Technische Universität Dresden
Institut für Theoretische Informatik
Lehrstuhl für Automatentheorie

Untersuchung der Berechnungsstärke eines gewichteten Insertion-Deletion-Systems

Bachelorarbeit

Monika Roth

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

Beginn: 01.05.2012
Abgabe: 24.07.2012

Inhaltsverzeichnis

1 Einleitung

In der modernen Gesellschaft haben Computer eine große Bedeutung und sind als technisches Hilfsmittel unabdingbar. Mit dem Wort "Computer" wird dabei stets eine elektronische Maschine verbunden. Jedoch stößt dieses konventionelle Rechenkonzept zunehmend sowohl auf physikalische als auch technische Grenzen. In den vergangenen Jahren wurde daher nach alternativen Computing-Konzepten gesucht, die in der Lage sein sollen, Berechnungen in kürzerer Zeit und mit weniger Speicherplatzbedarf durchzuführen. Eine Zielstellung ist beispielsweise das Lösen kombinatorischer Suchprobleme (NP-Probleme) mit angemessenem Ressourcenverbrauch. Das sogenannte *Future-Computing* mit seinen Ausprägungen *Quanten Computing*, *Neural Computing*, *Evolutionary Computing* und *Molecular Computing* steht derzeit im Mittelpunkt des Forschungsinteresses. Das Molecular Computing stellt in diesem Zusammenhang ein sehr aussichtsreiches Konzept dar, welches Prozesse in natürlichen Zellen nachahmen soll und in Bezug auf Nutzbarkeit sowie praktischer Anwendung sehr vielversprechend erscheint. Beim Molecular Computing werden Moleküle als Datenträger bzw. Speichermedium zum Einsatz gebracht und als Eingabedaten bereitgestellt. Die Rechenoperationen sollen dabei durch chemische Reaktionen auf diesen organischen Molekülen nachgebildet werden. Je nach gewähltem Datenträger wird im Molecular Computing zwischen *DNA*-, *RNA*- und *Protein*-Computing unterschieden. In diesem Zusammenhang hat das DNA-Computing, bei dem Desoxyribonucleinsäure (DNA) als Datenträger zum Einsatz kommt, große Bedeutung erlangt, weil es eine Vielzahl nützlicher Eigenschaften aufweist. Da eine Operation gleichzeitig auf mehrere DNA-Stränge wirkt, arbeitet das DNA-Computing massiv datenparallel. Darüber hinaus zeichnet sich das DNA-Computing durch eine hohe Speicherkapazität aus: die Leistungsparameter wie Speicherdichte und -kapazität konventioneller Rechner werden hierbei um mehrere Zehnerpotenzen übertroffen. Außerdem gilt das DNA-Computing als ein umweltfreundliches Konzept, da es ein verschleißfreies, recyclefähiges Konzept darstellt, das einen geringen Energieverbrauch aufweist. Zentrales Forschungsziel im DNA-Computing ist der Aufbau eines universellen DNA-Computers, wobei momentan noch offen ist, wann ein solcher Computer praktisch realisiert werden kann.[1]
Die Insertion-Deletion-Systeme, welche im Mittelpunkt dieser Arbeit stehen, stellen ein abstraktes Modell auf dem Gebiet des DNA-Computings dar. Ein abstraktes Modell bezeichnet in diesem Zusammenhang ein formales Konzept, bei dem die wesentlichen Eigenschaften eines natürlichen Systems formalisiert sowie komplexe Prozesse idealisiert dargestellt und mathematisch betrachtet werden. Insertion-Deletion-Systeme verwenden

1 vgl. [HS04] Kapitel 1

zwei Operationen, welche durch Manipulation von Zeichenketten Berechnungen simulieren: die Insertion-Operation bezeichnet das Einfügen von Zeichen in eine Zeichenkette und die Deletion-Operation das Löschen von Zeichen aus einer Zeichenkette. [2]
In dem einführenden Kapitel 2 werden zunächst wichtige Begriffe aus der Theoretischen Informatik sowie der Molekularbiologie vorgestellt und an kurzen Beispielen verdeutlicht. Das Kapitel 3 bildet den Kern dieser Arbeit. Dort werden zuerst wichtige Begriffe der Insertion-Deletion-Systeme definiert und anschließend eine laborpraktische Implementierung der Insertion- und Deletion-Operation betrachtet. Darüber hinaus wird die Universalität der Insertion-Deletion-Systeme gezeigt sowie untersucht, welchen Einfluss die Modellparameter eines Insertion-Deletion-Systems auf die Universalität des Modells haben. Kapitel 4 fasst die Resultate dieser Arbeit am Schluss noch einmal zusammen.

2 vgl. [Ste06] Kapitel 1

2 Grundlegende Definitionen

In diesem Kapitel sollen ausgewählte Begriffe definiert werden, die zum Verständnis der vorliegenden Arbeit elementar sind. Während im ersten Teil wichtige Definitionen zum Teilgebiet der Theoretischen Informatik vorgestellt werden, sollen im zweiten Teil des Kapitels Begriffe aus der Molekularbiologie definiert werden. Die Definitionen und Einführungen sind [Baa10], [Stu12], [PRS98] sowie [HS04] entnommen und an [Ste06] angelehnt.

2.1 Grundlagen aus der Theoretischen Informatik

2.1.1 Alphabete, Wörter und Sprachen

In der Theoretischen Informatik werden Daten durch Folgen von Zeichen dargestellt. Diese Zeichen stammen aus einem *Alphabet*, das mit Σ bezeichnet wird und als eine endliche nichtleere Menge von Zeichen (z.B. a, b, c, ...) definiert ist. Über diesem Alphabet können (analog zu natürlichen Sprachen) Wörter gebildet werden. Ein *Wort* über dem Alphabet Σ ist als eine endliche Folge von Zeichen $w = a_1a_2...a_n$ mit $a_i \in \Sigma$ definiert. Wörter können miteinander verkettet werden mittels der Operation *Konkatenation*: $w_1 \cdot w_2 := w_1w_2$ (z.B. $ab \cdot bb = abbb$). Mit $|w|$ wird die *Länge* des Wortes w bezeichnet, welche der Anzahl der Zeichen in w entspricht (z.B. $|w| = |ab| = 2$). Das Zeichen ε kennzeichnet das *leere Wort* und besitzt die Länge Null. Σ^* bezeichnet die Menge aller Wörter über dem Alphabet Σ und $\Sigma^+ := \Sigma^* \setminus \{\varepsilon\}$ die Menge aller nichtleeren Wörter über Σ. Eine *formale Sprache* über dem Alphabet Σ bezeichnet in diesem Zusammenhang eine Menge L von Wörtern mit $L \subseteq \Sigma^*$. Beispielsweise ist die Menge aller binären Wörter eine formale Sprache über dem Alphabet $\Sigma = \{0, 1\}$ (z.B. $w = 011110111$).
Seien L, L_1 und L_2 formale Sprachen über dem Alphabet Σ. Dann lassen sich folgende Operationen auf Sprachen definieren:

- $L_1 \cup L_2 = \{w \mid w \in L_1 \text{ oder } w \in L_2\}$ bezeichnet die *Vereinigung* zweier Sprachen.
- $L_1 \cap L_2 = \{w \mid w \in L_1 \text{ und } w \in L_2\}$ bezeichnet den *Schnitt* zweier Sprachen.
- $\overline{L} = \Sigma^* \setminus L$ bezeichnet das *Komplement* einer Sprache.

- $L_1 \otimes L_2 = \{w_1 \cdot w_2 \mid w_1 \in L_1,\ w_2 \in L_2\}$ bezeichnet die Konkatenation (also die Verkettung) zweier Sprachen, wobei $L_1 \otimes \{\varepsilon\} = L_1$ und $L_1 \otimes \emptyset = \emptyset$ gilt. $\{\varepsilon\}$ kennzeichnet dabei die Sprache, welche nur das leere Wort ε enthält und ist das neutrale Element der Konkatenation über Sprachen; $\emptyset$ kennzeichnet die leere Sprache und ist das absorbierende Element der Konkatenation über Sprachen.
- Die Operation *Kleene-Stern* * wird als iterative Konkatenation definiert:

$$L^0 := \{\varepsilon\}$$

$$L^{n+1} := L^n \cdot L$$

$$L^* := \bigcup_{n \geq 0} L^n$$

$$L^+ := \bigcup_{n \geq 1} L^n.$$

2.1.2 Chomsky-Grammatiken

Formale Sprachen umfassen oftmals eine unendliche Anzahl von Wörtern. Daher können sie nicht explizit aufgeschrieben werden, sondern es wird ein anderes Modell zur Beschreibung dieser Sprachen benötigt. Chomsky-Grammatiken sind ein endliches Beschreibungsmodell, die sich für diesen Zweck sehr gut eignen.

Definition 2.1.1 (Chomsky-Grammatik)**.**
Eine *Chomsky-Grammatik* G ist ein Quadrupel $G = (V, \Sigma, P, S)$, wobei die Komponenten folgende Bedeutung haben: V kennzeichnet das Alphabet der Nichtterminalsymbole (Variablenmenge), Σ das Alphabet der Terminalsymbole mit $V \cap \Sigma = \emptyset$, die nichtleere endliche Menge $P \subseteq ((V \cup \Sigma)^* \otimes V \otimes (V \cup \Sigma)^*) \times ((V \cup \Sigma)^*)$ bezeichnet die Regelmenge und $S \in V$ das Startsymbol (Axiom).

Mit Hilfe einer Grammatik können also Wörter über einem Alphabet generiert werden, wobei die Menge P die Regeln definiert, die angewendet werden dürfen, um ein Wort durch ein anderes Wort zu ersetzen (bzw. es aus einem anderen Wort abzuleiten). Formal wird das direkte Ableiten eines Wortes y aus einem Wort x dargestellt als $x \Rightarrow_G y$, wobei $x_1, x_2 \in (V \cup \Sigma)^*$ existieren müssen sowie $u \Rightarrow v \in P$, so dass $x = x_1 u x_2$ und $y = x_1 v x_2$ gilt. Mit $x \Rightarrow_G^n y$ wird entsprechend ausgedrückt, dass y in n Schritten aus x ableitbar ist und mit $x \Rightarrow_G^* y$, dass y aus x ableitbar ist, d.h. es existiert ein $n \geq 0$, so dass $x \Rightarrow_G^n y$ gilt. Die durch eine Grammatik beschriebene Sprache ist nun die Menge der Wörter, die ausgehend vom Startsymbol S durch wiederholtes Ersetzen unter Anwendung der Regeln aus P, generiert werden können.

Definition 2.1.2 (Die von einer Chomsky-Grammatik beschriebene Sprache)**.**
Sei $G = (V, \Sigma, P, S)$ eine Chomsky-Grammatik. Die durch G beschriebene (generierte) Sprache ist definiert durch $L(G) = \{x \in \Sigma^* \mid S \Rightarrow_G^* x\}$.

Um die obigen Definitionen zu verdeutlichen, wird nun ein kurzes Beispiel betrachtet:

Beispiel 2.1.3 (Die von einer Chomsky-Grammatik beschriebene Sprache)**.**
Gegeben sei die Grammatik $G = (V, \Sigma, P, S)$ mit $V = \{S\}, \Sigma = \{a\}$ und $P = \{S \Rightarrow \varepsilon, S \Rightarrow aaS\}$. Die von G beschriebene Sprache ist $L(G) = \{a^{2n} \mid n \in \mathbb{N}\}$.

Die von Grammatiken generierten Sprachen können in verschiedene Sprachklassen unterteilt werden. Die Chomsky-Grammatiken beschreiben genau die Klasse der rekursiv aufzählbaren Sprachen *RE*, mit deren Hilfe alle berechenbaren Funktionen beschrieben werden können. Diese Sprachklasse erzielt also eine Berechnungsstärke äquivalent zur Turingmaschine. Jedes Berechnungsmodell, das eine solche Sprache beschreiben kann, wird universell genannt. Indem die Regeln einer Grammatik eingeschränkt werden, wird die Sprachklasse verkleinert und es lassen sich die folgenden Typen unterscheiden:

Definition 2.1.4 (Typen von Chomsky-Grammatiken, Sprachklassen)**.**
Sei $G = (V, \Sigma, P, S)$ eine Chomsky-Grammatik.

- Jede Grammatik G heißt Grammatik vom *Typ 0*. Eine Typ-0-Grammatik beschreibt die Sprachklasse der *rekursiv aufzählbaren Sprachen (RE)*.
- G heißt Grammatik vom *Typ 1*, falls jede Produktion von G die Form $u_1 A u_2 \Rightarrow u_1 w u_2$ mit $A \in V$, $u_1, u_2, w \in (\Sigma \cup V)^*$ und $|w| \geq 1$ oder $S \Rightarrow \varepsilon$ hat. Ist $S \Rightarrow \varepsilon \in P$, so kommt S nicht auf der rechten Seite einer Produktion vor. Die von einer Typ-1-Grammatik beschriebene Sprache wird *kontextsensitiv (CS)* genannt.
- G heißt Grammatik vom *Typ 2*, falls jede Regel von G die Form $A \Rightarrow w$ hat mit $A \in V$ und $w \in (\Sigma \cup V)^*$. Eine Grammatik vom Typ 2 beschreibt die Sprachklasse der *kontextfreien Sprachen (CF)*.
- G heißt Grammatik vom *Typ 3*, falls jede Regel von G die Form $A \Rightarrow uB$ oder $A \Rightarrow u$ hat, wobei $A, B \in V$ und $u \in \Sigma^*$. Die von einer Typ-3-Grammatik generierte Sprache wird *rechtslinear* oder *regulär (REG)* genannt.

Die Chomsky-Hierarchie setzt die vorgestellten Grammatik-Typen in folgende Beziehung zueinander:
$REG \subset CF \subset CS \subset RE$ bzw. $L(G_3) \subseteq L(G_2),\ L(G_1) \subseteq L(G_0)$.

2.1.3 Normalformen

Grammatiken verschiedenen Typs lassen sich in sogenannte Normalformen transformieren bzw. standardisieren. Nachfolgend sollen drei verschiedene Normalformen für Typ-0-Grammatiken definiert werden, welche in der vorliegenden Arbeit in späteren Abschnitten verwendet werden und für die effiziente Anwendung verschiedener Algorithmen von

großer Bedeutung sind. Darüber hinaus sollen Transformationsvorschriften angegeben werden, die jede beliebige Chomsky-Grammatik $G = (V, \Sigma, P, S)$ effizient in eine äquivalente Grammatik $G' = (V', \Sigma, P', S)$ in Kuroda-, Penttonen- bzw. Geffert-Normalform mit *L(G)=L(G')* überführen können. Die Definitionen sowie der Transformationsalgorithmus zur Umwandlung einer beliebigen Grammatik in eine äquivalente Grammatik in Kuroda-Normalform sind [Stu12] entnommen.

Definition 2.1.5 (Kuroda-Normalform)**.**
Sei $G = (V, \Sigma, P, S)$ eine Chomsky-Grammatik vom Typ 0 sowie $X, Y, U, Z \in V, x \in \Sigma$. G ist in *Kuroda-Normalform*, wenn alle Regeln $\in P$ eine der folgenden Formen besitzen:

- $X \Rightarrow x$,
- $X \Rightarrow \varepsilon$,
- $X \Rightarrow YZ$ oder
- $XY \Rightarrow UZ$.

