1)}$ Bestandteil von Φ ist.

(v) $\forall (r,s) \in T_{N_{(p,q)}}$ gilt $(r+1), \ldots, s \in D$ und $\pi|_U$ ist sortiert;

wobei $e, p, q, r, s, t, u, v, w, z_1, z_2 \in \Sigma_n$.

Wir wissen, dass die Menge $S_{X_{(p,q)}}$ genau die Sequenzintervalle enthält, die zur Bewegung der Elemente aus $X_{(p,q)}$ verwendet werden können. Die Partition $S_{X_{(p,q)}} = T_{X_{(p,q)}} \cup F_{X_{(p,q)}}$ gibt an, welche der Sequenzintervalle tatsächlich benutzt werden.

In **Bedingung** (i) wird die Menge $T_{(p,q)}$ ermittelt. Diese Menge setzt sich wie folgt zusammen:

- Das Sequenzintervall (p,q);
- Sequenzintervalle (r,s) der Mengen $T_{X_{(p,q)}}$, wobei es sich um direkt aus (p,q) ermittelte Operationsvoraussetzungen handelt; sowie die Operationsvoraussetzungen zu den Sequenzintervallen (r,s). In dieser Menge darf (p,q) nicht vorkommen, da sonst $\mathrm{sd}_{(p+1,q-1)}$ im Widerspruch zu Lemma 6.11 von sich selbst abhängig wäre.

Die eigentliche Bedingung beruht auf den Beobachtungen von Lemma 6.11 und ist gültig, wenn es keine zu den Sequenzintervallen in $T_{(p,q)}$ gehörigen Operationen gibt, welche sich gegenseitig ausschließen. Zu einem beliebigen Sequenzintervall $(u,v) \in S_{Op}$ kann es verschiedene Ausprägungen der Menge $T_{(u,v)}$ geben. Dies ist der Fall, wenn die direkten Operationsvoraussetzungen, oder auch deren Voraussetzungen, durch verschiedene Kombination von Operationen erfüllt werden können.
Bedingung (ii) verlangt, dass jede zu einem aus der Menge $T_{X_{(p,q)}}$ stammenden Sequenzintervall (r,s) zugehörige Operation $\mathrm{sd}_{(r+1,s-1)}$ tatsächlich angewendet werden kann. Dies ist notwendig, da $\mathrm{sd}_{(r+1,s-1)}$ eine Voraussetzung für $\mathrm{sd}_{(p+1,q-1)}$ darstellt. Wenn $\mathrm{sd}_{(r+1,s-1)}$ nicht anwendbar ist, so wäre demnach $\mathrm{sd}_{(p+1,q-1)}$ niemals anwendbar.
Wie wir zuvor ermittelt haben, sind die Elemente der Mengen $X_{(p,q)},\ X \in A \backslash N$ genau die,

welche bewegt werden müssen, damit sich das Teilwort $pq \leq \pi$ bilden kann und die Subsequenz $(p+1)(q-1) \leq_s \pi$ von später nicht zum sortierten Block $[(p+1),(q-1)]$ gehörenden Elementen befreit wird. **Bedingung** (iii) fordert demnach für jedes Element e der Mengen $X_{(p,q)}$, dass es eine Operation $\mathrm{sd}_{(r+1,s-1)}$ geben muss, welche e bewegt. Das Element e ist Bestandteil dieser Operation, da $r < e < s$ gilt, und e bei einer Ausführung $\mathrm{sd}_{(r+1,s-1)}$ somit zum benötigten sortierten Block $[(r+1),(s-1)]$ gehört.
Bedingung (iv) besagt, dass zunächst alle Operationsvoraussetzungen zu $\mathrm{sd}_{(p+1,q-1)}$ ausgeführt werden, bevor $\mathrm{sd}_{(p+1,q-1)}$ selbst angewendet werden kann.
Über **Bedingung** (v) wird die Partition $\{(p+1),\ldots,(q-1)\} = D \cup U$ bestimmt. Eine Operation $\mathrm{sd}_{u+1,v-1}$ zu einem Sequenzintervall (u,v) ergibt bei ihrer Anwendung den sortierten Block $[u,v]$. Lemma 6.11 besagt, dass einmal sortierte Elemente nicht mehr voneinander getrennt werden. Diese Beobachtung ist die Grundlage für Bedingung (v). Die Menge $T_{N_{(p,q)}}$ enthält die Sequenzintervalle, deren zugehörige Operationen zur Bildung des Blockes $[(p+1),(q-1)]$ verwendet werden. So ergibt sich für eine angewendete Operation $\mathrm{sd}_{(r+1,s-1)}$, dass die Elemente $r,\ldots,s$ als Block $[r,s]$ vorliegen. Die Sortierung der Elemente $(r+1),\ldots,s$ ist demnach gesichert und so können diese Elemente in D aufgenommen werden. Das Element r wird nicht in D aufgenommen, weil über dieses Element die Position des Blockes $[r,s]$ in Relation zu den Elementen $(p+1),\ldots,(q-1)$, welche ebenfalls nicht in D aufgenommen wurden, bestimmt wird.

Es folgt der Beweis zu Satz 6.14.

