,

Eine DNA-Arithmetik auf der Basis von Chomsky-Grammatiken

THOMAS HINZE MONIKA STURM
Technische Universität Dresden, Institut für Theoretische Informatik
e-mail: {hinze,sturm}@tcs.inf.tu-dresden.de

KURZFASSUNG

Numerische Berechnungen gehören zum täglichen Handwerk in vielen Bereichen der Natur-, Ingenieur- und Sprachwissenschaften. Das DNA-Computing etabliert sich zunehmend als innovativer Ansatz, um Rechenvorgänge massiv datenparallel auf biologischer Grundlage auszuführen. Als wichtiges Einsatzgebiet gilt die Bearbeitung besonders rechen- und speicherintensiver Aufgabenstellungen, die jedoch häufig auch numerische Berechnungen als Teilprobleme enthalten. Die aus der Linguistik stammende Idee der Chomsky-Grammatiken, Rechenvorgänge durch gezieltes Ineinander-Einsetzen von Zeichenketten nachzubilden, steht der Verarbeitung von DNA im DNA-Computing sehr nahe. Viele DNA-Computer lassen sich mittels Chomsky-Grammatiken vorteilhaft programmieren. Es wird gezeigt, wie sich damit die Funktionalität von Taschenrechnern umsetzen lässt, indem beliebige Terme auf natürlichen Zahlen, die die Bausteinoperationen Addition, Subtraktion, Multiplikation und Division enthalten können, konstruktiv in Chomsky-Grammatiken transformiert werden, so dass die anschließende Berechnung durch ein grammatikbasiertes Splicing- oder P-System effizient möglich ist.

Stichwörter: Chomsky-Grammatik, DNA-Arithmetik, DNA-Computing, formale Sprache, numerischer Algorithmus

1. Einführung

Chomsky-Grammatiken haben sich als geeignetes Programmierwerkzeug zur Notation von Algorithmen für universelle DNA-Computer erwiesen [4]. DNA-Computer verwenden Moleküle der Erbsubstanz DNA als Datenträger und Speichermedium. Rechenvorgänge werden durch Abfolgen geeigneter molekularbiologischer und biochemischer Prozesse realisiert, die auf einen DNA-Pool im Reagenzglas oder innerhalb einer Zelle einwirken. Dabei kann gleichzeitig eine Vielzahl von DNA-Molekülen analysiert, selektiert und/oder modifiziert werden. Die formal-abstrakte Beschreibung von DNA-basierten Berechnungen stützt sich zumeist auf Chomsky-Grammatiken [5]. Die durch Regeln gesteuerten Ersetzungen von Zeichenkettenteilen durch andere Zeichenketten korrespondieren mit dem gezielten sequenzspezifischen Heraustrennen und Einfügen von Abschnitten innerhalb von DNA-Strängen. Es konnte konstruktiv gezeigt werden, dass der allgemeinste Typ von Chomsky-Grammatiken (Typ 0), der die Generierung aller rekursiv aufzählbaren Sprachen ermöglicht, die Berechnungsstärke von Turingmaschinen erzielt. Aufgrund des Nichtdeterminismus von Chomsky-Grammatiken, der eine massive Datenparallelität bei der Ableitung von Wörtern formaler Sprachen zulässt, erscheint eine weitgehende Ausnutzung des Potenzials des DNA-Computing erreichbar.

DNA-Computing kann als innovatives Computingkonzept vor allem dann überzeugen, wenn neben der massiv datenparallelen molekularbiologischen Hardware auch ein großes Repertoire darauf abgestimmter Algorithmen für praxisrelevante Aufgabenstellungen zur Verfügung steht. Klassische Algorithmensammlungen wie [8] notieren die beschriebenen Problemlösungsverfahren gewöhnlich bezogen auf das imperative oder funktionale Programmierparadigma und in einer zumeist auf sequentiell arbeitende Rechnerarchitekturen zugeschnittenen Form. Regelbasierte algorithmische Ideen werden in diesem Sinne kaum betrachtet. Eine Transformation von imperativ oder funktional notierten Algorithmen in

Chomsky-Grammatiken ist zwar konstruktiv möglich [7], so dass diese Verfahren prinzipiell auch mittels DNA-Computing ausgeführt werden können, aber der Vorteil der massiven Datenparallelität bleibt hierbei ungenutzt. In diesem Kontext erscheint es sinnvoll, eine Algorithmenbibliothek auf der Basis von Chomsky-Grammatiken zu erarbeiten, die auf einen effizienten Einsatz im DNA-Computing ausgerichtet ist. Anhand einer Arithmetik für die Grundrechenarten soll ein einfaches Beispiel dafür angegeben werden, wie sich Chomsky-Grammatiken als formales Beschreibungsmittel der Linguistik vorteilhaft in der Algorithmenkonstruktion der Informatik einsetzen lassen.