Mit folgendem Algorithmus lässt sich jede Chomsky-Grammatik $G = (V, \Sigma, P, S)$ effizient in eine äquivalente Grammatik $G' = (V', \Sigma, P', S)$ in Kuroda-Normalform transformieren:
Sei $G = (V, \Sigma, P, S)$ eine Chomsky-Grammatik und $X_i, Y_i \in V$, $x_i \in \Sigma$, $i \in V \setminus \{0\}$.

1. Es wird V':=V und P':=P gesetzt.
2. In jeder Regel $(u, v) \in P'$ mit $length(u) > 1$ und $length(v) > 1$ wird jedes Vorkommen von x_i durch ein neues Nichtterminalsymbol $D_i \in V$ ersetzt und P' die Regel $D_i \Rightarrow x_i$ hinzugefügt.
3. Jede Regel der Form $X_1...X_m \Rightarrow Y_1...Y_n \in P'$ mit $m > 2$, $n < m$ wird durch folgende Regeln $\in P'$ ersetzt:
   $X_1...X_m \Rightarrow Y_1...Y_nE_1...E_p$, $E_j \Rightarrow \varepsilon$ mit $p = m - n$, $j \in \{1, ..., p\}$ und die neuen Nichtterminalsymbole $E_1...E_p$ der Menge V' hinzugefügt.
4. Jede Regel der Form $X_1...X_m \Rightarrow Y_1...Y_n \in P'$ mit $2 \leq m \leq n$, wobei m und n nicht beide gleich 2 sind, wird durch die folgenden Regeln $\in P'$ ersetzt:
   $X_1X_2 \Rightarrow Y_1F_2$, $F_2X_3 \Rightarrow Y_2F_3$, ..., $F_{m-1}X_m \Rightarrow Y_{m-1}F_m$,
   $F_m \Rightarrow Y_mF_{m+1}$, $F_{m+1} \Rightarrow Y_{m+1}F_{m+2}$, ..., $F_{n-1} \Rightarrow Y_{n-1}Y_n$
   und die neuen Nichtterminalsymbole $F_2, ..., F_{n-1}$ der Menge V' hinzugefügt.
5. Jede Regel der Form $X_i \Rightarrow Y_1...Y_k \in P', k \geq 3$ wird durch $k - 1$ Regeln der Form $X_i \Rightarrow Y_1C_2$, $C_2 \Rightarrow Y_2C_3$, ..., $C_{k-1} \Rightarrow Y_{k-1}C_k \in P'$ ersetzt und die neuen Nichtterminalsymbole $C_2, ..., C_{k-1}$ der Menge V' hinzugefügt.
6. Jede Regel der Form $X_i \Rightarrow Y_j \in P'$ wird durch die Regeln $X_i \Rightarrow Y_jE \in P'$ und $E \Rightarrow \varepsilon \in P'$ ersetzt.

Neben der Kuroda-Normalform stellt auch die Penttonen-Normalform eine wichtige Standardform für Typ-0-Grammatiken dar.

Definition 2.1.6 (Penttonen-Normalform)**.**
Sei $G = (V, \Sigma, P, S)$ eine Chomsky-Grammatik vom Typ 0. G ist in *Penttonen-Normalform*, wenn alle Regeln $\in P$ eine der folgenden Formen besitzen:

- $X \Rightarrow x$,
- $X \Rightarrow \varepsilon$,
- $X \Rightarrow YZ$ oder
- $XY \Rightarrow XZ$ mit $X, Y, Z \in V$ und $x \in \Sigma$.

Mit folgendem Algorithmus lässt sich jede Chomsky-Grammatik $G = (V, \Sigma, P, S)$ effizient in eine äquivalente Grammatik $G' = (V', \Sigma, P', S)$ in Penttonen-Normalform transformieren:
Sei $G = (V, \Sigma, P, S)$ eine Chomsky-Grammatik und $X_i, Y_i \in V$, $x_i \in \Sigma$, $i \in V \setminus \{0\}$.

1. Es wird V':=V und P':=P gesetzt.
2. In jeder Regel $(u, v) \in P'$ mit $length(u) > 1$ und $length(v) > 1$ wird jedes Vorkommen von x_i durch ein neues Nichtterminalsymbol $D_i \in V$ ersetzt und P' die Regel $D_i \Rightarrow x_i$ hinzugefügt.
3. Jede Regel der Form $X_1...X_m \Rightarrow Y_1...Y_n \in P'$ mit $m > 2$, $n < m$ wird durch folgende Regeln $\in P'$ ersetzt:
   $X_1...X_m \Rightarrow Y_1...Y_nE_1...E_p$, $E_j \Rightarrow \varepsilon$ mit $p = m - n$, $j \in \{1, ..., p\}$ und die neuen Nichtterminalsymbole $E_1...E_p$ der Menge V' hinzugefügt.
4. Jede Regel der Form $X_1...X_m \Rightarrow Y_1...Y_n \in P'$ mit $2 \leq m \leq n$, wobei m und n nicht beide gleich 2 sind, wird durch die folgenden Regeln $\in P'$ ersetzt:
   $F_1X_1 \Rightarrow F_1Y_1$, $F_2X_2 \Rightarrow F_2Y_2$, ..., $F_mX_m \Rightarrow F_mY_m$,
   $F_i \Rightarrow \varepsilon$ für alle $1 \leq i \leq m$, $Y_m \Rightarrow Y_mF_{m+1}$, $F_{m+1} \Rightarrow Y_{m+1}F_{m+2}$, ..., $F_{n-1} \Rightarrow Y_{n-1}Y_n$
   und die neuen Nichtterminalsymbole $F_1, ..., F_{n-1}$ der Menge V' hinzugefügt.
   Ist $m = n = 2$ und $X_1 = Y_1$, dann tue nichts.
5. Jede Regel der Form $X_i \Rightarrow Y_1...Y_k \in P', k \geq 3$ wird durch $k - 1$ Regeln der Form $X_i \Rightarrow Y_1C_2$, $C_2 \Rightarrow Y_2C_3$, ..., $C_{k-1} \Rightarrow Y_{k-1}C_k \in P'$ ersetzt und die neuen Nichtterminalsymbole $C_2, ..., C_{k-1}$ der Menge V' hinzugefügt.
6. Jede Regel der Form $X_i \Rightarrow Y_j \in P'$ wird durch die Regeln $X_i \Rightarrow Y_jE \in P'$ und $E \Rightarrow \varepsilon \in P'$ ersetzt.

Zuletzt soll nun die Geffert-Normalform definiert werden:

Definition 2.1.7 (Geffert-Normalform)**.**
Jede rekursiv aufzählbare Sprache lässt sich mittels einer Grammatik $G = (V, \Sigma, P, S)$

1. mit $V = \{S, X, Y, Z\}$ und Regeln $\in P$ der Formen
   - $S \Rightarrow uSv$,

- $S \Rightarrow x$

mit $u, v, x \in (\Sigma \cup \{X, Y, Z\})^*$ und *nur einer* nicht-kontextfreien Regel $XYZ \Rightarrow \varepsilon$ beschreiben.

2. mit $V = \{S, X, Y, U, Z\}$ und Regeln $\in P$ der Formen
   - $S \Rightarrow uSv$,
   - $S \Rightarrow x$

   mit $u, v, x \in (\Sigma \cup \{X, Y, U, Z\})^*$ und *nur zwei* nicht-kontextfreien Regel $XY \Rightarrow \varepsilon$, $UZ \Rightarrow \varepsilon$ beschreiben.

Auch hier gilt, dass sich jede Grammatik $G = (V, \Sigma, P, S)$ in eine äquivalente Grammatik in Geffert-Normalform transformieren lässt.

Zur Verdeutlichung der obigen Definitionen wird nun ein kurzes Beispiel ausgeführt:

Beispiel 2.1.8 (Transformation einer Grammatik in eine Normalform)**.**
Gegeben sei erneut die Grammatik $G = (V, \Sigma, P, S)$ mit $V = \{S\}$, $\Sigma = \{a\}$ und $P = \{S \Rightarrow \varepsilon, S \Rightarrow aaS\}$. Die Grammatik $G' = (V', \Sigma, P', S)$ mit $V' = \{S, A, B\}$ und $P' = \{S \Rightarrow \varepsilon, S \Rightarrow AS, A \Rightarrow BB, B \Rightarrow a\}$ beschreibt die korrespondierende Grammatik in Kuroda-Normalform. Darüber hinaus ist G' in Penttonen-Normalform. Durch Transformation in Geffert-Normalform entsteht die Grammatik $G'' = (V, \Sigma, P'', S)$ mit $P'' = \{S \Rightarrow \varepsilon, S \Rightarrow aSa\}$.

2.2 Molekularbiologische Grundlagen

Im Gegensatz zu konventionellen Computingkonzepten, welche auf naturwissenschaftlichen Gesetzmäßigkeiten der Elektrotechnik basieren, liegen dem DNA-Computing molekularbiologische Prinzipien zugrunde. Daher soll im Folgenden eine kurze Einführung in die Grundlagen der Molekularbiologie gegeben werden, wobei zuerst die Struktur der DNA näher erklärt und im Anschluss ein Überblick über die wichtigsten, für diese Arbeit relevanten Operationen auf DNA-Stränge gegeben werden soll. Die Informationen und Abbildungen sind [HS04] sowie [Stu12] entnommen.

2.2.1 Struktur von DNA-Strängen

Ein DNA-Einzelstrang besteht aus einer linearen Folge von *Nucleotiden*, wobei diese Nucleotide über Phosphodiesterbindungen zu einem linearen Strang verknüpft sind. Jedes Nucleotid ist aus der Zuckerart Desoxyribose, der Phosphorsäure und einer Base aufgebaut. Je nachdem, welche Base an das Nucleotid gebunden ist, werden vier mögliche Nucleotid-Arten unterschieden: A (mit der Base Adenin), C (mit der Base Cytosin), G (mit

der Base Guanin) und T (mit der Base Thymin).

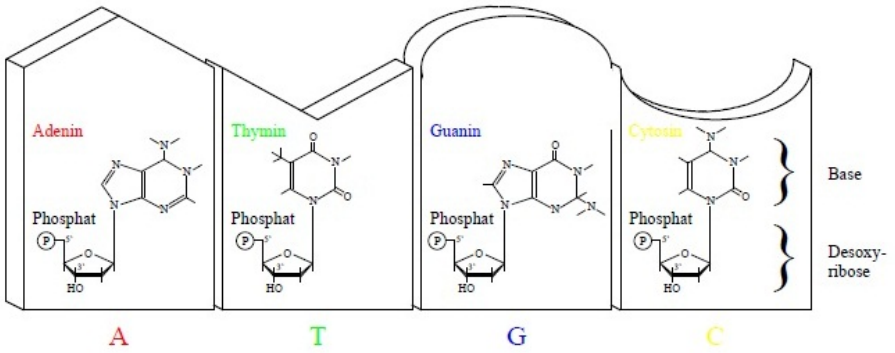

Abbildung 2.1: Die vier verschiedenen Nucleotide und ihre Struktur

Die in einem Nucleotid enthaltene Desoxyribose besitzt eine ringförmige Struktur mit fünf Kohlenstoffatomen, deren Positionen mit 1′ bis 5′ gekennzeichnet sind. An das 3′- und 5′-Kohlenstoffatom kann mittels Phosphodiester-Brücken je eine weitere Desoxyribose angelagert und dadurch ein DNA-Einzelstrang gebildet werden. Dies bedeutet, dass zwei verschiedene Arten von Enden eines DNA-Stranges unterschieden werden können: das *3′-Ende* und das *5′-Ende*. Somit sind DNA-Stränge gerichtet, da sie entweder in 3′-5′-Richtung (als *antisense* bezeichnet) oder in 5′-3′-Richtung (als *sense* bezeichnet) notiert werden können, wobei die letztere Schreibweise üblich ist. Durch die Abfolge der Basen in der Nucleotidkette sowie den Strangendenmarkierungen werden alle Informationen bestimmt, die der entsprechende DNA-Strang kodiert. Demzufolge kann ein DNA-Strang durch seine Nucleotidsequenz und seiner Strangendenmarkierungen eindeutig charakterisiert werden, falls die entsprechende Leserichtung (sense oder antisense) angegeben ist. Dies wird auch als *Primärstruktur* eines DNA-Einzelstranges bezeichnet. Als Strangendenmarkierung werden hierbei chemische Gruppen oder Moleküle verstanden, die spezifisch an 5′- oder 3′-Enden von DNA-Strängen angelagert bzw. abgebaut werden können. Mögliche Strangendenmarkierungen sind P für eine oder mehrere angelagerte Phosphatgruppen bzw. B für eine oder mehrere angelagerte Biotingruppen, wobei die Enden von natürlich vorkommenden DNA-Strängen mit Phosphatgruppen ($-PO_4$) oder Hydroxylgruppen ($-OH$) versehen sind. Da anstelle der Hydroxylgruppe auch leicht eine andere Strangendenmarkierungen angebracht werden kann, wird ein solches Strangende als *frei* bezeichnet. Folgende Definition fasst die obigen Erläuterungen noch einmal knapp zusammen:

Definition 2.2.1 (DNA-Einzelstrang)**.**
Ein *DNA-Einzelstrang* ist eine Sequenz (Abfolge) der Nucleotide A, C, G und T. Die Enden der Nucleotidsequenzen werden mit 5′ und 3′ angegeben. Jedes der beiden Enzelstrangenden trägt eine Strangendenmarkierung.