Beweis Wir zeigen zunächst, dass für eine beliebige ausführbare sd-Operation die Bedingungen $(i)-(v)$ erfüllt sind. Sei π eine beliebige nicht vorzeichenbehaftete Permutation und sei $\mathrm{sd}_{(p+1,q-1)}$ die zugehörige sd-Operation zu einem beliebigen Sequenzintervall (p,q). Sei $\Phi = \mathrm{sd}_{(p+1,q-1)} \circ \mathrm{sd}_{(r_k,s_k)} \circ \ldots \circ \mathrm{sd}_{(r_1,s_1)}$ eine anwendbare Komposition von Operationen auf π, wobei $\mathrm{sd}_{(r_k,s_k)} \circ \ldots \circ \mathrm{sd}_{(r_1,s_1)}$ die Operationsvoraussetzungen zu $\mathrm{sd}_{(p+1,q-1)}$ seien.
Die relativen Positionen der Elemente aus U wechseln nicht die Positionen und so gilt (v). Da $\mathrm{sd}_{p+1,q-1}$ anwendbar ist, müssen alle Operationsvoraussetzungen zuvor erfüllt worden sein, wonach (ii), (iii) und (iv) gilt. Bedingung (i) fasst mit der Menge $T_{(p,q)}$ alle Operationen zusammen, welche ausgeführt werden müssen, damit $\mathrm{sd}_{(p+1,q-1)}$ anwendbar ist und überprüft, ob die Operationen in $T_{(p,q)}$ sich nicht nach Lemma 6.11 gegenseitig ausschließen. Aufgrund dessen, dass Φ anwendbar ist, gibt es offensichtlich keinen Widerspruch gegen (i).

Für die reverse Implikation zeigen wir, dass, wenn die Bedingungen $(i)-(v)$ für eine sd-Operation gelten, diese per Definition anwendbar sein muss.
Sei π eine beliebige nicht vorzeichenbehaftete Permutation und sei $\mathrm{sd}_{(p+1,q-1)}$ die zugehörige sd-Operation zu einem beliebigen Sequenzintervall (p,q) und die Bedingungen $(i)-(v)$ seien gültig.
Die Operation $\mathrm{sd}_{(p+1,q-1)}$ ist nach Lemma 6.1 anwendbar, wenn das Teilwort $pq \leq \pi$ und der sortierte Block $[(p+1),(q-1)]$ in π vorliegen. Nach (iii) existiert für jedes Element e innerhalb der Subsequenz $pq \leq_s \pi$ eine Operation $\mathrm{sd}_{(r+1,s-1)}$ mit $r < e < s$, welche e bewegt. Nach (ii) sind diese Operationen anwendbar, denn es gilt $\beta(r,s) = true$. Da Bedingung (i) gilt, schließen sich keine dieser Operationen nach den Bedingungen aus Lemma 6.11 gegenseitig aus. Aus (iv) folgt, dass jede dieser Sequenzvoraussetzungen ausgeführt wird, bevor $\mathrm{sd}_{(p+1,q-1)}$ angewandt wird. Das Teilwort $pq \leq \pi$ liegt somit vor.
Nach (iii) existiert für jedes Element e mit $e < p$ oder $q < e$ innerhalb der Subsequenz $(p+1)(q-1) \leq_s \pi$ eine Operation $\mathrm{sd}_{(r+1,s-1)}$ mit $r < e < s$, welche e bewegt. Hinzu kommen Operationen zu den Sequenzintervallen der Menge $T_{N_{(p,q)}}$. Nach (ii) sind diese Operationen

anwendbar, da $\beta(r,s) = true$ gilt. Da Bedingung (i) gilt, schließen sich keine dieser Operationen nach den Bedingungen aus Lemma 6.11 gegenseitig aus. Aus (iv) folgt, dass jede dieser Sequenzvoraussetzungen ausgeführt wird, bevor $\mathrm{sd}_{(p+1,q-1)}$ angewandt wird.
Operationen $\mathrm{sd}_{u+1,v-1}$ zu den Sequenzintervallen der Menge $T_{N_{(p,q)}}$ ergeben bei ihrer Anwendung den sortierten Block $[u,v]$. Lemma 6.11 besagt, dass einmal sortierte Elemente nicht mehr voneinander getrennt werden. Demnach können die Elemente $(u+1)\ldots v$ in die Menge D aufgenommen werden, da deren korrekte Reihenfolge gesichert ist. Das Element u wird nicht in D aufgenommen, da über dieses die Position des Blockes ermittelt wird. Nach (v) ist $\pi|_U$ sortiert und somit liegt der sortierte Block $[(p+1),(q-1)]$ vor.
Das heißt, die Bedingungen aus Lemma 6.1 sind erfüllt, wenn $(i)-(v)$ gelten. Somit ist die Operation $\mathrm{sd}_{(p+1,q-1)}$ per Definition anwendbar. Dies bedeutet, dass es keine nicht anwendbare sd-Operation gibt, welche $(i)-(v)$ erfüllt. □

Wir können nun die Menge S_{Ok} von Sequenzintervallen angeben, welche für eine gegebene nicht vorzeichenbehaftete Permutation π anwendbar sind.

$$S_{Ok} = \{(p,q) \mid \ (p,q) \in S_{Op}, \beta(p,q) = true\}$$

Beispiel 6.15 Wir betrachten weiterhin die Permutation $\pi = 3\ 9\ 1\ 5\ 7\ 11\ 4\ 6\ 8\ 2\ 10\ 12$ mit

$$S_{Op} = \{(1,11),(2,10),(3,9),(4,6),(4,8),(4,12),\\ (5,7),(5,11),(6,8),(6,12),(7,11),(8,10),(10,12)\}.$$

Um die Menge S_{Ok} zu bestimmen, muss für jedes Sequenzintervalle in S_{Op} geprüft werden, ob die zugehörige Operation ausführbar ist.

(p,q)	X	$X_{(p,q)}$	$S_{X_{(p,q)}}$	$T_{X_{(p,q)}}$	
$(1,11)$	M	$\{5,7\}$	$\{(4,6),(6,8),(4,8)\}$	$\{(4,6),(6,8)\}$	ja(iii)
				$\{(4,8)\}$	ja(iii)
	N	$\{2,10\}$	$\{(2,10)\}$	$\{(2,10)\}$	ja(v)

Die Tabelle zeigt die Mengen $T_{X_{(p,q)}}$ für das Sequenzintervall (p,q). Die letzte Spalte kennzeichnet, mit ja() das eine Bedingung aus Satz 6.14 erfüllt ist und mit nein() einen Verstoß. Aus der Menge $S_{M_{(1,11)}}$ heraus lassen sich zwei Partitionen $T_{M_{(p,q)}}$ bilden, die Bedingung (iii) aus Satz 6.14 erfüllen. Im weiteren Verlauf müssen beide berücksichtigt werden. Bedingung (v) ist durch die Partition $\{2,\ldots,10\} = D \cup U$ mit $D = \{2,\ldots,10\}$ und $U = \{2\}$ ebenfalls erfüllt, denn natürlich ist $\pi|_2$ sortiert. Um zu überprüfen, ob auch Bedingung (ii) erfüllt ist müssen wir untersuchen, ob $\beta(2,10) = true$ ist und ob entweder $\beta(4,8) = true$ und/oder $\beta(4,6) = true, \beta(6,8) = true$ gilt.