Die Ausführung numerischer Operationen mittels DNA-Computing ist frühzeitig in den Fokus des wissenschaftlichen Interesses gerückt. Ausgehend von einem ersten laborpraktischen Ansatz zur Addition ([3]) existieren verschiedene Vorschläge, die jedoch die jeweiligen numerischen Operationen losgelöst voneinander betrachten [2]. Infolge der Vielzahl unterschiedlicher genutzter Kodierungen lassen sie sich nicht zur Berechnung umfangreicher Terme ergänzen. Zudem kann ein Berechnungsergebnis nur in wenigen Fällen wieder unmittelbar als Eingabe für nachgeschaltete numerische Operationen dienen. Universelle, frei programmierbare DNA-Computing-Modelle gestatten einen einheitlichen Ansatz. Zu ihnen gehören grammatikbasierte Splicing-Systeme ([5], [4]), die eine Abarbeitung in vitro modellieren, wie auch P-Systeme ([6], [1]), die auf eine Abarbeitung in vivo abzielen. In beiden Fällen wird das auszuführende Programm durch eine Chomsky-Grammatik vorgegeben. Das zur Abarbeitung verwendete System nimmt die schrittweise Ableitung der durch die Chomsky-Grammatik beschriebenen Sprache vor.

2. Grundlagen

Chomsky-Grammatiken sind endliche Erzeugungssysteme für formale Sprachen und beruhen auf einer durch Regeln gesteuerten, fortlaufenden Teilwortersetzung, die ausgehend von einem Startsymbol zu jedem beliebigen Wort der beschriebenen formalen Sprache führt. Eine Chomsky-Grammatik G ist ein Quadrupel $G = (V, \Sigma, P, S)$ mit folgender Bedeutung der Komponenten: V bezeichnet das Alphabet der Nichtterminalsymbole, Σ das Alphabet der Terminalsymbole, die nichtleere endliche Menge $P \subset ((V \cup \Sigma)^* \otimes V \otimes (V \cup \Sigma)^*) \times ((V \cup \Sigma)^*)$ bezeichnet die Regelmenge und $S \in V$ das Startsymbol. Zusätzlich gilt: $V \cap \Sigma = \emptyset$. Grammatikregeln $(u, v) \in P$ notiert man zumeist durch $u \rightarrow v$. Ein Ableitungsschritt (Regelanwendung) von x nach y ist eine Relation $\vdash_G \subset ((V \cup \Sigma)^*) \times ((V \cup \Sigma)^*)$, definiert durch $x \vdash_G y = \{(x, y) \mid x = x_1 u x_2 \wedge y = x_1 v x_2 \wedge \exists x_1, x_2 \in (V \cup \Sigma)^* \,.\, ((u, v) \in P)\}$. Er drückt aus, dass x durch Anwendung genau einer Grammatikregel $(u, v) \in P$ zu y abgeleitet wird. Die Relation $\vdash_G$ ist i.A. keine Funktion, zu einem x kann es mehrere y geben. Die Ausführung von Ableitungsschritten ist deshalb ein nichtdeterministischer Prozess. Die transitive Hülle von $\vdash_G$ wird mit $\vdash_G^+$ bezeichnet, die reflexive transitive Hülle mit $\vdash_G^*$. Die durch G beschriebene (generierte) Sprache $L(G)$ ist definiert durch $L(G) = \{x \in \Sigma^* \mid S \vdash_G^* x\}$. Jede Chomsky-Grammatik lässt sich konstruktiv in Kuroda-Normalform transformieren [7].