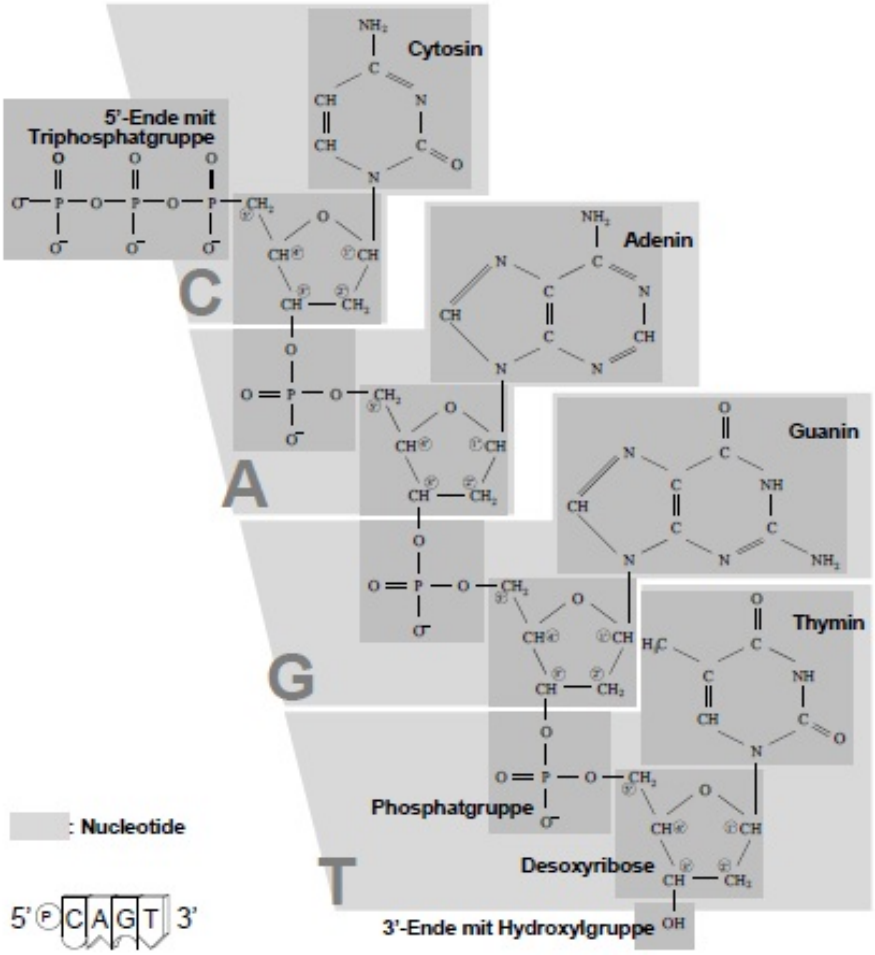

Abbildung 2.2: Primärstruktur des DNA-Einzelstranges 5'P-CAGT-3'

Die obige Abbildung zeigt die Primärstruktur des DNA-Einzelstranges 5'P-CAGT-3', wobei an das 3'-Ende eine Hydroxilgruppe (freies Ende) sowie an das 5'-Ende eine Triphosphatgruppe (mit P bezeichnet) angelagert ist.

Nachdem die Primärstruktur eines DNA-Stranges vorgestellt wurde, kann nun die *Sekundärstruktur* betrachtet werden. DNA-Stränge besitzen unter bestimmten Voraussetzungen die Möglichkeit zur Basenpaarung, wobei eine solche Basenpaarung nur zwischen benachbarten DNA-Einzelsträngen auftreten kann. Zwischen je zwei gegenüberliegenden Basen bilden sich Wasserstoffbrückenbindungen aus, wobei zu beachten ist, dass sich nur komplementäre Basen bzw. Nucleotide paaren können. Das heißt, dass es ausschließlich folgende zwei Paarungsmöglichkeiten gibt: die Basen Adenin und Thymin können sich paaren, wobei zwei Wasserstoffbrückenbindungen entstehen, sowie die Basen Cytosin und Guanin, wobei drei Wasserstoffbrückenbindungen ausgeprägt werden. Folglich ist

die Bindung zwischen Cytosin und Guanin stärker als zwischen Adenin und Thymin, da sie eine Wasserstoffbrücke mehr besitzt.

Definition 2.2.2 (Komplementarität von Nucleotiden)**.**
Die Nucleotide A und T sind zueinander *komplementär*, ebenso die Nucleotide C und G.

Die Basenpaarung zwischen einzelnen Nucleotiden lässt sich auf Nucleotidketten erweitern, so dass sich mehrere DNA-Einzelstränge zu einem DNA-Doppelstrang zusammen lagern können. Die DNA-Einzelstränge müssen dabei *antiparallel* sein, das heißt die einzelsträngige DNA muss entgegengesetzt ausgerichtet sein: ein Strang in 5′-3′ Leserichtung und der andere Strang in 3′-5′-Leserichtung. Zur Erläuterung des Begriffes des DNA-Doppelstranges soll die folgende Definition aufgeführt werden:

Definition 2.2.3 (DNA-Doppelstrang)**.**
DNA, bei der sich mehrere abschnittweise antiparallel-komplementäre DNA-Einzelstränge (bzw. ein DNA-Einzelstrang mit sich selbst) durch Basenpaarung (d.h. durch Wasserstoffbrückenbindungen) miteinander verbunden haben, wird als *doppelsträngig* bezeichnet, ein entsprechender DNA-Strang heißt *DNA-Doppelstrang*.

Auch bei einem DNA-Doppelstrang können zwei verschiedene Arten von äußeren Strangenden unterschieden werden: ein sogenanntes *blunt-Ende* und ein sogenanntes *sticky-Ende*. Während bei einem blunt-Ende kein Einzelstrangüberhang vorhanden ist, können bei einem sticky-Ende entweder 3′- und/oder 5′-Einzelstrangüberhänge auftreten. Folgende Abbildung zeigt einige Beispiele für blunt- bzw. sticky-Enden:

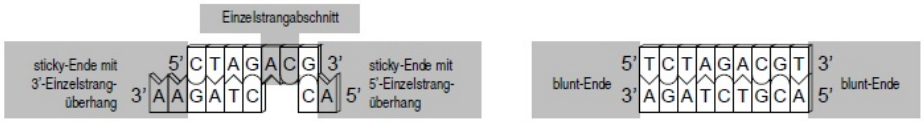

Abbildung 2.3: Beispiele für sticky-Enden, blunt-Enden, Einzelstrangüberhänge und Einzelstrangabschnitte bei DNA-Doppelsträngen

Für das DNA-Computing ist eine Form der Sekundärstruktur besonders interessant, da diese sehr einfach ist und es ermöglicht entsprechende DNA-Doppelstränge in linearisierter Form zu notieren:

Definition 2.2.4 (linearer DNA-Strang)**.**
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 nichtkomplementär-antiparallel ausgerichtete Nucleotide enthält.

Die nachfolgende Abbildung zeigt einige Beispiele für nichtlineare DNA-Stränge:

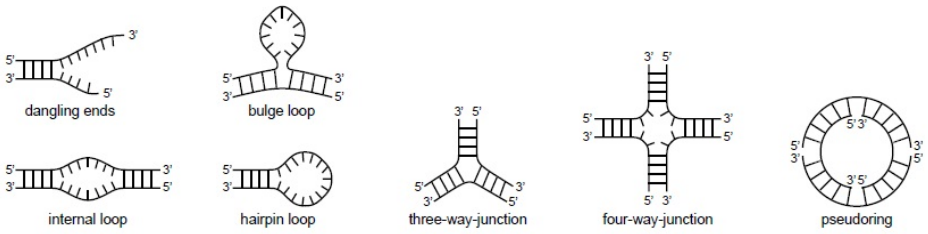

Abbildung 2.4: Beispiele für mögliche Sekundärstrukturen nichtlinearer DNA-Doppelstränge

Neben der Sekundärstruktur besitzt die DNA eine räumliche Struktur, die sogenannte *Tertiärstruktur*, welche die Form einer Doppelhelix besitzt. Der DNA-Doppelstrang bildet räumlich die Gestalt einer Spirale mit gedrehten, gestapelten Sprossen, so dass die Form einer verdrillten Sprossenleiter entsteht. Jede Sprosse wird dabei durch Wasserstoffbrückenbindungen zwischen den entsprechenden Nucleotiden zusammen gehalten. Die beiden antiparallel-komplementären DNA-Einzelstränge winden sich also um eine gemeinsame helikale Achse wie die folgende Abbildung noch einmal graphisch veranschaulicht:

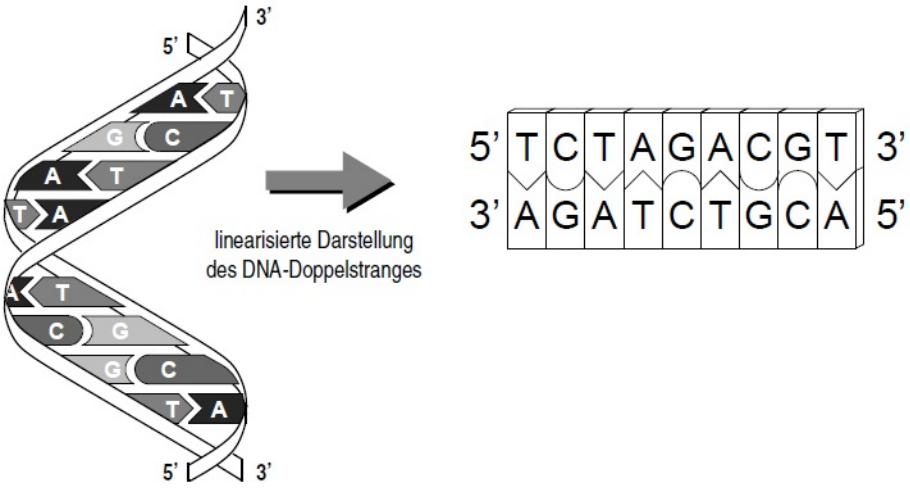

Abbildung 2.5: DNA-Doppelhelix und linearisierte Darstellung eines DNA-Doppelstranges

Diese gewundene Struktur von DNA-Strängen erlaubt eine sehr kompakte Darstellung, woraus resultiert, dass DNA im Unterschied zu konventionellen Speichermedien wenig

Speicherplatz benötigt (ca. 10^{21} Basenpaare pro Liter). Darüber hinaus besitzen DNA-Stränge eine Reihe weiterer nützlicher Eigenschaften, weshalb sie sich für die Nutzung als Datenträger besonders qualifizieren. Im Folgenden soll ein kurzer Überblick über die wichtigsten Eigenschaften von DNA-Strängen gegeben werden:
Dass DNA-Stränge nicht nur in Organismen (*in-vivo*), sondern auch außerhalb von lebenden Zellen im Reagenzglas (*in-vitro*) verarbeitet und aufbewahrt werden können, stellt eine erste wichtige Eigenschaft dar. Unter gewissen Voraussetzungen kann DNA beliebig lang konserviert werden, wodurch sie sich auch als persistentes Speichermedium sehr gut eignet. Diese Informationsspeicherung erfolgt in einer redundanten und verlustsicheren Form, da sich DNA-Stränge sehr leicht in großer Anzahl duplizieren und auf mehrere Reagenzgläser verteilen lassen. Durch die elektrisch negative Ladung von DNA-Strängen, können auch Analysemethoden wie die Gel-Elektrophorese zum Einsatz gebracht werden. Darüber hinaus können DNA-Daten einfach kodiert und visualisiert werden, da DNA-Einzelstränge einfach synthetisiert und deren Sequenzen bestimmt werden können. Die Kodierung und Dekodierung von Daten in DNA-Sequenzen ist in einer einfachen Weise möglich, da die DNA-Stränge - wie oben bereits erwähnt - richtungsbehaftet sind und sich dadurch leicht mit Hilfe von Zeichenketten darstellen lassen. Die Informationsverarbeitung und -speicherung erfolgt in einer verschleißfreien und umweltfreundlichen Form, da DNA-Stränge recyclebar und wiederverwendbar sind. Mit etwa $2 * 10^{19}$ Operationen pro Joule ist außerdem eine sehr energieeffiziente Verarbeitung von DNA-Strängen möglich. DNA kann durch eine Vielzahl spezifisch wirkender Enzyme verarbeitet werden, wodurch eine große Anzahl molekularbiologischer Operationen auf DNA-Strängen möglich ist. Im nachfolgenden Kapitel sollen die wichtigsten, für die vorliegende Arbeit relevanten Operationen näher erläutert werden.

2.2.2 Operationen auf DNA-Strängen

Voraussetzung für die laborpraktische Umsetzung des DNA-Computings ist die Gewinnung bzw. Erzeugung von DNA-Einzelsträngen - die sogenannte *Synthesis*. Hierbei werden zwei prinzipielle Strategien unterschieden: die *DNA-Einzelstrangsynthese* und die *DNA-Isolation aus Trägerorganismen*. Bei Letzterem wird die DNA aus Organismen gewonnen, wobei entsprechend ihrer Herkunft *genomische* DNA, *Plasmid*-DNA und *virale* DNA unterschieden werden kann. Während genomische DNA aus Zellkernen stammt und zumeist aus Blut, Gewebe, Pflanzen, Bakterien, Hefen und Zellkulturen isoliert wird, kommt Plasmid-DNA außerhalb des Zellkerns in Mikroorganismen vor und tritt hauptsächlich in Form von DNA-Doppelsträngen auf. Virale DNA wird zum Großteil aus Bakteriophagen gewonnen, welche in Flüssigkulturen oder auf Agarplatten in Bakterienkulturen vermehrt werden. Bei der DNA-Einzelstrangsynthese hingegen wird die DNA künstlich hergestellt und nicht (wie bei der DNA-Isolation aus Trägerorganismen) aus lebenden Organismen isoliert. Es gibt eine Vielzahl von Methoden zur künstlichen Gewinnung von DNA, jedoch hat sich das Verfahren von Letsinger aus dem Jahr 1975 bis heute durchgesetzt. Hierbei können DNA-Einzelstränge mit bis zu 100 Nucleotiden in großer Reinheit generiert werden.

Die Synthese erfolgt in antisense-Richtung und wird nach dem Prinzip der wachsenden Kette ausgeführt. Dies bedeutet, dass jeder DNA-Einzelstrang Nucleotid für Nucleotid zyklisch aufgebaut wird, wie folgende Abbildung verdeutlicht:

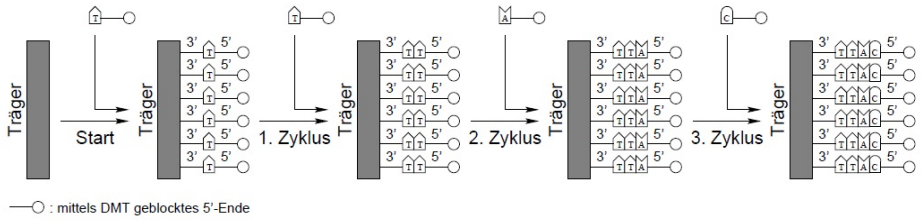

Abbildung 2.6: DNA-Einzelstrangsynthese - Beispiel für das Prinzip der wachsenden Kette

Neben einfachen Operationen wie dem Zusammenführen von mehreren Reagenzglasinhalten in ein gemeinsames Reagenzglas (die sogenannte *Vereinigung*, *Merging* oder *Union*) oder dem Aufteilen eines Reagenzglasinhaltes auf mehrere Reagenzgläser (*Aliquotieren*, *Auteilen* oder *Split* genannt), gibt es auch komplexere Reaktionen, die auf dem Knüpfen bzw. Aufbrechen von Wasserstoffbrückenbindungen basieren. Dabei werden DNA-Einzelstränge in Doppelstränge überführt bzw. umgekehrt. Sinkt die Temperatur unterhalb der DNA-spezifischen Schmelztemperatur (Richtwert: $+94°C$), so werden Wasserstoffbrücken zwischen DNA-Einzelsträngen ausgebildet, so dass Doppelstränge entstehen. Dieser Prozess wird treffenderweise als *Erstarren*, *Annealing*, *Hybridisierung*, *Reassoziation* oder *Renaturierung* bezeichnet.