(p,q)	X	$X_{(p,q)}$	$S_{X_{(p,q)}}$	$T_{X_{(p,q)}}$	
$(2,10)$	N	$\{3,9\}$	$\{(3,9)\}$	$\{(3,9)\}$	ja(iii,v)
$(4,6)$	N	$\{5\}$	$\emptyset$	$\emptyset$	ja$(i-v)$
$(6,8)$	N	$\{7\}$	$\emptyset$	$\emptyset$	ja$(i-v)$
$(4,8)$	M	$\{6\}$	$\{(5,7)\}$	$\{(5,7)\}$	ja(iii)
	N	$\{5,7\}$	$\{(5,7)\}$	$\{(5,7)\}$	ja(v)

FORTSETZUNG BEISPIEL 6.15 Wir sehen, dass für die Sequenzintervalle $(4,6)$ und $(6,8)$ und deren zugehörigen Operationen $\mathrm{sd}_{(5,5)}$ und $\mathrm{sd}_{(7,7)}$ keine Operationsvoraussetzungen existieren. Dementsprechend sind die Bedingungen $(i) - (v)$ erfüllt. Bei den Sequenzintervallen $(2,10)$ und $(4,8)$ existieren die Voraussetzungen $(3,9)$ beziehungsweise $(5,7)$. Wir werden auch diese auf Ausführbarkeit hin überprüfen.

(p,q)	X	$X_{(p,q)}$	$S_{X_{(p,q)}}$	$T_{X_{(p,q)}}$	
$(3,9)$	N	$\{4,8\}$	$\{(4,6),(6,8),(4,8)\}$	$\{(4,6),(6,8)\}$ $\{(4,8)\}$	ja(iii,v) ja(iii,v)
$(5,7)$	N	$\{6\}$	$\emptyset$	$\emptyset$	ja$(i-v)$

Die Operation $\mathrm{sd}_{(6,6)}$ zum Sequenzintervall $(5,7)$ gehört ebenfalls zu den anwendbaren Operationen. Für $(3,9)$ gibt es weitere Voraussetzungen, doch wie wir sehen haben wir diese schon untersucht.
Wir können nun zu unserem zu erst untersuchten Sequenzintervall $(1,11)$ überprüfen, ob Bedingung (i) gilt. Dafür stellen wir die Menge $T_{(1,11)}$ auf. Wie wir sehen werden, gibt es dafür vier Varianten.

Variante 1

$$
\begin{aligned}
T_{(1,11)} &= \{(1,11)\} \cup (T_{(2,10)} \cup T_{(4,8)}) \\
T_{(2,10)} &= \{(2,10)\} \cup T_{(3,9)} \\
T_{(3,9)} &= \{(3,9)\} \cup T_{(4,8)} \\
T_{(4,8)} &= \{(4,8)\} \cup T_{(5,7)} \\
T_{(5,7)} &= \{(5,7)\}
\end{aligned}
$$

Variante 2

$$
\begin{aligned}
T_{(1,11)} &= \{(1,11)\} \cup (T_{(2,10)} \cup T_{(4,6)} \cup T_{(6,8)}) \\
T_{(2,10)} &= \{(2,10)\} \cup T_{(3,9)} \\
T_{(3,9)} &= \{(3,9)\} \cup (T_{(4,6)} \cup T_{(6,8)}) \\
T_{(4,6)} &= \{(4,6)\} \\
T_{(6,8)} &= \{(6,8)\}
\end{aligned}
$$

Variante 3

$$
\begin{aligned}
T_{(1,11)} &= \{(1,11)\} \cup (T_{(2,10)} \cup T_{(4,8)}) \\
T_{(2,10)} &= \{(2,10)\} \cup T_{(3,9)} \\
T_{(3,9)} &= \{(3,9)\} \cup (T_{(4,6)} \cup T_{(6,8)}) \\
T_{(4,6)} &= \{(4,6)\} \\
T_{(6,8)} &= \{(6,8)\} \\
T_{(4,8)} &= \{(4,8)\} \cup T_{(5,7)} \\
T_{(5,7)} &= \{(5,7)\}
\end{aligned}
$$

Variante 4

$$
\begin{aligned}
T_{(1,11)} &= \{(1,11)\} \cup (T_{(2,10)} \cup T_{(4,6)} \cup T_{(6,8)}) \\
T_{(2,10)} &= \{(2,10)\} \cup T_{(3,9)} \\
T_{(3,9)} &= \{(3,9)\} \cup T_{(4,8)} \\
T_{(4,8)} &= \{(4,8)\} \cup T_{(5,7)} \\
T_{(5,7)} &= \{(5,7)\} \\
T_{(4,6)} &= \{(4,6)\} \\
T_{(6,8)} &= \{(6,8)\}
\end{aligned}
$$

Bedingung (i) wird nur von Variante 1 und 2 erfüllt. In Variante 3 und 4 schließen sich einige Sequenzintervalle gegenseitig aus. So ist es zum Beispiel niemals möglich die zwei zu $(4,6)$ und $(5,7)$ zugehörigen Operationen $\mathrm{sd}_{(5,5)}$ und $\mathrm{sd}_{(6,6)}$ in einer Strategie auszuführen. Gleiches gilt zum Beispiel auch für die Operationen zu $(4,8)$ und $(6,8)$.
Entscheidend für die Erfüllung von Bedingung (i) zu einem Sequenzintervall (p,q) ist, dass letztlich mindestens eine Variante von $T_{(p,q)}$ existiert, in der sich keine der Sequenzintervalle ausschließen. Dies ist an dieser Stelle mit Variante 1 und 2 für das Sequenzintervall $(1,11)$ gegeben, wodurch wir wissen, dass es zumindest eine (Teil)-Strategie zu π gibt, in der $\mathrm{sd}_{(2,10)}$ ausgeführt wird.