3. Darstellung natürlicher Zahlen und Zahlenfolgen

Jede natürliche Zahl $n \in \mathbb{N}$ wird durch das Wort $a^n = \underbrace{a \ldots a}_{n-\text{mal}}$ über dem Alphabet $\Sigma = \{a\}$ repräsentiert, wobei das leere Wort ε die Zahl 0 kodiert. Mithin liegt eine unäre Darstellung vor, bei der die Menge $\mathbb{N}$ der natürlichen Zahlen eineindeutig in die reguläre Menge $\{a\}^*$ abgebildet wird. Als Beispiel für die Erzeugung einer Zahlenfolge dient die Menge der Quadratzahlen.

Erzeugung einer natürlichen Zahl n

Eine natürliche Zahl n lässt sich durch die Chomsky-Grammatik $G_{\#n} = (\{S\}, \{a\}, P_{\#n}, S)$ mit $P_{\#n} = \{S \rightarrow a^n\}$ erzeugen, so dass $L(G_{\#n}) = \{a^n\}$.

Vereinigung von zahlenkodierenden Sprachen

Sei $G_1 = (V_1, \{a\}, P_1, S_1)$ mit $S_1 \in V_1$ und $L(G_1) = \{a^x \mid x \in X \wedge X \subseteq \mathbb{N}\}$ eine Chomsky-Grammatik zur Erzeugung einer beliebigen Teilmenge X der natürlichen Zahlen und sei $G_2 = (V_2, \{a\}, P_2, S_2)$ mit $S_2 \in V_2$ und $L(G_2) = \{a^y \mid y \in Y \wedge Y \subseteq \mathbb{N}\}$ eine Chomsky-Grammatik zur Erzeugung einer beliebigen Teilmenge Y der natürlichen Zahlen. Ferner gelte o.B.d.A.: $V_1 \cap V_2 = \emptyset$. Dann lässt sich daraus eine Chomsky-Grammatik $G_\cup$ zur Generierung der Sprache $L(G_\cup) = L(G_1) \cup L(G_2)$ konstruieren durch $G_\cup = (V_1 \cup V_2 \cup \{S\}, \{a\}, P_1 \cup P_2 \cup \{S \to S_1, S \to S_2\}, S)$, wobei das neue Nichtterminalsymbol $S \notin V_1 \cup V_2$ hinzugenommen wird.

Erzeugung der Menge der Quadratzahlen

Eine Chomsky-Grammatik $G_q = (V_q, \{a\}, P_q, S)$ zur Erzeugung der Menge der Quadratzahlen $L(G_q) = \{a^{(n^2)} \mid n \in \mathbb{N}\} = \{\varepsilon, a, a^4, a^9, a^{16}, a^{25}, \ldots\}$ ist durch $V_q = \{A, B, C, D, E, G, H, L, R, S\}$ und $P_q = \{S \to \varepsilon, S \to ALBREH, LBR \to AALRC, CB \to BC, CE \to EC, LRB \to LBR, LREC \to ALBRBG, GC \to BG, GH \to EH, EH \to D, BD \to D, RD \to D, LD \to D, D \to \varepsilon, A \to a\}$ gegeben.

Die Idee dieser Grammatik folgt der rekursiven Bildungsvorschrift $0^2 = 0$ (Regel $S \to \varepsilon$), $1^2 = 1$ und $(n+1)^2 = n^2 + 2n + 1$. Zur Ableitung jeder Quadratzahl n^2 wird zunächst eine Zwischenform der Gestalt $A^{(n^2)}LB^nREH$ ausgehend von $ALBREH$ für 1^2 induktiv geschaffen. Beim Übergang von einer Quadratzahl n^2 auf ihren Nachfolger $(n+1)^2$ müssen der Zwischenform genau $2n+1$ zusätzliche Symbole A sowie genau ein zusätzliches Symbol B hinzugefügt werden. Diesem Zweck dienen die Regeln $LBR \to AALRC$ und $LREC \to ALBRBG$, wobei die Symbole L, R, E und H als interne Begrenzer innerhalb jeder Zwischenform fungieren. Durch die Regeln $CB \to BC$, $CE \to EC$, $LRB \to LBR$, $GC \to BG$ und $GH \to EH$ wird sichergestellt, dass in jeder Zwischenform die Symbole A und B entsprechend der Vorgabe angeordnet werden und die Bildung jeder Zwischenform abgeschlossen ist, bevor die Zwischenform der nachfolgenden Quadratzahl generiert werden kann. Die Regeln $EH \to D$, $BD \to D$, $RD \to D$, $LD \to D$, $D \to \varepsilon$, $A \to a$ bewirken die Umwandlung der Zwischenform in die korrespondierende Quadratzahl. Hierzu werden die Symbole H, E, R, B und L von rechts beginnend schrittweise abgebaut und jedes A in ein a überführt.