Definition 2.2.5 (Erstarren, Annealing, Hybridisierung, Reassoziation, Renaturierung)**.** Unter *Annealing* wird das Zusammenlagern von mindestens zwei Molekülen einzelsträngiger DNA (oder eines DNA-Einzelstranges mit sich selbst) an ihren antiparallelkomplementären Stellen zu DNA-Doppelsträngen unter Bildung aller Anlagerungsmöglichkeiten verstanden. Beim Zusammenlagern werden zwischen jeweils zwei antiparallelkomplementär gegenüberliegenden Nucleotiden A und T je zwei, C und G je drei Wasserstoffbrücken ausgebildet, die eine entsprechende Basenpaarung bewirken. Zwei DNA-Stränge bleiben miteinander durch Wasserstoffbrücken verbunden, wenn sie über mindestens 50% der Länge mindestens eines beteiligten DNA-Stranges antiparallel komplementär sind.

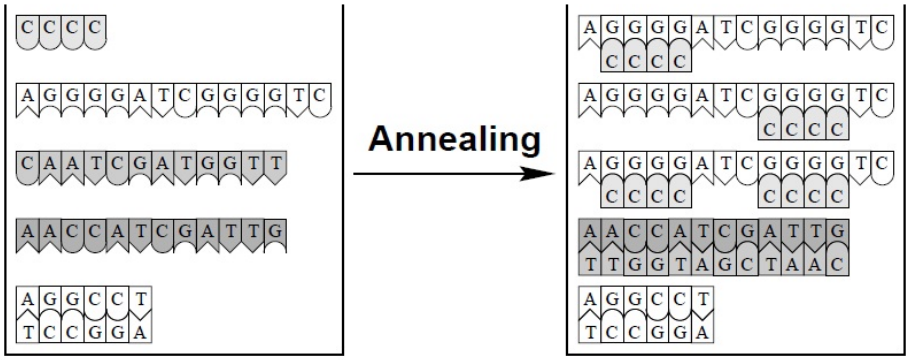

Abbildung 2.7: Beispiel für Annealing

Bei der Umkehrung des erläuterten Annealing-Prozesses werden DNA-Doppelstränge oberhalb der DNA-spezifischen Schmelztemperatur erhitzt, so dass Wasserstoffbrückenbindungen aufgespalten und DNA-Einzelstränge ausgebildet werden. Dieses Verfahren wird als *Schmelzen*, *Denaturierung*, *Melting* oder *Dissoziation* bezeichnet.

Definition 2.2.6 (Schmelzen, Denaturierung, Melting, Dissoziation)**.**
Unter *Melting* wird das Aufspalten von DNA-Doppelsträngen in die zugrundeliegenden DNA-Einzelstränge verstanden.

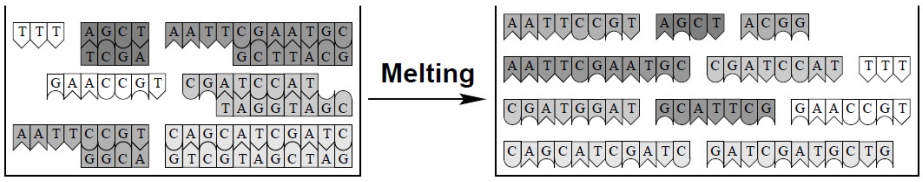

Abbildung 2.8: Beispiel für Melting

Neben Operationen, die auf dem Aufspalten bzw. Knüpfen von Wasserstoffbrücken basieren, gibt es noch eine Reihe enzymatischer Reaktionen. Enzyme, die auch als die "Katalysatoren des Lebens" gelten, haben große Bedeutung in der Molekularbiologie, da sie chemische Umwandlungen in lebenden Zellen fördern und beschleunigen können. Die sogenannten *Restriktionsenzyme* haben beispielsweise die Aufgabe Fremd-DNA, welche in den Organismus eindringt, zu zerstören. Dies geschieht, indem DNA-Doppelstränge aufgeschnitten werden. Dieser Prozess wird daher treffenderweise als *Restriktionsspaltung*, *Schnitt*, *Cut* oder *Digestion* bezeichnet.

Definition 2.2.7 (Restriktionsspaltung, Schnitt, Cut, Digestion)**.**
Unter *Digestion* wird eine molekularbiologische Reaktion verstanden, bei der DNA-Doppelstränge an jedem Vorkommen einer durch das Enzym bestimmten Subsequenz

(Erkennungssequenz) an ebenfalls durch das Enzym bestimmten Spaltstellen geschnitten werden, wodurch bei jedem ausgeführten Schnitt zwei neue 3′-5′-Endenpaare entstehen und auch Einzelstrangüberhänge auftreten können. Die beiden bei jedem ausgeführten Schnitt gebildeten 5′-Enden sind mit einer Phosphatgruppe markiert, die 3′-Enden mit einer Hydroxylgruppe (freie Enden).

Ein Beispiel für ein bekanntes Restriktionsenzym ist ClaI. Die entsprechende Erkennungssequenz ist in der nachfolgenden Abbildung dargestellt, wobei die DNA-Doppelstränge an den mit Pfeilen markierten Stellen aufgespalten werden:

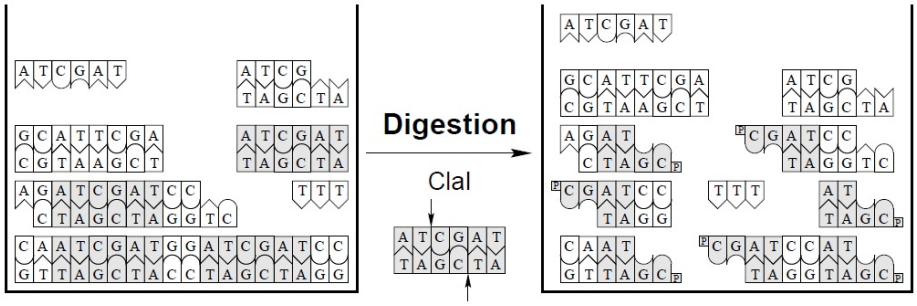

Abbildung 2.9: Beispiel für die Restriktionsspaltung mit dem Enzym ClaI

Eine weitere wichtige Reaktion, die durch Enzyme katalysiert wird, ist die *Polymerisation*. Dabei werden DNA-Doppelstränge mit Einzelstrangüberhang bzw. -abschnitten zu DNA-Doppelsträngen mit blunt-Enden vervollständigt, weswegen diese Operation auch als *Blunting* bezeichnet wird. Das verwendete Enzym ist die Polymerase.

Definition 2.2.8 (Polymerisation, Blunting)**.**
Unter *Polymerisation* wird eine molekularbiologische Reaktion verstanden, bei der 5′-Überhänge von DNA-Doppelsträngen zu blunt-Enden aufgefüllt sowie entsprechende 3′-Überhänge zu blunt-Enden abgebaut werden. Die aufzufüllenden 3′-Enden an 5′-Überhängen müssen mit Hydroxylgruppen markiert sein. DNA-Einzelstränge werden vollständig abgebaut. Die Reaktion wird durch eine geeignete DNA-Polymerase mit 3′-5′-Exonuclease-Aktivität katalysiert.

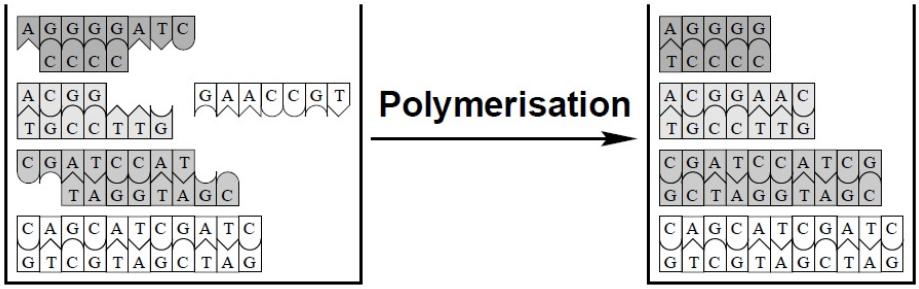

Abbildung 2.10: Beispiel für Polymerisation

Die Operation Polymerisation kann auch iteriert angewendet werden: Durch wiederholte Ausführung der Operationsfolge Melting, Annealing und Polymerisation ist es möglich DNA-Doppelstränge millionenfach zu duplizieren. Diese Reaktionsfolge wird als *Polymerase-Kettenreaktion* bezeichnet. Sie ist für die weiteren Betrachtungen in der vorliegende Arbeit jedoch nicht weiter relevant. Darüber hinaus sind die Operationen Ligation, bei welcher DNA-Doppelstränge fortgesetzt miteinander verkettet werden, und die Gel-Elektropherese, welche zur DNA-Längenbestimmung und -seperation verwendet wird, erwähnenswert.

3 Insertion-Deletion-Systeme

Insertion-Deletion-Systeme - oder kurz: InsDel-Systeme - sind formale Berechnungsmodelle, die auf zwei Operationen basieren: die kontextsensitive Insertion (Einfügung) und die kontextsensitive Deletion (Löschung). Diese Operationen lassen sich als Rekombination in der Natur wiederfinden, indem gezielt bestimmte DNA-Abschnitte in einem DNA-Strang eingefügt bzw. aus diesem entfernt werden können. Vor allem in Form von Punktmutationen in lebenden Zellen treten Insertion- bzw. Deletion-Operationen in lebenden Organismen auf. Die nachfolgenden Definitionen und Betrachtungen sind [Ver10], [Ste06] sowie [HS04] entnommen.

3.1 Definitionen

In diesem Abschnitt sollen zunächst die wichtigsten Definitionen in Bezug auf das vorliegende Thema vorgestellt werden, die für das Grundverständnis dieser Arbeit wichtig sind.

Definition 3.1.1 (Insertion-Deletion-System)**.**
Ein *Insertion/Deletion-System* ist ein Quadrupel $\gamma = (V, \Sigma, A, R)$, wobei V ein beliebiges Alphabet, $\Sigma \subseteq V$ das Alphabet der Terminalsymbole und die endliche Sprache $A \subset V^*$ die Menge der Axiome kennzeichnet. R bezeichnet die endliche Menge von Tripeln der Form $(u\ ,\ \alpha/\beta\ ,\ v)$ mit $u, v \in V^*, (\alpha, \beta) \in (V^+ \times \{\varepsilon\}) \cup (\{\varepsilon\} \times V^+)$ und beschreibt die Insertion-Deletion-Regeln des Systems.

Die Regel $(u\ ,\ \varepsilon/\beta\ ,\ v)$ bedeutet, dass die Sequenz β zwischen den Kontexten u und v eingefügt werden kann - sie beschreibt also eine Insertion-Operation.

Definition 3.1.2 (Insertion-Operation)**.**
Sei V ein Alphabet und $s, \beta \in \Sigma^*$. Dann ist die *Insertion* von β in s definiert als

$$s \leftarrow \beta = \{x\beta y \mid xy = s,\ x, y \in \Sigma^*\}.$$

Sei $(u, v) \in \Sigma^* \times \Sigma^*$ ein *Kontext*. Dann ist die *(u,v)-kontextuelle Insertion* von β in s definiert als

$$s \leftarrow_{(u,v)} \beta = \{xu\beta vy \mid s = xuvy,\ x, y \in \Sigma^*\}.$$

Analog dazu bedeutet die Regel $(u, \alpha/\varepsilon, v)$, dass die Sequenz α zwischen den Kontexten u und v entfernt werden kann - sie beschreibt also eine Deletion-Operation.

Definition 3.1.3 (Deletion-Operation)**.**
Sei V ein Alphabet und $s, \alpha \in \Sigma^*$. Die *Deletion* von α aus s ist definiert als

$$s \to \alpha = \{xy \mid s = x\alpha y,\ x, y \in \Sigma^*\}.$$

Sei $(u, v) \in \Sigma^* \times \Sigma^*$ ein *Kontext*. Dann ist die *(u,v)-kontextuelle Deletion* von α aus s definiert als

$$s \to_{(u,v)} \alpha = \{xuvy \mid s = xu\alpha vy,\ x, y \in \Sigma^*\}.$$

Nun wird ein Beispiel betrachtet, um die Operationen Insertion und Deletion zu verdeutlichen:

Beispiel 3.1.4 (Insertion- und Deletion-Operation)**.**
Gegeben sei das Alphabet $\Sigma = \{a, b, x\}$ und die Sequenz $s = ababbabb$.

1. Zuerst soll die Sequenz $\beta = xxx$ in s unter Berücksichtigung des Kontextes (b, b) eingefügt werden: $s \leftarrow_{(b,b)} \beta = \{ababxxxbabb, ababbabxxxb\}$. Da der Kontext (b, b) zweimal in s vorkommt, ergeben sich im Resultat zwei neue Wörter. Im ersten Wort wurde in das erste Vorkommen des Kontextes (b, b) die Sequenz $\beta = xxx$ eingefügt, im zweiten resultierenden Wort analog dazu in das entsprechend zweite Vorkommen des Kontextes (b, b).
2. Nun soll die Sequenz $\alpha = bb$ aus s unter Berücksichtigung des Kontextes (a, a) gelöscht werden: $s \to_{(a,a)} \alpha = \{abaabb\}$. Zu beachten ist hierbei, dass die Sequenz α in s zweimal vorkommt, jedoch nur aus dem ersten Vorkommen gelöscht wird. Dies resultiert daraus, dass sich das zweite Vorkommen von α in s nicht im geforderten Kontext (a, a) befindet.

Kann eine Sequenz s′ aus einer Sequenz s mit Hilfe einer Regel aus R abgeleitet werden, so wird dieser Sachverhalt mit $s \Rightarrow_\gamma s'$ notiert. Dies bedeutet, dass s′ aus s durch Anwendung einer Insertion- bzw. Deletion-Regel entstanden ist und wird auch als *Regelanwendung* oder *Produktion* bezeichnet.

Definition 3.1.5 (Ableitungsschritt eines Insertion-Deletion-Systems)**.**
Sei $\gamma = (V, \Sigma, A, R)$ ein Insertion-Deletion-System und $s, s' \in \Sigma^*$ sowie die Kontexte $u, v \in V^*$. Ein Ableitungsschritt von s nach s′ wird durch die Ableitungsrelation $\Rightarrow_\gamma \subset V^* \times V^*$ wie folgt definiert:

$$s \Rightarrow_\gamma s' = \{(s, s') \mid \begin{cases} s = s_1 uvs_2,\ s' = s_1 u\beta vs_2,\ (u, \varepsilon/\beta, v) \in R,\ s, s' \in V^*\} \\ s = s_1 u\alpha vs_2,\ s' = s_1 uvs_2,\ (u, \alpha/\varepsilon, v) \in R,\ s, s' \in V^*\} \end{cases}$$

$\Rightarrow_\gamma^+$ bezeichnet die reflexive und $\Rightarrow_\gamma^*$ die reflexive, transitive Hülle der Relation $\Rightarrow_\gamma$. Während $s \Rightarrow_\gamma s'$ kennzeichnet, dass s' aus s durch Anwendung von genau einer Insertion- bzw. Deletion-Regel entstanden ist, kann $s \Rightarrow_\gamma^* s'$ mehrere Ableitungsschritte (also die Anwendung mehrerer Insertion- und/oder Deletion-Regeln) einschließen.