FORTSETZUNG BEISPIEL 6.15 Durch die Betrachtung von $(1, 11)$ konnten wir auch gleichzeitig bestimmen, dass die zu den Sequenzintervallen $(2, 10), (3, 9), (4, 6), (4, 8), (5, 7)$ und $(6, 8)$ gehörigen Operationen ebenfalls auf π anwendbar sind.
Wir betrachten nun die verbliebenen Sequenzintervalle:

(p, q)	X	$X_{(p,q)}$	$S_{X_{(p,q)}}$	$T_{X_{(p,q)}}$	
$(4, 12)$	K	$\{2\}$	$\emptyset$	$\emptyset$	nein(iii)
	M	$\{6, 8, 10\}$	$\{(5, 7), (5, 11), (7, 11)\}$	$\{(5, 7), (7, 11)\}$	ja(iii)
				$\{(5, 11)\}$	ja(iii)
	N	$\{5, 7, 11\}$	$\{(5, 7), (5, 11), (7, 11)\}$	$\{(5, 7), (7, 11)\}$	ja(v)
				$\{(5, 11)\}$	ja(v)
$(5, 11)$	M	$\{7\}$	$\{(6, 8)\}$	$\{(6, 8)\}$	ja(iii)
	L	$\{2\}$	$\emptyset$	$\emptyset$	nein(iii)
	N	$\{6, 8, 10\}$	$\{(6, 8), (8, 10)\}$	$\{(6, 8), (8, 10)\}$	ja(v)
$(6, 12)$	K	$\{2\}$	$\emptyset$	$\emptyset$	nein(iii)
	M	$\{8, 10\}$	$\{(7, 11)\}$	$\{(7, 11)\}$	ja(iii)
	N	$\{7, 11\}$	$\{(7, 11)\}$	$\{(7, 11)\}$	ja(v)
$(7, 11)$	L	$\{2\}$	$\emptyset$	$\emptyset$	nein(iii)
	N	$\{8, 10\}$	$\{(8, 10)\}$	$\{(8, 10)\}$	ja(v)
$(8, 10)$	K	$\{2\}$	$\emptyset$	$\emptyset$	nein(iii)
	N	$\{9\}$	$\emptyset$	$\emptyset$	ja(v)
$(10, 12)$	N	$\{11\}$	$\emptyset$	$\emptyset$	ja$(i - v)$

Mit Ausnahme von $(10, 12)$ gibt es zu jedem Sequenzintervall einen direkten Verstoß gegen Bedingung (iii). Zusätzlich wird in vielen Fällen Bedingung (ii) gebrochen. Zum Beispiel verlangt $(4, 12)$ für die Bildung des sortierten Blocks $[5, 11]$ das $\beta(5, 7) = \beta(7, 11) = \mathit{true}$ oder $\beta(5, 11) = \mathit{true}$ gilt. Dies trifft aber nur auf $(5, 7)$ zu.
Aufgrund dessen, dass die Bedingungen von Satz 6.14 nicht eingehalten werden, ist keine der zu den Sequenzintervallen $(4, 12), (5, 11), (6, 12), (7, 11)$ und $(8, 10)$ zugehörigen sd-Operationen jemals in einer (Teil)-Strategie zu π anwendbar.

Nachdem wir zu einer gegebenen nicht vorzeichenbehafteten Permutation π genau die Operationen ermittelt haben, die in einer Strategie zu π anwendbar sind, können wir auch bestimmen, ob die Permutation sortierbar ist. An dieser Stelle, sei noch einmal die Bildung der Menge $T_{(p,q)}$ zu einem Sequenzintervall (p, q) angegeben, wie sie in Satz 6.14 verwendet wurde, da $T_{(p,q)}$ auch für die Charakterisierung der sd-Sortierbarkeit benötigt wird.

Zu einem Sequenzintervall $(p, q) \in S_{Ok}$ bestimmen wir die Menge $T_{(p,q)}$ mit

$$T_{(p,q)} = \{(p, q)\} \cup \left(\bigcup_{(r,s) \in T_{X_{(p,q)}}} T_{(r,s)} \right) \text{ und } X \in \{K, M, G, L, H, N\},$$

wobei $S_{X_{(p,q)}} = T_{X_{(p,q)}} \cup F_{X_{(p,q)}}$ eine Partition von Sequenzintervallen ist, welche die Bedingungen aus Satz 6.14 erfüllt.

SATZ 6.16 Eine nicht vorzeichenbehaftete Permutation π über Σ_n ist sd-sortierbar, genau dann, wenn eine Teilmenge von Sequenzintervallen $T \subseteq S_{Ok}$ und eine Partition von Elementen $\Sigma_n = D \cup U$ existiert, die folgende Bedingungen erfüllen:

- Wenn $(u, v) \in T$, dann $T_{(u,v)} \subseteq T$;
- $\forall (u, v) \in T$ gilt $(t, z_1) \notin T$ mit $t \leq u < z_1 < q$ und $(z_2, w) \notin T$ mit $u < z_2 < v \leq w$;
- $\forall (u, v) \in T$ gilt $(u + 1), \ldots, v \in D$ und $\pi|_U$ ist sortiert;

wobei $t, u, v, w, z_1, z_2 \in \Sigma_n$.