4. Numerische Operationen

Für alle nachfolgend betrachteten numerischen Operationen sei $G_1 = (V_1, \{a\}, P_1, S_1)$ mit $S_1 \in V_1$ und $L(G_1) = \{a^x \mid x \in X \wedge X \subseteq \mathbb{N}\}$ eine Chomsky-Grammatik zur Erzeugung einer beliebigen Teilmenge X der natürlichen Zahlen sowie $G_2 = (V_2, \{a\}, P_2, S_2)$ mit $S_2 \in V_2$ und $L(G_2) = \{a^y \mid y \in Y \wedge Y \subseteq \mathbb{N}\}$ eine Chomsky-Grammatik zur Erzeugung einer beliebigen Teilmenge Y der natürlichen Zahlen, und es gelte $V_1 \cap V_2 = \emptyset$. G_1 und G_2 liegen o.B.d.A. in Kuroda-Normalform vor.

Addition

Zur Erzeugung der Sprache $L(G_{\text{add}}) = \{a^{x+y} \mid x \in X \wedge y \in Y \wedge X, Y \subseteq \mathbb{N}\}$, die eine Addition bewirkt, dient die Chomsky-Grammatik $G_{\text{add}} = (V_1 \cup V_2 \cup \{S\}, \{a\}, P_1 \cup P_2 \cup \{S \to S_1S_2\}, S)$ unter Hinzunahme des neuen Nichtterminalsymbols $S \notin V_1 \cup V_2$.

Aus G_1 mit $L(G_1) = \{a^2, a^5, a^9\}$ und G_2 mit $L(G_2) = \{a, a^6\}$ z.B. entsteht G_{add} mit $L(G_{\text{add}}) = \{a^3, a^6, a^8, a^{10}, a^{11}, a^{15}\}$.

Nichtnegative Subtraktion

Die Sprache $L(G_{\text{nsub}}) = \{a^{x \dot{-} y} \mid x \in X \wedge y \in Y \wedge X, Y \subseteq \mathbb{N}\}$ kann durch die Chomsky-Grammatik $G_{\text{nsub}} = (V_1 \cup V_2 \cup \{A, B, L, S\}, \{a\}, P_1' \cup P_2' \cup \{S \to LS_1S_2, AB \to \varepsilon, LB \to L, L \to \varepsilon, A \to a\}, S)$

mit $P_1' = \{w \to A \mid w \to a \in P_1\} \cup (P_1 \setminus \{w \to a \in P_1\})$ und $P_2' = \{w \to B \mid w \to a \in P_2\} \cup (P_2 \setminus \{w \to a \in P_2\})$ generiert werden, wobei o.B.d.A. gelte: $A, B, L, S \notin V_1 \cup V_2$. Die Regelmenge P_1' entsteht, indem in den Regeln aus P_1 jedes Vorkommen von a durch A ersetzt wird. Analog bildet man die Regelmenge P_2', indem in den Regeln aus P_2 jedes Vorkommen von a durch B ersetzt wird. Die nichtnegative Subtraktion $\dot{-}\colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ führt nicht aus dem Bereich der natürlichen Zahlen hinaus. Es gilt $x \dot{-} y = 0$, falls $x \leq y$.

Aus G_1 mit $L(G_1) = \{a^2, a^5, a^9\}$ und G_2 mit $L(G_2) = \{a^4, a^7\}$ z.B. entsteht G_{nsub} mit $L(G_{\text{nsub}}) = \{\varepsilon, a, a^2, a^5\}$.