Definition 3.1.6 (Die durch ein Insertion-Deletion-System generierte Sprache)**.**
Sei $\gamma = (V, \Sigma, A, R)$ ein Insertion-Deletion-System. Die durch γ generierte Sprache $L(\gamma)$ wird definiert durch $L(\gamma) = \{w \in \Sigma^* \mid s \Rightarrow_\gamma^* w,\ s \in A\}$.

Darüber hinaus lässt sich die Komplexität eines Insertion-Deletion-Systems durch einen Vektor $(n, n, n'; p, q, q')$ beschreiben.

Definition 3.1.7 (Größe eines Insertion-Deletion-System)**.**
Sei $\gamma = (V, \Sigma, A, R)$ ein Insertion-Deletion-System. Der Vektor $(n, n, n'; p, q, q')$ beschreibt die *Größe (size)* dieses Systems, wobei seine Komponenten wie folgt definiert sind:

$$n = max\{|\beta|,\ (u\ ,\ \varepsilon/\beta\ ,\ v) \in R\}, \qquad p = max\{|\alpha|,\ (u\ ,\ \alpha/\varepsilon\ ,\ v) \in R\},$$

$$m = max\{|u|,\ (u\ ,\ \varepsilon/\beta\ ,\ v) \in R\}, \qquad q = max\{|u|,\ (u\ ,\ \alpha/\varepsilon\ ,\ v) \in R\},$$

$$m' = max\{|v|,\ (u\ ,\ \varepsilon/\beta\ ,\ v) \in R\}, \qquad q' = max\{|v|,\ (u\ ,\ \alpha/\varepsilon\ ,\ v) \in R\}.$$

Mit $INS_n^{m,m'}DEL_p^{q,q'}$ wird die korrespondierende Sprachklasse des Insertion-Deletion-Systems bezeichnet. Die *totale Größe (total size)* des Insertion-Deletion-Systems wird als Summe aller Komponenten definiert: $\psi = n + m + m' + p + q + q'$.

Wird die Komponente m und/oder q (bzw. m' und/oder q') mit 0 spezifiziert, während m' und/oder q' (bzw. m und/oder q) ungleich 0 ist, so wird die korrespondierende Sprachklasse als einseitig-kontextuell bezeichnet. Darüber hinaus kennzeichnet * , dass einer der Parameter n, m, m', p, q, q' nicht spezifiziert ist. $INS_*^{0,0}DEL_*^{0,0}$ beschreibt beispielsweise die Sprachklasse der kontextfreien Insertion-Deletion-Systeme.

In früherer Literatur, beispielsweise in [PRS98], wird ein anderes Komplexitätsmaß verwendet: das *Gewicht (weight)* eines Insertion-Deletion-Systems. Das Gewicht wird als Quadtrupel $(n, \tilde{m}, p, \tilde{q})$ definiert, wobei $\tilde{m} = max\{m, m'\}$ und $\tilde{q} = max\{q, q'\}$:

Definition 3.1.8 (Gewicht eines Insertion-Deletion-System)**.**
Ein Insertion-Deletion-System $\gamma = (V, \Sigma, A, R)$ ist vom *Gewicht* $(n, m; p, q)$ genau dann, wenn

$$\begin{aligned} n &= max\{|\beta|,\ (u\ ,\ \varepsilon/\beta\ ,\ v) \in R\}, \\ m &= max\{|u|,\ (u\ ,\ \varepsilon/\beta\ ,\ v) \in R \vee (v\ ,\ \varepsilon/\beta\ ,\ u) \in R\}, \\ p &= max\{|\alpha|,\ (u\ ,\ \alpha/\varepsilon\ ,\ v) \in R\}, \\ q &= max\{|u|,\ (u\ ,\ \alpha/\varepsilon\ ,\ v) \in R \vee (v\ ,\ \alpha/\varepsilon\ ,\ u) \in R\}. \end{aligned}$$

Auch hier wird mit $INS_n^m DEL_p^q$ die korrespondierende Sprachklasse des Insertion-Deletion-Systems bezeichnet.

Bevor eine mögliche laborpraktische Implementierung der Insertion- und Deletion-Operation diskutiert werden soll, wird zuvor noch ein kurzes Beispiel betrachtet, welches die Arbeitsweise eines Insertion-Deletion-Systems verdeutlichen soll:

Beispiel 3.1.9 (Insertion-Deletion-System)**.**
Gegeben sei das Insertion-Deletion-System $\gamma = (V, \Sigma, A, R)$ mit $V = \{S, a\}$, $\Sigma = \{a\}$, $A = \{S\}$ und $R = \{(S\ ,\ \varepsilon/aa\ ,\ \varepsilon), (\varepsilon\ ,\ S/\varepsilon\ ,\ \varepsilon)\}$. Dieses InsDel-System generiert ebenfalls die Sprache $L(G) = \{a^{2n} \mid n \in \mathbb{N}\}$ und hat das Gewicht (2, 1; 1, 0).

3.2 Laborpraktische Implementierung

Die Insertion- und die Deletion-Operation können mit DNA-Einzelsträngen recht einfach simuliert werden und lassen sich daher mit geringem Aufwand im Labor implementieren. Durch diese Einfachheit ist das Modell der Insertion-Deletion-Systeme besonders interessant für die Molekularbiologie geworden und stellt derzeit einen wesentlichen Forschungsschwerpunkt im Bereich des DNA-Computings dar.
Zunächst wird eine mögliche laborpraktische Implementierung der kontextsensitiven Insertion-Operation betrachtet, wobei die Sequenz y in den Kontext (u, v) eingefügt werden soll: Zu Beginn liege in einem Reagenzglas ein DNA-Einzelstrang der Form 5′-x_1uvx_2z-3′ vor, wobei $x_1, u, v, x_2, z \in \{A, C, G, T\}^*$. Darüber hinaus wird für die (u, v)-kontextuelle Insertion der Sequenz y ein weiterer DNA-Einzelstrag 3′-$\bar{u}\bar{y}\bar{v}$-5′ benötigt, wobei $\bar{u}, \bar{v}$ die Watson-Crick-Komplemente von u bzw. v darstellen und $\bar{y}$ das Watson-Crick-Komplement einer neuen Sequenz y entspricht (vgl. Abb. 3.1 (a)). Wird angestrebt, dass zwischen allen möglichen Vorkommen des Kontextes (u, v) im ursprünglichen DNA-Strang die Sequenz y eingefügt werden soll, so muss sowohl der Ausgangsstrang als auch der Strang 3′-$\bar{u}\bar{y}\bar{v}$-5′ in entsprechend hoher Anzahl im Reagenzglas vorhanden sein. Wurde der DNA-Strang in entsprechender Anzahl in das Reagenzglas gegeben, so können die beiden komplementären DNA-Stränge naturieren. Dabei lagert sich $\bar{u}$ an u, $\bar{v}$ an v und $\bar{y}$ bildet zwischen $\bar{u}$ und $\bar{v}$ eine Schleife (vgl. Abb. 3.1 (b)). Da die Sequenz $\bar{y}$ kein entsprechendes Watson-Crick-Komplement findet und deswegen eine Schleife zwischen den Sequenzen $\bar{u}$ und $\bar{v}$ entsteht, wird die Ausprägung eines linearen Doppelstrangs zunächst verhindert. Diese Form der Naturierung wird deswegen auch als *unpassendes Annealing (mismatching annealing)* bezeichnet. Wird nun ein passendes Restriktionsenzym in das Reagenzglas gegeben, so wird der DNA-Strang zwischen den Sequenzen u und v aufgespalten und es entsteht eine Struktur wie sie in Abb. 3.1 (c) dargestellt ist. Anschließend wird die Sequenz $\bar{z}$ als Primer in das Reagenzglas gegeben und der DNA-Strang kann mittels Polymerisation zu einem linearen Doppelstrang vervollständigt werden (vgl. Abb. 3.1 (d)). Durch anschließende Denaturierung wird der Doppelstrang in zwei Einzelstränge aufgespalten, so dass das Ergebnis der (u, v)-kontextuellen Insertion (der DNA-Einzelstrang 5′-x_1uyvx_2z-3′ wie in Abb. 3.1 (e) dargestellt) sowie dessen Watson-Crick-Komplement (der Strang 3′-$\bar{u}\bar{y}\bar{v}$-5′) in entsprechender Anzahl im Reagenzglas vorhanden sind.

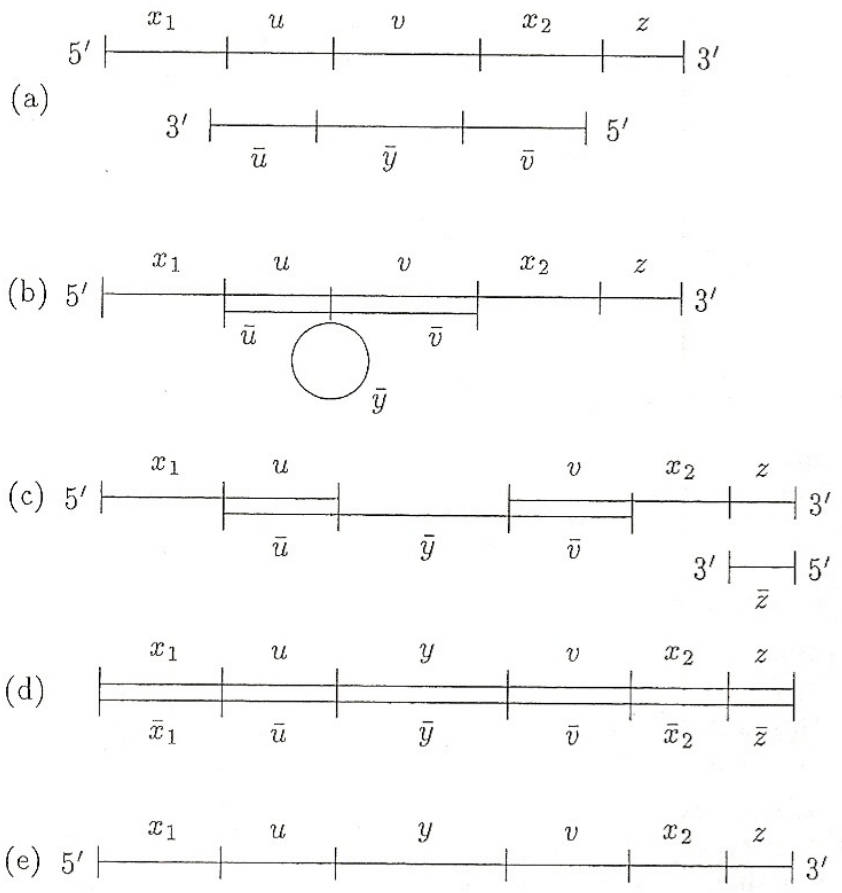

Abbildung 3.1: Insertion der Sequenz y in einen DNA-Einzelstrang

Analog dazu lässt sich auch die kontextsensitive Deletion-Operation laborpraktisch implementieren. Hier liege zu Beginn der DNA-Strang 5′-x_1uyvx_2z-3′ im Reagenzglas vor, wobei die Sequenz y aus dem Kontext (u, v) gelöscht werden soll. Nun wird der DNA-Einzelstrang 3′-$\bar{u}\bar{v}$-5′ in das Reagenzglas gegeben (vgl. Abb. 3.2 (a)), so dass sich nach einem unpassenden Annealing die entsprechende Struktur in Abb. 3.2 (b) ausbildet. Durch Zugabe eines Restriktionsenzyms kann die entstandene Schleife y mittels Restriktionsspaltung entfernt werden. Anschließend bildet sich durch Hinzufügen eines Primers ein vollständiger DNA-Doppelstrang (vgl. Abb. 3.2 (c)) aus, welcher anschließend denaturiert wird. Dadurch entstehen die resultierenden DNA-Einzelstränge 5′-x_1uvx_2z-3′ (das Deletion-Ergebnis wie in Abb. 3.2 (d) gezeigt) sowie das zugehörige Watson-Crick-Komplement 3′-$\bar{u}\bar{v}$-5′. Auch hier werden die DNA-Stränge (der Ausgangsstrang sowie das Watson-Crick-Komplement $\bar{u}\bar{v}$) in entsprechend hoher Anzahl benötigt, wenn die Deletion-Operation mehrfach durchgeführt werden soll.

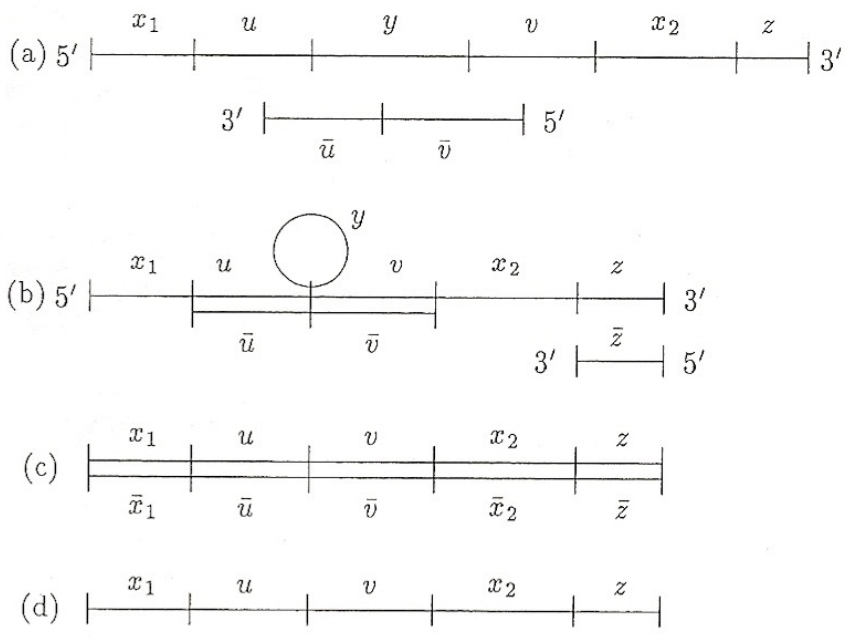

Abbildung 3.2: Deletion der Sequenz y aus einem DNA-Einzelstrang

3.3 Universalität der Insertion-Deletion-Systeme

In diesem Abschnitt wird sich mit der Frage beschäftigt, inwieweit Insertion-Deletion-Systeme universell sind, d.h. welche Eigenschaften InsDel-Systeme aufweisen müssen, damit sie alle Sprachen der Klasse RE generieren können. Interessant bei dieser Betrachtung sind die Gewichte eines InsDel-Systems. Genauer gesagt, hat die Frage, welche minimalen Gewichte das InsDel-System aufweisen muss, damit das Modell universell ist, aus sprachtheoretischer Sicht große Bedeutung.
Aus den obigen Definitionen (vgl. Abschnitt 3.1) sowie den Resultaten aus [Ver10] (vgl. Theorem 3.4.) ist leicht ersichtlich, dass ein beliebig gewichtetes InsDel-System (d.h. ein InsDel-System, bei dem die Gewichte unbegrenzt sind, also n=m=p=q=* gilt) jede beliebige Grammatik generieren kann. Somit gilt die Inklusion: $INS^*_*DEL^*_* \subseteq RE$. Kann gezeigt werden, dass auch die Umkehrung $RE \subseteq INS^*_*DEL^*_*$ gilt (und somit $INS^*_*DEL^*_* = RE$), so ist das Modell universell und damit äquivalent zur Berechnungsstärke einer Turingmaschine. Dieser Beweis soll im Folgenden mit unterschiedlich gewichteten InsDel-Systemen geführt werden (vgl. [PRS98], [Ste06]). Als erstes werden die Gewichten mit den Werten (3, 2; 3, 0) belegt, d.h. die maximale Länge der Lösch- bzw. Einfügesequenz beträgt jeweils drei Zeichen (n=p=3), die maximale Länge des linken bzw. rechten Kontextes beim Einfügen einer Sequenz beträgt zwei Zeichen (m=2) und die maximale Länge des linken bzw. rechten Kontextes beim Löschen einer Sequenz beträgt null Zeichen (q=0), d.h. die

Deletion-Operation ist unabhängig von dem Kontext, indem die zu löschende Sequenz steht.