Bedingung (i) verlangt, dass für jedes Sequenzintervall (u, v) in T, auch die Sequenzintervalle aus $T_{(u,v)}$ in T enthalten sind. Dies sind genau die Sequenzintervalle deren zugehörige Operationen die Operationsvoraussetzungen zu $\mathrm{sd}_{(u+1,v-1)}$ darstellen.
Bedingung (ii) stellt sicher, dass die Sequenzintervalle aus T, beziehungsweise deren zugehörige sd-Operationen, sich nicht gegenseitig ausschließen.
Bedingung (iii) bestimmt die Partition $\Sigma_n = D \cup U$. Eine Operation $\mathrm{sd}_{(p+1,q-1)}$ zu einem Sequenzintervall (p, q) ergibt bei ihrer Anwendung den sortierten Block $[p, q]$. Einmal sortierte Elemente können sich nach Lemma 6.11 nicht wieder trennen. Dieses Verhalten bildet die Grundlage für Bedingung (iii). Jede Operation $\mathrm{sd}_{(u+1,v-1)}$ zu einem Sequenzintervall (u, v) aus T bildet den sortierten Block $[u, v]$. Demnach ist die Reihenfolge der Elemente $u \ldots v$ gesichert. Bis auf das Element u können diese Elemente in D aufgenommen werden. Das Element u wird nicht in D aufgenommen, da es die Position des Blockes $[u, v]$ in Relation zu den Elementen aus U bestimmt. Sind die Elemente aus U eine sortierte Subsequenz innerhalb der Permutation π, so ist demnach die ganze Permutation sortiert.

Beweis Sind die Bedingungen $(i) - (iii)$ zu einer beliebigen vorzeichenbehafteten Permutation π erfüllt, ist leicht zu erkennen, dass π sortierbar ist. Eine Operation $\mathrm{sd}_{(u+1,v-1)}$ zu einem Sequenzintervall aus T bildet bei ihrer Anwendung den sortierten Block $[u, v]$. Einmal sortierte Elemente können sich nach Lemma 6.11 nicht wieder trennen. Die Sortierung der Elemente $u \ldots v$ ist demnach fest. Nur u wird nicht in D aufgenommen und kennzeichnet so die Position des sortierten Blockes $[u, v]$ in Relation zu weiteren Elementen aus U. Nur wenn die Elemente aus U sortiert in π vorliegen, ist π auch insgesamt sortiert. So muss (iii) gelten. Bedingung (ii) verlangt den Einhalt von Lemma 6.11. Aus einer Menge von Operationen, welche diese Bedingung verletzt, kann nie eine anwendbare Strategie gebildet werden. Bedingung (i) verlangt, dass zu jedem in T aufgenommen Sequenzintervall (u, v), auch die Sequenzintervalle aufgenommen werden, deren zugehörige Operationen die Operationsvoraussetzungen zu $\mathrm{sd}_{(u+1,v-1)}$ darstellen. Ein Verstoß gegen (i) hätte zur Folge, dass die Operation $\mathrm{sd}_{(u+1,v-1)}$ nie ausgeführt werden kann, da ihre Voraussetzungen nicht erfüllt werden können. Eine entsprechende Strategie mit $\mathrm{sd}_{(u+1,v-1)}$ als Bestandteil wäre demnach niemals ausführbar.

Für die reverse Implikation zeigen wir, dass jede beliebige sd-sortierbare, nicht vorzeichenbehaftete Permutation π die Bedingungen $(i) - (iii)$ erfüllt. Sei π eine beliebige nicht vorzeichenbehaftete Permutation und $\Phi = \mathrm{sd}_{(u_k+1,v_k-1)} \circ \ldots \circ \mathrm{sd}_{(u_1+1,v_1-1)}(\pi)$ eine sortierende Strategie zu π. Somit gilt $(u_k, v_k), \ldots, (u_1, v_1) \in T$.
Die relativen Positionen der Elemente aus U wechseln nicht die Positionen und so gilt (iii). Da Φ ausführbar ist, gibt es innerhalb von Φ offensichtlich keine sich ausschließenden Operationen. Demnach schließen sich auch die zugehörigen Sequenzintervalle in T nicht gegenseitig

aus und (ii) folgt. Weil Φ anwendbar ist, müssen auch entsprechend die Operationsvoraussetzungen für jede der Operationen $\mathrm{sd}_{(u_k+1,v_k-1)}, \ldots, \mathrm{sd}_{(u_1+1,v_1-1)}$ in Φ enthalten sein. Dies gilt demnach auch für die Sequenzintervalle in T und (i) folgt. □

BEISPIEL 6.17 Wir betrachten weiterhin die Permutation $\pi = 3\ 9\ 1\ 5\ 7\ 11\ 4\ 6\ 8\ 2\ 10\ 12$ mit

$$S_{Ok} = \{(1,11),(2,10),(3,9),(4,6),(4,8),(5,7),(6,8),(10,12)\}$$

Es gibt genau vier Varianten um T zu bilden. Das kommt daher, dass wir zum einen die Wahl haben, ob wir $(4,6)$ und $(6,8)$ oder $(4,8)$ und $(5,7)$ in T aufnehmen und zum anderen zwischen $(1,11)$ und $(10,12)$ entscheiden müssen. Diese Entscheidungen sind aufgrund von Bedingung (ii) notwendig, welche sich gegenseitig ausschließende Sequenzintervalle verbietet. In der nachfolgenden Tabelle sind die vier Bildungsmöglichkeiten aufgelistet und für jede der Varianten wird gezeigt, dass Bedingung (iii) ebenso erfüllt ist.