Multiplikation

Im Falle einer einelementigen Sprache $L(G_2)$ dient die vereinfachte Chomsky-Grammatik $G_{\text{mul1}} = (V_1 \cup V_2, \{a\}, P_1' \cup P_2, S_1)$ mit $P_1' = \{w \to S_2 \mid w \to a \in P_1\} \cup (P_1 \setminus \{w \to a \in P_1\})$ zur Erzeugung von $L(G_{\text{mul1}}) = \{a^{x \cdot y} \mid x \in X \wedge y \in Y \wedge |Y| = 1 \wedge X, Y \subseteq \mathbb{N}\}$. Zur Bildung von P_1' wird in jeder Regel aus P_1 jedes Vorkommen von a durch S_2 ersetzt.

Für beliebige $L(G_2) \subseteq \{a\}^*$ lässt sich die Chomsky-Grammatik $G_{\text{mul}} = (V_1 \cup V_2 \cup \{A, B, D, I, L, R, S\}, \{a\}, P_1' \cup P_2' \cup \{S \to LS_1S_2R, AR \to R, AB \to BID, DB \to BD, IB \to BID, ID \to DI, IR \to R, LB \to L, L \to \varepsilon, R \to \varepsilon, D \to a\}, S)$ mit $P_1' = \{w \to A \mid w \to a \in P_1\} \cup (P_1 \setminus \{w \to a \in P_1\})$ sowie $P_2' = \{w \to B \mid w \to a \in P_2\} \cup (P_2 \setminus \{w \to a \in P_2\})$ verwenden, wobei o.B.d.A. gelte: $A, B, D, I, L, R, S \notin V_1 \cup V_2$. Zur Bildung von P_1' wird in jeder Regel aus P_1 jedes Vorkommen von a durch A ersetzt und analog zur Bildung von P_2' in jeder Regel aus P_2 jedes Vorkommen von a durch B. Es entsteht die Sprache $L(G_{\text{mul}}) = \{a^{x \cdot y} \mid x \in X \wedge y \in Y \wedge X, Y \subseteq \mathbb{N}\}$.

Der Chomsky-Grammatik G_{mul} liegt die Idee eines inkrementierenden Durchlaufs zugrunde. Aus jeweils zwei miteinander zu multiplizierenden Faktoren m und n wird zunächst mit Hilfe der Regel $S \to LS_1S_2R$ und unter Benutzung der Chomsky-Grammatiken G_1 sowie G_2 die Zwischenform LA^mB^nR generiert. Für jedes einzelne Symbol A werden danach n Symbole D erzeugt und unmittelbar vor dem rechten Begrenzer R plaziert. Dies führt z.B. bei $LAABBBR$ zu der Ableitungsfolge $LAABBBR \vdash_G^+ LABBBDDDR \vdash_G^+ LBBBDDDDDDR$. Nachdem die Symbole L, B und R jeweils vom Rand beginnend entfernt sind und jedes D in ein a umgewandelt wurde, liegt das Multiplikationsergebnis vor. Das Einfügen der Symbole D innerhalb der Zwischenform geschieht durch einen inkrementierenden Durchlauf, der über ein zusätzliches Hilfssymbol I gesteuert wird. Mit der Regel $AB \to BID$ entsteht das erste D hinter dem ersten B. Durch $ID \to DI$ lässt sich das I hinter eine Abfolge von D vor das nächste B schieben. Die Regel $IB \to BID$ ermöglicht es, hinter diesem B wiederum ein zusätzliches D zu erzeugen. Das Symbol I passiert schrittweise ein B nach dem anderen und fügt hinter jedem B genau ein D ein. Über $DB \to BD$ lassen sich die Symbole in der gewünschten Reihenfolge anordnen sowie die eingebrachten Hilfssymbole I vom rechten Rand aus mittels $IR \to R$ entfernen.

Aus G_1 mit $L(G_1) = \{a^2, a^5, a^9\}$ und G_2 mit $L(G_2) = \{a^4, a^7\}$ z.B. entsteht G_{mul} mit $L(G_{\text{mul}}) = \{a^8, a^{14}, a^{20}, a^{35}, a^{36}, a^{63}\}$.