Theorem 3.3.1. $RE = INS_3^2 DEL_3^0$

Beweis. Sei $L \subseteq \Sigma^*$ eine beliebige Sprache, welche durch eine Typ-0-Grammatik $G = (V, \Sigma, P, S)$ notiert wurde. Da die Chomsky-Grammatiken vom Typ 0 genau die Sprachen aus RE beschreiben, soll zu einer solchen Grammatik G ein InsDel-System γ konstruiert werden, welches die gleiche Sprache beschreibt. O.B.d.A. wird angenommen, dass G in Kuroda-Normalform vorliegt (vgl. Definition 2.1.5). Es wird nun das folgende InsDel-System konstruiert:

$$
\begin{aligned}
\gamma &= (V_\gamma, \Sigma, A_\gamma, R_\gamma) \text{ , wobei} \\
V_\gamma &= V \cup \Sigma \cup \{E, K_1, K_2\}, \\
A_\gamma &= \{SEE\}, \\
R_\gamma &= \{(X, \varepsilon / K_1 x, \alpha_1 \alpha_2) \mid X \Rightarrow x \in P \text{ mit } X \in V, \\
&\qquad x \in (V \cup \Sigma)^*, |x| \leq 2 \text{ und } \alpha_1, \alpha_2 \in V \cup \Sigma \cup \{E\}\} \\
&\quad \cup \{(XY, \varepsilon / K_2 UZ, \alpha_1 \alpha_2) \mid XY \Rightarrow UZ \in P \text{ mit } X, Y, U, Z \in V \text{ und } \alpha_1, \alpha_2 \in V \cup \Sigma \cup \{E\}\} \\
&\quad \cup \{(\varepsilon, XK_1 / \varepsilon, \varepsilon) \mid X \in V\} \\
&\quad \cup \{(\varepsilon, XYK_2 / \varepsilon, \varepsilon) \mid X, Y \in V\} \\
&\quad \cup \{(\varepsilon, EE / \varepsilon, \varepsilon)\}
\end{aligned}
$$

Das Zeichen E wird ausschließlich als Kennzeichnung für das rechte Ende einer Wortform verwendet und am Schluss mittels einer Deletion-Regel wieder gelöscht. Die Zeichen K_1 sowie K_2 sind sogenannte "Killer": Das Symbol K_1 löscht ein Zeichen unmittelbar links seiner Position und analog dazu löscht K_2 zwei Zeichen unmittelbar links seiner Position. Die Insertion-Regeln des InsDel-Systems dienen der Simulation der Produktionen in der Grammatik G: Zunächst werden die rechten Regelseiten der in P vorkommenden Produktionen an die entsprechende linke Regelseite angefügt und mit einem Killersymbol markiert - je nachdem, ob die linke Regelseite der Produktion ein oder zwei Nichtterminalsymbole enthält, wird entweder K_1 oder K_2 eingefügt. Anschließend wird mittels einer entsprechenden Deletion-Regel die linke Regelseite entfernt. Ist ein Zeichen bereits mit einem "Killer"K_1 bzw. K_2 markiert, so kann es nicht mehr substituiert werden, da ihr rechter Kontext (bestehend aus den Zeichen α_1 sowie α_2) laut obiger Definition des InsDel-Systems keine Killersymbole mehr enthalten darf. Somit ergibt sich das Resultat: $L(G) = L(\gamma)$. □

Um diese Beweisführung noch einmal zu verdeutlichen, soll im Folgenden beispielhaft eine gegebene Grammatik in ein InsDel-System nach dem obigen Verfahren transformiert werden:

Beispiel 3.3.2 (Konstruktion eines Insertion-Deletion-Systems I)**.**
Gegeben sei die Grammatik $G = (V, \Sigma, P, S)$ mit $V = \{S\}, \Sigma = \{a\}$ und $P = \{S \Rightarrow \varepsilon, S \Rightarrow$

$Saa\}$, welche die Sprache $L(G) = \{a^{2n} \mid n \in \mathbb{N}\}$ beschreibt. Durch Transformation von G in Kuroda-Normalform ergibt sich die Grammatik $G' = (V', \Sigma, P', S)$ mit $V' = \{S, A, B\}$ und $P' = \{S \Rightarrow \varepsilon, S \Rightarrow SA, A \Rightarrow BB, B \Rightarrow a\}$ (vgl. Beispiel 2.1.8). Zu dieser Grammatik in Kuroda-Normalform wird nun das folgende InsDel-System γ_1 analog zum oben beschriebenen Verfahren konstruiert:

$$
\begin{aligned}
&\gamma_1 = (V_1, \Sigma, A_1, R_1) \text{ , wobei} \\
&V_1 = \{S, A, B, E, K_1, K_2, a\}, \\
&A_1 = \{SEE\}, \\
&R_1 = \{(S, \varepsilon / K_1, \alpha_1\alpha_2), (S, \varepsilon / K_1SA, \alpha_1\alpha_2), (A, \varepsilon / K_1BB, \alpha_1\alpha_2), (B, \varepsilon / K_1a, \alpha_1\alpha_2), \\
&\qquad (\varepsilon, SK_1 / \varepsilon, \varepsilon), (\varepsilon, AK_1 / \varepsilon, \varepsilon), (\varepsilon, BK_1 / \varepsilon, \varepsilon), \\
&\qquad (\varepsilon, XYK_2 / \varepsilon, \varepsilon), \\
&\qquad (\varepsilon, EE / \varepsilon, \varepsilon)\}
\end{aligned}
$$

für $\alpha_1, \alpha_2 \in \{S, A, B, a, E\}$ sowie $X, Y \in V$. Es ist zu beachten, dass γ_1 insgesamt mehr als neun Regeln besitzt, da für α_1 und α_2 verschiedene Zeichenkombinationen möglich sind. Von diesen Regeln werden allerdings einige gar nicht benötigt. Da in G' keine Regel der Form $XY \Rightarrow UZ \in P'$ enthalten ist, muss beispielsweise diese Produktion bei der Konstruktion des InsDel-Systems γ_1 gar nicht berücksichtigt werden, wodurch auch das Killersymbol K_2 vernachlässigt werden könnte. Dies zeigt, dass die obige Konstruktion nicht als effizient bezeichnet werden kann. Darüber hinaus wurde in Beispiel 3.1.9 ein InsDel-System konstruiert, welches die gleiche Sprache konstruiert, jedoch nur zwei Regeln besitzt.

Aus sprachtheoretischer Sicht ist die Frage, ob das Ergebnis des obigen Resultats durch Minimierung der Werte für die Gewichte m, n, p und q gestärkt werden kann, von großer Bedeutung. Auch für die Vereinfachung bei der laborpraktischen Implementierung der Insertion- und Deletion-Operation ist diese Frage interessant. Insbesondere ist der Fall bedeutsam, bei dem ausschließlich ein Zeichen eingefügt bzw. gelöscht wird (d.h. *n=1* sowie *p=1*), da derartige Operationen den Punktmutationen in der genetischen Entwicklung entsprechen. In dem folgenden Theorem werden die Gewichte (1, 2; 1, 1) betrachtet, d.h. die maximale Länge der Lösch- bzw. Einfügesequenz beträgt jeweils ein Zeichen (n=p=1), die maximale Länge des linken bzw. rechten Kontextes beim Einfügen einer Sequenz beträgt zwei Zeichen (m=2) und die maximale Länge des linken bzw. rechten Kontextes beim Löschen einer Sequenz beträgt ein Zeichen (q=1). Die Länge der Insertion- und Deletion-Sequenz kann also im Vergleicht zum ersten Resultat um zwei Zeichen minimiert werden. Allerdings wird bei dem folgenden Theorem ein Deletion-Kontext von maximal einem Zeichen benötigt.

Theorem 3.3.3. $RE = INS_1^2 DEL_1^1$

Beweis. Wie zu Beginn des Abschnittes bereits erwähnt, ist es ausreichend die Inklusion $RE \subseteq INS_1^2 DEL_1^1$ zu beweisen. Dazu wird eine beliebige Sprache $L \subseteq \Sigma^*$, $L \in RE$ betrachtet, welche durch eine Typ-0-Grammatik $G = (V, \Sigma, P, S)$ in Penttonen-Normalform (vgl.

Definition 2.1.6) notiert wurde. Dies bedeutet, dass die Grammatik ausschließlich kontextfreie Regeln der Form $X \Rightarrow x$ mit $|x| \leq 2$ (d.h. $x = \varepsilon$, $x \in \Sigma$ oder $x = YZ$ mit $Y, Z \in V$) sowie nicht-kontextfreie Regeln der Form $XY \Rightarrow XZ$ mit $X, Y, Z \in V$ beinhaltet.
O.B.d.A. wird angenommen, dass für jede Regel $X \Rightarrow \alpha_1\alpha_2 \in P$ gilt: $X \neq \alpha_1$, $X \neq \alpha_2$ sowie $\alpha_1 \neq \alpha_2$. (Ist dies nicht der Fall, so wird $X \Rightarrow \alpha_1\alpha_2$ mit den Regeln $X \Rightarrow X^{'}$, $X^{'} \Rightarrow \alpha_1\alpha_2^{'}$ sowie $\alpha_2^{'} \Rightarrow \alpha_2$ ersetzt, wobei $X^{'}$ und $\alpha_2^{'}$ neue Symbole sind.) Analog dazu wird angenommen, dass für jede Regel $XY \Rightarrow XZ$ gilt: $X \neq Y$, $X \neq Z$ sowie $Y \neq Z$. Darüber hinaus wird jede Regel der Form $X \Rightarrow \alpha \in P$ mit $\alpha \in (V \cup \Sigma)$ durch die beiden Regeln $X \Rightarrow \alpha Z$ und $Z \Rightarrow \varepsilon$ ersetzt, so dass eine zu G äquivalente Grammatik entsteht. Zusammengefasst enthält die Grammatik G also nur Regeln der folgenden Typen:

1. $X \Rightarrow \alpha_1\alpha_2$ mit $\alpha_1\alpha_2 \in (V \cup \Sigma)$ und $X \neq \alpha_1$, $X \neq \alpha_2$, $\alpha_1 \neq \alpha_2$,
2. $X \Rightarrow \varepsilon$,
3. $XY \Rightarrow XZ$ mit $X, Y, Z \in V$ und $X \neq Y$, $X \neq Z$ sowie $Y \neq Z$.

Außerdem wird angenommen, dass jede Regel in P in der gleichen Weise markiert ist.
Nun wird das folgende InsDel-System konstruiert:

$$\begin{aligned} \gamma &= (V_\gamma, \Sigma, A_\gamma, R_\gamma), \text{ wobei} \\ V_\gamma &= V \cup \Sigma \cup \{[r], (r) \mid \text{ r ist die Markierung der Regel in } P \} \cup \{B, E\}, \\ A_\gamma &= \{BSE\} \end{aligned}$$

und für die Menge R_γ die folgenden Regeln erstellt:

1. Für jede Regel $r : X \Rightarrow \alpha_1\alpha_2 \in P$ vom Typ 1, mit $\alpha_1\alpha_2 \in V \cup \Sigma$, werden die folgenden Insertion-Regeln konstruiert:
   a) $(\beta_1, \varepsilon/[r], X\beta_2)$ mit $\beta_1 \in V \cup \Sigma \cup \{B\}$ und $\beta_2 \in V \cup \Sigma \cup \{E\}$,
   b) $([r]X, \varepsilon/(r), \beta)$ mit $\beta \in V \cup \Sigma \cup \{B\}$,
   c) $([r], X/\varepsilon, (r))$,
   d) $([r], \varepsilon/\alpha_1, (r))$,
   e) $(\alpha_1, \varepsilon/\alpha_2, (r))$,
   f) $(\varepsilon, [r]/\varepsilon, \alpha_1)$,
   g) $(\alpha_2, (r)/\varepsilon, \varepsilon)$.
2. Für jede Regel $r : X \Rightarrow \varepsilon \in P$ vom Typ 2 wird folgende Deletion-Regel eingeführt:
   $(\beta_1, X/\varepsilon, \beta_2)$ mit $\beta_1 \in V \cup \Sigma \cup \{B\}$ und $\beta_2 \in V \cup \Sigma \cup \{E\}$.
3. Für jede Regel $r : XY \Rightarrow XZ \in P$ vom Typ 3, mit $X, Y, Z \in V$, werden die folgenden Insertion-Deletion-Regeln konstruiert:
   a) $(\beta_1 X, \varepsilon/[r], Y\beta_2)$ mit $\beta_1 \in V \cup \Sigma \cup \{B\}$ und $\beta_2 \in V \cup \Sigma \cup \{E\}$,

b) $([r]Y, \varepsilon/(r), \beta)$ mit $\beta \in V \cup \Sigma \cup \{B\}$,

c) $([r], Y/\varepsilon, (r))$,

d) $([r], \varepsilon/Z, (r))$,

e) $(X, [r]/\varepsilon, Z)$,

f) $(Z, (r)/\varepsilon, \varepsilon)$.

4. Außerdem werden die folgenden Deletion-Regeln konstruiert:

   a) $(\varepsilon, B/\varepsilon, \varepsilon)$,

   b) $(\varepsilon, E/\varepsilon, \varepsilon)$.