T	D	U	$\pi\|_U$
$T = \{(1,11),(2,10),(3,9),(4,6),(6,8)\}$	$\{2,\ldots 11\}$	$\{1,12\}$	$\pi\|_U = 1\ 12$ ist sortiert
$T = \{(1,11),(2,10),(3,9),(4,8),(5,7)\}$	$\{2,\ldots 11\}$	$\{1,12\}$	$\pi\|_U = 1\ 12$ ist sortiert
$T = \{(10,12),(2,10),(3,9),(4,6),(6,8)\}$	$\{3,\ldots 12\}$	$\{1,2\}$	$\pi\|_U = 1\ 2$ ist sortiert
$T = \{(10,12),(2,10),(3,9),(4,8),(5,7)\}$	$\{3,\ldots 12\}$	$\{1,2\}$	$\pi\|_U = 1\ 2$ ist sortiert

Die Permutation π ist somit sortierbar. Eine sortierende Strategie wäre zum Beispiel

$$\mathrm{sd}_{(1,11)} \circ \mathrm{sd}_{(2,10)} \circ \mathrm{sd}_{(3,9)} \circ \mathrm{sd}_{(4,6)} \circ \mathrm{sd}_{6,8}(\pi) = 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10\ 11\ 12.$$

Wir schließen das Kapitel mit einem Beispiel zur Charakterisierung sd-sortierbarer Permutationen für eine nicht sortierbare Permutation ab.

BEISPIEL 6.18 Gegeben sei die Permutation $\pi = 1\ 10\ 3\ 9\ 5\ 7\ 11\ 4\ 6\ 2\ 8$. Folgende Sequenzintervalle ergeben sich:

$$\begin{aligned} S = \{&(1,10),(1,3),(1,9),(1,5),(1,7),(2,8),(3,9),(3,5),(3,7),\\ &(3,11),(4,6),(4,8),(5,7),(5,11),(6,8),(7,11),(9,11),(10,12)\} \end{aligned}$$

Davon ausgehend ergibt sich die Menge S_{Op} mit

$$S_{Op} = \{(1,3),(1,9),(2,8),(3,9),(3,5),(3,7),(4,6),(4,8),(5,7),(6,8),(9,11)\}.$$

(p,q)	X	$X_{(p,q)}$	$S_{X_{(p,q)}}$	$T_{X_{(p,q)}}$	
$(1,3)$	G	$\{10\}$	$\{(9,11)\}$	$\{(9,11)\}$	ja(iii), nein(i)
	N	$\{2\}$	$\emptyset$	$\emptyset$	ja(v)
$(1,9)$	M	$\{3\}$	$\emptyset$	$\emptyset$	nein(iii)
	G	$\{10\}$	$\{(9,11)\}$	$\{(9,11)\}$	ja(iii), nein(ii)
	N	$\{2,8\}$	$\{(2,8)\}$	$\{(2,8)\}$	ja(v), nein(ii)
$(2,8)$	H	$\{9\}$	$\emptyset$	$\emptyset$	nein(iii)
	N	$\{3,5,7\}$	$\{(3,5),(5,7),(3,7)\}$	$\{(3,5),(5,7)\}$	ja(v), nein(ii)
				$\{(3,7)\}$	ja(v), nein(ii)

Fortsetzung Beispiel 6.18

(p,q)	X	$X_{(p,q)}$	$S_{X_{(p,q)}}$	$T_{X_{(p,q)}}$	
$(3,9)$	L	$\{2\}$	$\{(1,3)\}$	$\{(1,3)\}$	ja(iii), nein(ii)
	N	$\{4,6,8\}$	$\{(4,6),(6,8),(4,8)\}$	$\{(4,6),(6,8)\}$	ja(v), nein(ii)
				$\{(4,8)\}$	ja(v), nein(ii)
$(3,5)$	G	$\{9\}$	$\emptyset$	$\emptyset$	nein(iii)
	N	$\{4\}$	$\emptyset$	$\emptyset$	ja(v)
$(3,7)$	M	$\{5\}$	$\{(4,6)\}$	$\{(4,6)\}$	ja(iii)
	G	$\{9\}$	$\emptyset$	$\emptyset$	nein(iii)
	N	$\{4,6\}$	$\{(4,6)\}$	$\{(4,6)\}$	ja(v)
$(4,6)$	N	$\{5\}$	$\emptyset$	$\emptyset$	ja$(i-v)$
$(4,8)$	K	$\{2\}$	$\{(1,3)\}$	$\{(1,3)\}$	ja(iii), nein(i)
	M	$\{6\}$	$\{(5,7)\}$	$\{(5,7)\}$	ja(iii)
	N	$\{5,7\}$	$\{(5,7)\}$	$\{(5,7)\}$	ja(v)
$(5,7)$	N	$\{6\}$	$\emptyset$	$\emptyset$	ja$(i-v)$
$(6,8)$	K	$\{2\}$	$\{(1,3)\}$	$\{(1,3)\}$	ja(iii), nein(i)
	N	$\{7\}$	$\emptyset$	$\emptyset$	ja(v)
$(9,11)$	K	$\{5,7\}$	$\{(4,6),(6,8),(4,8)\}$	$\{(4,6),(6,8)\}$	ja(iii), nein(ii)
				$\{(4,8)\}$	ja(iii), nein(i)
	N	$\{10\}$	$\emptyset$	$\emptyset$	ja(v)

Die Sequenzintervalle $(1,3),(9,11)$ und $(4,8)$, sind zyklisch voneinander abhängig und verstoßen damit gegen Bedingung (i) aus Satz. Eine weitere zyklische Abhängigkeit besteht zwischen $(1,3),(9,11)$ und $(6,8)$.
Die Tabelle zeigt, dass lediglich die zu den Sequenzintervallen $(4,6)$ und $(5,7)$ zugehörigen Operationen anwendbar sind. Die Menge anwendbarer Operationen S_{Ok} besteht somit lediglich aus zwei Sequenzintervallen: $S_{Ok} = \{(4,6),(5,7)\}$.
Diese beiden Operationen schließen sich bei Bildung der Menge T nach Satz 6.16 zusätzlich noch gegenseitig aus. Mit $T = (4,6)$ ergibt sich die Partition der Elemente in $D = \{5,6\}$ und $U = \{1,\dots 4,7,\dots 11\}$, wonach $\pi|_U$ unsortiert vorliegt. Dies ist ebenso der Fall bei $T = (5,7)$ mit $D = \{6,7\}$ und $U = \{1,\dots 5,8,\dots 11\}$. Demnach ist die Permutation π nicht sortierbar.