Nun muss noch gezeigt werden, dass die vom obigen InsDel-System γ beschriebene Sprache $L(\gamma)$ äquivalent zu der von der Grammatik G beschriebenen Sprache $L(G)$ ist. Diese Gleichheit $L(G) = L(\gamma)$ wird bewiesen, indem gezeigt wird, dass $L(G)$ Teilmenge von $L(\gamma)$ sowie $L(\gamma)$ Teilmenge von $L(G)$ ist:

$(L(G) \subseteq L(\gamma))$
γ simuliert jeden Ableitungsschritt $w \Rightarrow w'$ der Grammatik G, indem ausgehend von BwE sukzessiv die Wortform $Bw'E$ abgeleitet wird. Dabei werden diejenigen der oben aufgeführten Regeln verwendet, welche mit den entsprechenden Regeln in P für die Ableitung $w \Rightarrow w'$ assoziiert werden. Dies bedeutet, dass für einen Ableitungsschritt $r : X \Rightarrow \alpha_1\alpha_2 \in P$ das InsDel-System γ die sieben unter Punkt 1 aufgeführten Insertion-Deletion-Regeln simuliert. Beispielsweise ergibt sich bei dem Wort $w = w_1Xw_2$, welches nach $w' = w_1\alpha_1\alpha_2w_2$ abgeleitet werden soll (also die Regel $r : X \Rightarrow \alpha_1\alpha_2 \in P$ simuliert wird):

$$
\begin{aligned}
Bw_1Xw_2E &\Rightarrow_\gamma Bw_1[r]Xw_2E && \text{(Regel (a))}\\
&\Rightarrow_\gamma Bw_1[r]X(r)w_2E && \text{(Regel (b))}\\
&\Rightarrow_\gamma Bw_1[r](r)w_2E && \text{(Regel (c))}\\
&\Rightarrow_\gamma Bw_1[r]\alpha_1(r)w_2E && \text{(Regel (d))}\\
&\Rightarrow_\gamma Bw_1[r]\alpha_1\alpha_2(r)w_2E && \text{(Regel (e))}\\
&\Rightarrow_\gamma Bw_1\alpha_1\alpha_2(r)w_2E && \text{(Regel (f))}\\
&\Rightarrow_\gamma Bw_1\alpha_1\alpha_2w_2E && \text{(Regel (g))}\\
&= Bw'E
\end{aligned}
$$

Analog dazu erfolgt die Ableitung $w \Rightarrow w'$ des Wortes $w = w_1XYw_2$ nach $w' = w_1XZw_2$

unter Verwendung der Regel $r : XY \Rightarrow XZ \in P$:

$$\begin{aligned} Bw_1XYw_2E &\Rightarrow_\gamma Bw_1X[r]Yw_2E && \text{(Regel (a))} \\ &\Rightarrow_\gamma Bw_1X[r]Y(r)w_2E && \text{(Regel (b))} \\ &\Rightarrow_\gamma Bw_1X[r](r)w_2E && \text{(Regel (c))} \\ &\Rightarrow_\gamma Bw_1X[r]Z(r)w_2E && \text{(Regel (d))} \\ &\Rightarrow_\gamma Bw_1XZ(r)w_2E && \text{(Regel (e))} \\ &\Rightarrow_\gamma Bw_1XZw_2E && \text{(Regel (f))} \\ &= Bw^{'}E \end{aligned}$$

Ausgehend vom Axiom BSE kann somit jedes Wort in γ abgeleitet werden, welches auch in G abgeleitet werden kann. Die Marker B und E können am Ende der Ableitung mit Hilfe der unter Punkt 4 aufgeführten Deletion-Regeln wieder entfernt werden. Insgesamt ergibt sich also das Resultat $L(G) \subseteq L(\gamma)$.

$(L(G) \supseteq L(\gamma))$
Betrachtet wird das Wort BwE, wobei initial $w = S$ gilt. Es besteht nun die Möglichkeit eine der folgenden vier Regeln auf diese Sequenz anzuwenden:

- die Regel (a) aus Punkt 1,
- die Deletion-Regel aus Punkt 2, welche $X \Rightarrow \varepsilon \in P$ simuliert,
- die Regel (a) aus Punkt 3 oder
- eine der beiden Deletion-Regeln aus Punkt 4.

Es wird die Annahme getroffen, dass sich für erstgenannte Variante entschieden wird, d.h. es wird die Insertion-Operation $(\beta_1, \varepsilon/[r], X\beta_2)$ angewendet, welche die Regel $r : X \Rightarrow \alpha_1\alpha_2 \in P$ simuliert. Daraus ergibt sich

$$Bw_1Xw_2E \Rightarrow_\gamma Bw_1[r]Xw_2E.$$

Da die Regeln in P in derselben Weise nummeriert sind, $X \neq \alpha$ gilt und die Regeln (a) der Punkte 1 bzw. 3 sowie die Deletion-Regel in Punkt 2 den Kontext, welcher sich direkt links von X befindet, überprüft, ist es nur möglich die Regel (b) aus Punkt 1 auf X anzuwenden. Wird diese Regel nicht angewendet, so kann die Berechnung nicht terminieren. Es ergibt sich also:

$$Bw_1X[r]Yw_2E \Rightarrow_\gamma Bw_1X[r]Y(r)w_2E.$$

Auch hier ist nur eine mögliche Fortsetzung möglich, nämlich die Regel (c) aus Punkt 1, welche das Zeichen X entfernt. Erst nach dem Einfügen des Zeichens α_1 zwischen $[r]$ und (r), kann das Zeichen $[r]$ gelöscht werden, weil die entsprechende Deletion-Regel den Kontext direkt rechts von α_1 überprüft. Wurde α_1 eingefügt, so kann nun auch das Zeichen α_2 mit Hilfe der Regel (e) eingeführt werden. Da (r) erst nach $[r]$ eingefügt wurde und $X \neq \alpha_1$ gilt, muss das Zeichen α_1 vor der Anwendung der Regel (e) eingeführt werden, da

α_1 als linker Kontext in dieser Regel verwendet wird. Daher muss vor der Regel (e) die Regel (d) aus Punkt 1 angewendet werden. Nachdem α_2, wobei $\alpha_2 \neq \alpha_1$ sowie $\alpha_2 \neq X$ gilt, kann (r) mittels der Regel (g) entfernt werden. Aufgrund der in den Regeln enthaltenen Kontexte, kann keine andere Regel die erwähnten Zeichen als Kontexte verwenden bzw. löschen. Daher müssen nach Anwendung der Regel (a) aus Punkt 1 alle Regeln (b), (c), ... , (g) des Punktes 1 angewendet werden, damit die Produktion $r : X \Rightarrow \alpha_1\alpha_2$ simuliert wird. Analog dazu müssen, nachdem eine Regel $(\beta_1 X, \varepsilon/[r], Y\beta_2)$ (Regel (a) des obigen Punkt 3) in Assoziation zu $r : XY \Rightarrow XZ \in P$ angewendet wurde, auch hier die Regeln (b), (c), ... , (g) des Punktes 3 (möglicherweise nicht unmittelbar bzw. in aufeinanderfolgender Weise, aber unter Verwendung der gleichen Zeichen des aktuellen Wortes) angewendet werden, um die Produktion $XY \Rightarrow XZ$ zu simulieren.
Die Deletion-Regel $(\beta_1, X/\varepsilon, \beta_2)$ entspricht dem unmittelbaren Löschen von Regeln in P. Die Marker B und E können in jedem Schritt mit Hilfe der Regeln aus Punkt (4) entfernt werden. Daraus ergibt sich, dass γ nur Wörter aus $L(G)$ konstruieren kann. □

Auch hier soll ein kurzes Beispiel betrachtet werden, um die beschriebene Konstruktion des InsDel-Systems noch einmal zu verdeutlichen:

Beispiel 3.3.4 (Konstruktion eines Insertion-Deletion-Systems II)**.**
Sei G die Grammatik aus 3.3.2, welche in Penttonen-Normalform transformiert wird, so dass sich die Grammatik $G' = (V', \Sigma, P', S)$ mit $V' = \{S, A, B\}$ sowie $P' = \{S \Rightarrow \varepsilon, S \Rightarrow SA, A \Rightarrow BB, B \Rightarrow a\}$ ergibt (vgl. Beispiel 2.1.8). Von diesem Ausgangspunkt aus wird das folgende InsDel-System γ_2 konstruiert:

$$
\begin{aligned}
\gamma_2 &= (V_2, \Sigma, A_2, R_2), \text{ wobei} \\
V_2 &= \{S, A, B, a, \{r\}, (r), D, E\}, \\
A_2 &= \{DSE\}, \\
R_2 &= \{(\beta_1, \varepsilon/[r], S\beta_2), (\beta_1, \varepsilon/[r], A\beta_2), (\beta_1, \varepsilon/[r], B\beta_2), \\
&\quad ([r]S, \varepsilon/(r), \beta), ([rAS, \varepsilon/(r), \beta), ([r]B, \varepsilon/(r), \beta), \\
&\quad ([r], S/\varepsilon, (r)), ([r], A/\varepsilon, (r)), ([r], B/\varepsilon, (r)), \\
&\quad ([r], \varepsilon/\alpha_1, (r)), \\
&\quad (\alpha_1, \varepsilon/\alpha_1, (r)), \\
&\quad (\varepsilon, [r]\varepsilon, \alpha_1), \\
&\quad (\alpha_2, (r)/\varepsilon, \varepsilon), \\
&\quad (\beta_1, S/\varepsilon, \beta_2), \\
&\quad (\varepsilon, D/\varepsilon, \varepsilon), (\varepsilon, E/\varepsilon, \varepsilon)\}.
\end{aligned}
$$

für $\beta_1 \in \{S, A, B, D, a\}$ sowie $\beta, \beta_2 \in \{S, A, B, E, a\}$ und $\alpha_1, \alpha_2 \in \{S, A, B, a\}$. Wird diese Konstruktion im Vergleich zur Konstruktion aus Beispiel 3.1.9 sowie 3.3.2 betrachtet, so ist leicht ersichtlich, dass sie wesentlich ineffizienter ist.

Bei dem letzten Resultat, welches ausführlich betrachtet werden soll, wird gezeigt, dass

ein InsDel-System mit den Gewichten (1, 1; 2, 0) die Sprachklasse der rekursiv aufzählbaren Sprachen beschreiben kann. Dies bedeutet, dass die maximale Länge der Einfügesequenz ein Zeichen umfasst (n=1), die maximale Löschsequenz zwei Zeichen (p=2) sowie der linke bzw. rechte Kontext beim Einfügen einer Sequenz maximal ein Zeichen beträgt (m=1) und der linke bzw. rechte Kontext beim Löschen einer Sequenz frei wählbar (q=0) ist. Im Gegensatz zum vorigen Theorem konnte also die Anzahl der Zeichen im linken bzw. rechten Kontext beim Löschen und Einfügen einer Sequenz um jeweils ein Zeichen reduziert werden. Allerdings ist für die Löschsequenz ein Zeichen mehr erforderlich.

Theorem 3.3.5. $RE = INS_1^1 DEL_2^0$

Beweis. Sei $L \in RE, L \subseteq \Sigma^*$ eine beliebige Sprache und $G = (V, \Sigma, P, S)$ eine Grammatik in Geffert-Normalform (vgl. Definition 2.1.7), so dass $L = L(G)$ gilt. Für die Menge der Produktionen P gelte $P = P_1 \cup P_2$, wobei

- P_1 nur kontextfreie Regeln der folgenden Formen enthält:
  - $S \Rightarrow uSv$, für $u, v \in (V \cup \Sigma - \{S\})^+$,
  - $S \Rightarrow x$, für $x \in (V \cup \Sigma - \{S\})^+$.
- P_2 nur Regeln der Form $XY \Rightarrow \varepsilon$ für $X, Y \in V$ beinhaltet.

Es wird das folgende InsDel-System konstruiert:

$$
\begin{aligned}
&\gamma = (V, \Sigma, A, R) \text{ , wobei} \\
&V = V \cup \Sigma \cup \{c, K, K', F\} \\
&\quad \cup \{[S, r] \mid r : S \Rightarrow x \in P_1\} \\
&\quad \cup \{[X, r, i] \mid r : S \Rightarrow zXw \in P_1,\ z, w \in (V \cup \Sigma)^*,\ i = |z| + 1,\ X \in V \cup \Sigma\}, \\
&A = \{Sc\},
\end{aligned}
$$

und die folgenden Regeln für die Menge R erstellt:

1. Zuerst wird jede Regel $S \Rightarrow uSv$ in P durch die Regel $S \Rightarrow uScv$ ersetzt, d.h. es wird auf der rechten Regelseite das Zeichen c zwischen S und v eingefügt. Die Regeln der Form $S \Rightarrow x$ mit $|x|_S = 0$ (d.h. alle Regeln, bei denen das Startsymbol S nicht auf der rechten Seite vorkommt) bleiben unverändert. Die so entstandene neue Menge wird mit P_1' gekennzeichnet. Für jede Regel der Form $r : S \Rightarrow X_1 X_2 ... X_k \in P_1'$ mit $X_i \in V \cup \Sigma \cup \{c\},\ 1 \leq i \leq k,\ k \geq 1$, werden die folgenden Regeln eingeführt:
   a) $(S, \varepsilon / [S, r], c)$,
   b) $(S, \varepsilon / K, [S, r])$,
   c) $(\varepsilon, SK / \varepsilon, \varepsilon)$,
   d) $([S, r], \varepsilon / [X_1, r, 1], c)$,

e) $([X_i, r, i], \varepsilon / [X_{i+1}, r, i+1], c)$, $1 \leq i \leq k-1$,

f) $([S, r], \varepsilon / K, [X_1, r, 1])$,

g) $(\varepsilon, [S, r]K / \varepsilon, \varepsilon)$,

h) $([X_i, r, i], \varepsilon / X_i, [X_{i+1}, r, i+1])$, $1 \leq i \leq k-1$,

i) $([X_i, r, i], \varepsilon / K, X_i)$, $1 \leq i \leq k-1$,

j) $(\varepsilon, [X_i, r, i]K / \varepsilon, \varepsilon)$, $1 \leq i \leq k-1$,

k) $([X_k, r, k], \varepsilon / F, c)$,

l) $([X_k, r, k], \varepsilon / X_k, F)$,

m) $([X_k, r, k], \varepsilon / K, X_k)$,

n) $(X_k, \varepsilon / K', F)$,

o) $(\varepsilon, K'F / \varepsilon, \varepsilon)$.

2. Darüber hinaus werden die folgenden Deletion-Regeln in die Menge R eingefügt:

   a) $(\varepsilon, XY / \varepsilon, \varepsilon)$ mit $XY \Rightarrow \varepsilon \in P_2$,

   b) $(\varepsilon, c / \varepsilon, \varepsilon)$.