Kapitel 7

Abschließende Betrachtungen

7.1 Zusammenfassung

Nachdem wir uns zunächst mit dem biologischen Hintergrund von Gene Assembly beschäftigt hatten, stellten wir mit dem universellen, dem elementaren und dem einfachen Modell drei Gene Assembly Modelle vor und gaben deren formale Definition an. Es folgte die Untersuchung von Gemeinsamkeiten und Unterschieden der Modelle. Dabei wurden die Modelle bezüglich Vollständigkeit, Konfluenz und sequentieller Komplexität betrachtet. Des Weiteren interessierten wir uns dafür, ob das so genannte Sortierproblem unter dem jeweiligen Modell entscheidbar ist. Wir gaben die Charakterisierung ed-sortierbarer, nicht vorzeichenbehafteter Permutationen an und stellten anschließend eine solche Charakterisierung für die sd-Operation des einfachen Modells auf. Folgende Tabelle gibt eine Übersicht der betrachteten Eigenschaften.

Eigenschaft	Universell	Einfach	Elementar
Vollständigkeit	vollständig	nein	nein
Konfluenz (Resultat)	nein	nein	nein
Konfluenz (Erfolg)	konfluent	konfluent	nein
Konfluenz (Struktur)	konfluent	konfluent	nein
Konstante sequentielle Komplexität	nein	konstant	konstant
Entscheidbarkeit des Sortierproblems	trivial	konfluent bzgl. Erfolg	verbotene Elemente
Charakterisierung von sortierbaren Permutationen	kein Problem	für sd-Operationen vorhanden, sonst offenes Problem	Abhängigkeitsgraph

7.2 Ausblick

Sowohl das einfache als auch das elementare Modell sind nicht in der Lage, jede beliebige vorzeichenbehaftete Permutation zu sortieren. Damit unterscheiden sie sich vom universellen Modell, dem Universalität nachgewiesen wurde. Es stellt sich dadurch die Frage, wie viele

Permutationen der Länge n in den jeweiligen Modellen sortierbar sind. In einer Permutation über Σ_n kann jedes Element an jeder Position stehen und jedes Element kann gekennzeichnet oder nicht gekennzeichnet sein. Um die Frage nach der Anzahl sortierbarer Permutation zu beantworten, müssen demnach $n \times 2^n$ Permutationen betrachten werden.
Für das elementare Modell ist die Entscheidung besonders schwierig, da es für eine Permutation sowohl sortierende als auch nicht sortierende Strategien geben kann. Wir haben in dieser Arbeit die Entscheidungsmethode aus [PR08] vorgestellt. In der Folgearbeit [Rog09] wird eine zweite Möglichkeit vorgeschlagen. Kann eine elementare Strategie von spezieller Form auf eine Permutation π angewandt werden, so ist π mit elementaren Gene Assembly Operationen sortierbar. Um elementar sortierende Strategien zu charakterisieren, wurde die Notation so genannter *fixed integers* und *blocks* für Permutationen eingeführt. Außerdem gelang es die maximale Anzahl sortierender Strategien zu einem Genmuster zu bestimmen. Die Ergebnisse aus [Rog09] könnten unter anderem hilfreich sein, die Zahl der sortierbaren Permutationen einer festen Länge n zu bestimmen oder auch die exakte Anzahl sortierender Strategien zu einer gegebenen Permuation π.
Für das einfache Modell kann durch fortlaufende Ausführung anwendbarer Operationen, bis die Permutation sortiert oder geblockt ist, entschieden werden, ob die Permutation sortierbar ist oder nicht. Dieses Vorgehen benötigt quadratischen Zeitaufwand. Ein offenes Problem besteht darin, ein lineares Verfahren zu entwickeln oder zu beweisen, dass es kein Verfahren mit weniger als quadratischen Zeitaufwand geben kann.
In dieser Arbeit haben wir eine Charakteristik für sd-sortierbare Permutationen aufgestellt. Dies könnte hilfreich sein, um Folgende offene Probleme des einfachen Modells zu lösen:

- Die Aufstellung einer Charakteristik sortierbarer Permutationen.
- Die Bestimmung der Anzahl sortierbarer, nicht vorzeichenbehafteter Permutationen fester Länge n.
- Die Bestimmung der Anzahl sortierbarer Permutationen fester Länge n.
- Die Bestimmung der Anzahl sortierender Strategien zu einer nicht vorzeichenbehafteten Permutation π.
- Die Bestimmung der Anzahl sortierender Strategien zu einer Permutation π.
- Die Entwicklung eines Verfahrens zur Bestimmung, ob eine Permutation sortierbar ist, mit weniger als quadratischen Aufwand.

Abbildungsverzeichnis

Literaturverzeichnis

[Adl94] Leonard M. Adleman. Molecular computation of solutions to combinatorial problems. *Science*, 266(11):1021–1024, 1994.

[Alb03] Bruce Alberts. *Lehrbuch der Molekularen Zellbiologie (German Edition)*. Wiley-VCH, Weinheim, 2 edition, 2003.

[APR07] Artiom Alhazov, Ion Petre, and Vladimir Rogojin. Solutions to computational problems through gene assembly. In Max H. Garzon and Hao Yan, editors, *Proceedings of the 13th International Meeting on DNA Computing*, pages 36–45, Memphis, TN, USA, 2007.

[CCL05] Andre R. O. Calvacanti, Thomas H. Clarke, and Laura F. Landweber. Mds_ies_db: a database of macronuclear and micronuclear genes in spirotrichous ciliates. *Nucleic Acids Research*, 33(1):396–398, 2005.

[EHP+03] Andrzej Ehrenfeucht, Tero Harju, Ion Petre, David M. Prescott, and Grzegorz Rozenberg. *Computation in Living Cells. Gene Assembly in Ciliates*. Springer Verlag, Berlin, Heidelberg, New York, 2003.

[EPPR01a] Andrzej Ehrenfeucht, Ion Petre, David M. Prescott, and Grzegorz Rozenberg. Circularity and other invariants of gene assembly in ciliates. In Masami Ito, Gheorghe Paun, and Sheng Yu, editors, *Words, Semigroups, and Transductions*, pages 81–97. World Scientific, 2001.

[EPPR01b] Andrzej Ehrenfeucht, Ion Petre, David M. Prescott, and Grzegorz Rozenberg. Universal and simple operations for gene assembly in ciliates. In Carlos Martín-Vide and Victor Mitrana, editors, *Where mathematics, computer science, linguistics and biology meet*, pages 329–342, Norwell, MA, USA, 2001. Kluwer Academic Publishers.

[EPR99] Andrzej Ehrenfeucht, David M. Prescott, and Grzegorz Rozenberg. Computational aspects of gene (un)scrambling in ciliates. In Laura F. Landweber and Erik Winfree, editors, *Evolution as Computation*, pages 216–256, Berlin, Heidelberg, New York, 1999. Springer.

[EPR01] Andrzej Ehrenfeucht, David M. Prescott, and Grzegorz Rozenberg. Molecular operations for dna processing in hypotrichous ciliates. *European Journal of Protistology*, 37:241–260, 2001.

[HLPR07] Tero Harju, Chang Li, Ion Petre, and Grzegorz Rozenberg. Complexity measures for gene assembly. 2007.

[HPR03] Tero Harju, Ion Petre, and Grzegorz Rozenberg. *Gene Assembly in Ciliates: Molecular operations*. TUCS Technical Report No 557, Turku Centre for Computer Science, 2003.

[HPR06] Tero Harju, Ion Petre, and Grzegorz Rozenberg. Modelling simple operations for gene assembly. In *Nanotechnology: Science and Computation*, pages 361–373, Berlin, Heidelberg, 2006. Springer Verlag.

[HPRR05] Tero Harju, Ion Petre, Vladimir Rogojin, and Grzegorz Rozenberg. Simple operations for gene assembly. In Lila Kari, editor, *Proceedings of DNA-based computers 11*, volume 3892 of *Lecture Notes in Computer Science*, pages 96–111, Berlin, Heidelberg, 2005. Springer.

[HPRR08] Tero Harju, Ion Petre, Vladimir Rogojin, and Grzegorz Rozenberg. Patterns of simple gene assembly in ciliates. *Discrete Appl. Math.*, 156(14):2581–2597, 2008.

[HS04] Thomas Hinze and Monika Sturm. *Rechnen mit DNA - Eine Einführung in Theorie und Praxis*. Oldenbourg Verlag, München, Wien, 2004.

[IPR07] Thseren-Onolt Ishdorj, Ion Petre, and Vladimir Rogojin. Computational power of intramolecular gene assembly. 2007.

[LK98] Laura F. Landweber and Lila Kari. The evolution of cellular computing: Nature's solution to a computational problem. In *Proceedings of the 4th DIMACS Meeting on DNA-Based Computers*, pages 3–15, Philadelphia, PA, USA, 1998.

[LK99] Laura F. Landweber and Lila Kari. Computational power of gene rearrangement. In Erik Winfree and David K. Gifford, editors, *Proceedings of DNA Bases Computers, V*, pages 207–216, American Mathematical Society, 1999.

[LP06] Miika Langille and Ion Petre. Simple gene assembly is deterministic. *Fundam. Inf.*, 73(1,2):179–190, 2006.

[LP08] Miika Langille and Ion Petre. Sequential vs. parallel complexity in simple gene assembly. *Theor. Comput. Sci.*, 395(1):24–30, 2008.

[LPR08] Miika Langille, Ion Petre, and Vladimir Rogojin. Three models for gene assembly in ciliates: a comparison. In *BIONETICS '08: Proceedings of the 3rd International Conference on Bio-Inspired Models of Network, Information and Computing Sytems*, pages 1–8, ICST, Brussels, Belgium, Belgium, 2008. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[Mun01] Katharina Munk. *Grundstudium Biologie - Genetik*. Spektrum Akademischer Verlag, Berlin, Heidelberg, 2001.

[PER01] David M. Prescott, Andrzej Ehrenfeucht, and Grzegorz Rozenberg. Molecular operations for dna processing in hypotrichous ciliates. *European Journal of Protistology*, 37:241–260, 2001.

[Pet06] Ion Petre. Invariants of gene-assembly in stichotrichous ciliates (invarianten der gen-assemblierung in ciliaten der unterklasse stichotrichia). *it - Information Technology*, 48(3):161–167, 2006.

[PR08] Ion Petre and Vladimir Rogojin. Decision problem for shuffled genes. *Inf. Comput.*, 206(11):1346–1352, 2008.

[Pre94] David M. Prescott. The dna of ciliated protozoa. *Microbiology Rev.*, 58(2):233–267, 1994.

[PRS98] Gheorghe Paun, Grzegorz Rozenberg, and Arto K. Salomaa. *DNA-Computing. New Computing Paradigms.* Springer Verlag, Berlin, Heidelberg, New York, 1998.

[Rog09] Vladimir Rogojin. Successful elementary gene assembly strategies. *International Journal of Foundations of Computer Science*, to appear, 2009.

[Sal78] Arto K. Salomaa. *Formale Sprachen.* Springer Verlag, Berlin, Heidelberg, New York, 1978.

[Sch01] Uwe Schöning. *Theoretische Informatik - kurzgefasst.* Spektrum Akademischer Verlag, Heidelberg, Berlin, 2001.

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbständig, unter Angabe aller Zitate und nur unter Verwendung der angegebenen Literatur angefertigt habe.

Dresden, den 05.06.2009

.....
(Rainer Hausdorf)