Es ist leicht ersichtlich, dass das InsDel-System γ die Sprache $L(G)$ beschreibt, wenn die kontextfreien Regeln aus P_1 in γ korrekt simuliert werden. Um diesen Sachverhalt zu zeigen, wird eine beliebige Wortform $w \in (V \cup \Sigma \cup \{c\})^*$ betrachtet. Es besteht nun die Möglichkeit eine der Regeln aus Punkt 2 sowie eine Regel des Typs a) aus Punkt 1 auf die vorliegende Sequenz anzuwenden. Wird sich für die letztgenannte Variante entschieden, so kann eine Teilsequenz Sc von w durch $S[S, r]c$ ersetzt werden für eine Produktion $r \in P_1'$. Angenommen, dass $w = w_1 S c w_2$ (initial ist $w_1 = w_2 = \varepsilon$) und die eben genannte Regel für $r : S \Rightarrow X_1 X_2 ... X_k$, $k \geq 1$ angewendet wird, dann ergibt sich die Sequenz $w_1 S[S, r]cw_2$. Ausgehend von $[S, r]$ soll nun $X_1 X_2 ... X_k$ produziert werden. Da $X_1 \neq c$ und $k \geq 1$, kann das Vorkommen von S in $w_1 S[S, r]cw_2$ nicht noch einmal von c gefolgt auftreten und daher die Regel vom Typ a) nicht erneut angewendet werden. Mit Hilfe der Regeln des Typs b) und c) muss dieses Vorkommen von S entfernt werden. Dabei spielt der sogenannte "Killer" K, welcher das Zeichen, das unmittelbar links von K steht, löscht, eine wichtige Rolle: Mit Hilfe der Regel b) wird das Killersymbol K unmittelbar rechts von S eingefügt und durch die Regel c) wird die Teilsequenz SK wieder gelöscht:

$$w_1 S[S, r]cw_2 \Rightarrow_\gamma w_1 SK[S, r]cw_2 \Rightarrow_\gamma w_1[S, r]cw_2.$$

Die einzige Möglichkeit mit der Ableitung fortzufahren, ist durch die Anwendung der Regeln des Typs d), e) oder h) gegeben, da das Killersymbol K (welches in der Lage ist die Zeichen $[S, r], [X_i, r, i], 1 \leq i \leq k-1$ zu entfernen) nur dann eingeführt werden kann, wenn sowohl im linken als auch im rechten Kontext von K nur Zeichen der Form

$[S,r], [X_i,r,i], 1 \leq i \leq k-1$ auftreten. Daher muss schlussendlich die folgende Ableitung ausgeführt werden:

$$w_1[S,r]cw_2 \Rightarrow_\gamma w_1[S,r][X_1,r,1]cw_2 \Rightarrow_\gamma w_1[S,r][X_1,r,1][X_2,r,2]cw_2$$
$$\Rightarrow_\gamma ... \Rightarrow_\gamma w_1[S,r][X_1,r,1]...[X_{k-1},r,k-1][X_k,r,k]cw_2.$$

Nach dieser Ableitung (bzw. zwischen die einzelnen Ableitungsschritte gelagert) wird das Killersymbol K zwischen $[S,r]$ und $[X_1,r,1]$ analog zur Regel f) eingeführt. Darüber hinaus wird mit Hilfe der Regel h) zwischen jedes $[X_i,r,i]$ und $[X_{i+1},r,i+1]$ für alle $1 \leq i \leq k-1$ das Zeichen X_i eingefügt. Somit ergibt sich das folgende Wort:

$$w_1[S,r]K[X_1,r,1]X_1[X_2,r,2]X_2...[X_{k-1},r,k-1]X_{k-1}[X_k,r,k]cw_2.$$

Der Block $[S,r]K$ kann nun mittels der Regel g) gelöscht werden (und dies ist auch die einzige Möglichkeit, um $[S,r]$ überhaupt jemals zu entfernen). Zwischen den Sequenzen $[X_i,r,i]$ und X_i kann erneut das Killersymbol K gemäß Regel i) eingeführt werden, welches im nächsten Schritt gemeinsam mit dem Block $[X_i,r,i]$ für alle $1 \leq i \leq k-1$ mit Hilfe der Regel j) wieder gelöscht wird. So ergibt sich das Wort:

$$w_1X_1X_2...X_{k-1}[X_k,r,k]cw_2.$$

Um nun noch die Teilsequenz $[X_k,r,k]$ zu entfernen, muss zunächst das Zeichen F gemäß Regel k) eingeführt sowie anschließend das Zeichen X_k mit Hilfe der Regel l) eingefügt werden:

$$w_1X_1X_2...X_{k-1}[X_k,r,k]cw_2 \Rightarrow_\gamma w_1X_1X_2...X_{k-1}[X_k,r,k]Fcw_2$$
$$\Rightarrow_\gamma w_1X_1X_2...X_{k-1}[X_k,r,k]X_kFcw_2$$

Nun kann die Teilsequenz $[X_k,r,k]$ analog zu jedem anderen Zeichen $[X_i,r,i]$ gelöscht werden, indem zunächst das Killersymbol K gemäß Regel m) zwischen $[X_k,r,k]$ und X_k eingeführt und anschließend gemeinsam mit dem Zeichen $[X_k,r,k]$ wieder gelöscht wird. Durch Einfügen des Killersymbols K' gemäß Regel n), kann im nächsten Schritt mit Hilfe der Regel o) das Zeichen F gemeinsam mit dem zuvor eingeführten Killersymbol K' wieder entfernt werden.
Zu beachten ist, dass ohne das Vorkommen des Startsymbols S in $X_1X_2...X_k$ kein weiterer Ableitungsschritt gemäß der Regeln von Punkt 1 ausgeführt werden kann. Gibt es jedoch ein weiteres Vorkommen von S in $X_1X_2...X_k$ gefolgt vom Zeichen c, so sind neue Applikationen der Regeln aus Punkt 1 möglich. Das Auftreten des Zeichens c kann am Ende der Applikation mittels der Deletion-Regel $(\varepsilon, c/\varepsilon, \varepsilon)$ aus Punkt 2 entfernt werden. Soll das Zeichen c entfernt werden, während das Startsymbol S in der Sequenz enthalten ist und noch nicht zur Simulation einer Regel aus P_1 verwendet wurde, dann kann das Startsymbol S nicht mehr entfernt werden und die Abarbeitung würde zu keinem terminalen Wort führen. Daher gilt $L(\gamma) = L(G)$. □

Auch hier soll ein Beispiel betrachtet werden, um die eben erläuterte Konstruktion noch einmal zu verdeutlichen:

Beispiel 3.3.6 (Konstruktion eines Insertion-Deletion-Systems III)**.**
Sei G erneut die Grammatik aus 3.3.2. Durch Transformation in Geffert-Normalform (vgl. Beispiel 2.1.8) ergibt sich die Grammatik $G' = (V, \Sigma, P', S)$ mit $P' = \{S \Rightarrow aSa, S \Rightarrow \varepsilon\}$. Davon ausgehend kann nun das InsDel-System γ_3 konstruiert werden:

$$\begin{aligned}
&\gamma_3 = (V_3, \Sigma, A_3, R_3) \text{ , wobei} \\
&V_3 = \{S, a, c, K, K', F, [S, S \Rightarrow \varepsilon], [S, S \Rightarrow aSa, 2]\}, \\
&A = \{Sc\},
\end{aligned}$$

und die folgenden Regeln für die Menge R_3 erstellt werden:

1. Die erste Produktion $S \Rightarrow aSa$ in P' wird zu $S \Rightarrow aSca$ transformiert und für diese Produktion die folgenden Regeln eingeführt:

   a) $(S, \varepsilon / [S, S \Rightarrow aSca], c)$,

   b) $(S, \varepsilon / K, [S, S \Rightarrow aSca])$,

   c) $(\varepsilon, SK / \varepsilon, \varepsilon)$,

   d) $([S, S \Rightarrow aSca], \varepsilon / [a, S \Rightarrow aSca, 1], c)$,

   e) $([a, S \Rightarrow aSca, 1], \varepsilon / [S, S \Rightarrow aSca, 2], c)$,
   $([S, S \Rightarrow aSca, 2], \varepsilon / [c, S \Rightarrow aSca, 3], c)$,
   $([c, S \Rightarrow aSca, 3], \varepsilon / [a, S \Rightarrow aSca, 4], c)$,

   f) $([S, S \Rightarrow aSca], \varepsilon / K, [a, S \Rightarrow aSca, 1])$,

   g) $(\varepsilon, [S, S \Rightarrow aSca]K / \varepsilon, \varepsilon)$,

   h) $([a, S \Rightarrow aSca, 1], \varepsilon / a, [S, S \Rightarrow aSca, 2])$,
   $([S, S \Rightarrow aSca, 2], \varepsilon / S, [c, S \Rightarrow aSca, 3])$,
   $([c, S \Rightarrow aSca, 3], \varepsilon / c, [a, S \Rightarrow aSca, 4])$,

   i) $([a, S \Rightarrow aSca, 1], \varepsilon / K, a)$,
   $([S, S \Rightarrow aSca, 2], \varepsilon / K, S)$,
   $([C, S \Rightarrow aSca, 3], \varepsilon / K, c)$,

   j) $(\varepsilon, [a, S \Rightarrow aSca, 1]K / \varepsilon, \varepsilon)$,
   $(\varepsilon, [S, S \Rightarrow aSca, 2]K / \varepsilon, \varepsilon)$,
   $(\varepsilon, [c, S \Rightarrow aSca, 3]K / \varepsilon, \varepsilon)$,
   $(\varepsilon, [a, S \Rightarrow aSca, 4]K / \varepsilon, \varepsilon)$,

k) $([a, S \Rightarrow aSca, 4], \varepsilon / F, c)$,

l) $([a, S \Rightarrow aSca, 4], \varepsilon / a, F)$,

m) $([a, S \Rightarrow aSca, 4], \varepsilon / K, a)$,

n) $(a, \varepsilon / K', F)$,

o) $(\varepsilon, K'F / \varepsilon, \varepsilon)$.

2. Die zweite Produktion $S \Rightarrow \varepsilon$ in P' bleibt unverändert, so dass sich die folgenden Regeln ergeben:

a) $(S, \varepsilon / [S, S \Rightarrow \varepsilon], c)$,

b) $(S, \varepsilon / K, [S, S \Rightarrow \varepsilon])$,

c) $(\varepsilon, SK / \varepsilon, \varepsilon)$,

d) $([S, S \Rightarrow \varepsilon], \varepsilon / [\varepsilon, S \Rightarrow \varepsilon, 1], c)$,

e) —

f) $([S, S \Rightarrow \varepsilon], \varepsilon / K, [\varepsilon, S \Rightarrow \varepsilon, 1])$,

g) $(\varepsilon, [S, S \Rightarrow \varepsilon]K / \varepsilon, \varepsilon)$,

h) —

i) —

j) $(\varepsilon, [\varepsilon, S \Rightarrow \varepsilon, 1]K / \varepsilon, \varepsilon)$,

k) $([\varepsilon, S \Rightarrow \varepsilon, 1], \varepsilon / F, c)$,

l) $([\varepsilon, S \Rightarrow \varepsilon, 1], \varepsilon / \varepsilon, F)$,

m) $([\varepsilon, S \Rightarrow \varepsilon, 1], \varepsilon / K, \varepsilon)$,

n) $(\varepsilon, \varepsilon / K', F)$,

o) $(\varepsilon, K'F / \varepsilon, \varepsilon)$.

Es ist leicht ersichtlich, dass dieses InsDel-System noch ineffizienter ist, als die aufgeführten InsDel-Systeme in Beispiel 3.3.4, 3.3.2 sowie 3.1.9. Die Regelmenge R_3 beinhaltet noch einmal mehr Regeln als die Regelmenge R_2, da allein für eine Produktion mehr als 10 Regeln in die Regelmenge aufgenommen werden müssen.

Die Frage, wie mächtig die Klasse $INS_1^1DEL_1^1$ ist, ist momentan noch ungeklärt. Es scheint jedoch so, dass mindestens einer der Parameter m (der Insertion-Kontext), q (der Deletion-Kontext) oder p (die Deletion-Sequenz) mindestens 1 Zeichen umfassen muss, um nicht-kontextfreie Grammatiken konstruieren zu können. Das Resultat $INS_*^1DEL_0^0 \subseteq CF$ wurde in [PRS98] (vgl. Theorem 6.4.) bewiesen und ist ein Indiz dafür, dass die obige Annahme zu stimmen scheint.
Auch die Klasse der kontextfreien InsDel-Systeme kann auf Universalität untersucht werden: In [Ver10] Abschnitt 4 wird die Gleichheit $INS_*^0DEL_*^0 = RE$ bewiesen. Die Länge der Insertion- und Deletion-Sequenz ist in diesem Resultat unbegrenzt, kann jedoch leicht mit einer Begrenzung versehen werden, indem die Länge der in den Grammatikregeln auftretenden Sequenzen betrachtet wird. Bei einer Grammatik in Kuroda-Normalform, kann die Insertion- und Deletion-Sequenz auf drei Zeichen begrenzt werden, d.h. $INS_3^0DEL_3^0 = RE$ (vgl Theorem 4.3. in [Ver10]). Auch dieses Resultat kann noch einmal verbessert werden, indem die Insertion- oder Deletion-Sequenz um ein Zeichen verringert wird. Das heißt auch die Gleichheit $INS_2^0DEL_3^0 = RE$ (vgl Theorem 4.4. in [Ver10]) sowie $INS_3^0DEL_2^0 = RE$ (vgl Theorem 4.5. in [Ver10]) konnten bewiesen werden.

4 Zusammenfassung

Insertion-Deletion-Systeme sind ein formales Berechnungsmodell des DNA-Computings, welche eine Reihe nützlicher Eigenschaften aufweisen. Zu diesen Eigenschaften zählt beispielsweise die Universalität von Insertion-Deletion-Systemen, da das Modell unter der Voraussetzung, dass die Modellparameter geeignet gewählt werden, universell ist. Die Gewichte (n, m; p, q) stellen die eben erwähnten Modellparameter eines Insertion-Deletion-Systems dar und haben Einfluss auf die Universalität des Modells. Daher ist die Feststellung, welche minimale Wertebelegung für diese Parameter notwendig sind, damit das entsprechende Insertion-Deletion-System alle Sprachen aus der Klasse RE konstruieren kann (d.h. universell ist) eine interessante Fragestellung, welche in der vorliegenden Arbeit diskutiert wurde. Als Resultat wurde zuerst gezeigt, dass ein Insertion-Deletion-System mit den Gewichten (3, 2; 3, 0) die Universalitätseigenschaft besitzt. Darauf aufbauend wurde der Beweis ausgeführt, dass auch Insertion-Deletion-Systeme mit kleineren Gewichten - nämlich (1, 2; 1, 1) sowie (1, 1; 2, 0) - die Sprachklasse RE konstruieren können. Auch Resultate bezüglich der Universalität von kontextfreien Insertion-Deletion-Systemen sind bekannt: Die minimalen Gewichte für n bzw. p betragen hierbei zwei und drei bzw. drei und zwei Zeichen.
Diese Ergebnisse verdeutlichen wie mächtig Insertion-Deletion-Systeme sind, da sie äquivalent zur Berechnungsstärke einer Turingmaschine sind. Daher ist es sehr wahrscheinlich, dass Insertion-Deletion-Systeme auch in der zukünftigen Forschung im Bereich des DNA-Computings eine bedeutende Rolle spielen werden.

Danksagung

Besonderer Dank gebührt ...

Literaturverzeichnis

[Baa10] Franz Baader. *Formale Systeme (Skript zur Vorlesung)*. TU Dresden, 2010.

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

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

[Ste06] Florian Stenger. *Insertion/Deletion-Systeme als Modell molekularen Rechnens*. TU Dresden, 2006.

[Stu12] Monika Sturm. *LV Molekulares Rechnen (in vitro und in vivo)*. TU Dresden, 2012.

[Ver10] Sergey Verlan. *Recent Developments on Insertion-Deletion Systems*. Computer Science Journal of Moldova, vol.18, no.2(53), 2010.

Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, dass alle Stellen der Arbeit, die wörtlich oder sinngemäß aus anderen Quellen übernommen wurden, als solche kenntlich gemacht und dass die Arbeit in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegt wurde.

Ort, Datum Unterschrift