Division

Die Sprache $L(G_{\lceil \text{div} \rceil}) = \{a^{\lceil \frac{x}{y} \rceil} \mid x \in X \wedge y \in Y \setminus \{0\} \wedge X, Y \subseteq \mathbb{N}\}$ kann durch die Chomsky-Grammatik $G_{\lceil \text{div} \rceil} = (V_1 \cup V_2 \cup \{A, B, D, L, M, S\}, \{a\}, P_1' \cup P_2' \cup \{S \to LS_1S_2, BD \to DB, AD \to M, AMD \to M, MB \to DBM, LMD \to LDM, DMD \to DDM, LD \to L, LB \to L, LB \to \varepsilon, M \to a\}, S)$ mit $P_1' = \{w \to A \mid w \to a \in P_1\} \cup (P_1 \setminus \{w \to a \in P_1\})$ sowie $P_2' = \{w \to DB \mid w \to a \in P_2\} \cup (P_2 \setminus \{w \to a \in P_2\})$ generiert werden, wobei o.B.d.A. gelte: $A, B, D, L, M, S \notin V_1 \cup V_2$. Die Regelmenge P_1' entsteht, indem in den Regeln aus P_1 jedes Vorkommen von a durch A ersetzt wird. Analog bildet man die Regelmenge P_2', indem in den Regeln aus P_2 jedes Vorkommen von a durch DB ersetzt wird.

Die Idee der Chomsky-Grammatik $G_{\lceil \text{div} \rceil}$ folgt einer fortgesetzten nichtnegativen Subtraktion. Bei der Division $\lceil \frac{m}{n} \rceil$ (natürlichzahlige Division mit Aufrunden) wird zunächst die Zwischenform $LA^m(DB)^n$ erzeugt. Anschließend wird ermittelt, wie oft der Divisor n im Dividend m enthalten ist, indem ein schrittweiser Abbau des Dividenden in der Zwischenform erfolgt und dabei über ein Markersymbol M mitgezählt wird. Im Fallbeispiel $\lceil \frac{6}{3} \rceil = 2$ führt die Ableitungsfolge $LS_1S_2 \vdash_G^+ LAAAAAADBDBDB$ zur anfänglichen Zwischenform. Mit Hilfe der Regel $BD \rightarrow DB$ lassen sich die Symbole D und B voneinander separieren, so dass $LAAAAAADDDBBB$ entsteht. Die Subtraktion des D-Abschnitts vom A-Abschnitt der Zwischenform über die Regeln $AD \rightarrow M$ und $AMD \rightarrow M$ baut eine Anzahl A vom Dividenden entsprechend der Größe des Divisors ab und fügt ein zusätzliches Markersymbol M ein, im Fallbeispiel $LAAAAAADDDBBB \vdash_G^+ LAAAMBBB$. Mittels eines inkrementierenden Durchlaufes (Regel $MB \rightarrow DBM$) wird das Markersymbol M an den rechten Rand geschoben und für jedes Symbol B wieder genau ein Symbol D hinzugefügt, im Fallbeispiel $LAAAMBBB \vdash_G^+ LAAADBDBDBM \vdash_G^+ LAAADDDBBBM$. Danach kann die nächste Subtraktion auf gleiche Weise geschehen, bis der Dividend vollständig abgebaut ist. Die Regeln $LMD \rightarrow LDM$ und $DMD \rightarrow DDM$ ermöglichen das Aufrunden, wenn ein Divisionsrest verbleibt. Beginnend vom linken Begrenzer L können die übrig gebliebenen Symbole D und B entfernt werden ($LD \rightarrow L, LB \rightarrow L, LB \rightarrow \varepsilon$), so dass aus den M mittels $M \rightarrow a$ das Divisionsergebnis als Wort der Sprache hervorgeht, im Fallbeispiel $LAAADDDBBBM \vdash_G^+ LMBBBM \vdash_G^+ LDBDBDBMM \vdash_G^+ MM \vdash_G^+ aa$.

Aus G_1 mit $L(G_1) = \{a^{10}, a^{12}\}$ und G_2 mit $L(G_2) = \{a^2, a^3\}$ z.B. entsteht $G_{\lceil \text{div} \rceil}$ mit $L(G_{\lceil \text{div} \rceil}) = \{a^4, a^5, a^6\}$. Division durch 0 bei positivem Dividenden generiert kein Wort von $L(G_{\lceil \text{div} \rceil})$. Weitere numerische Operationen lassen sich basierend auf $+$, $\dot{-}$, $\cdot$ und $/$ ebenfalls effizient durch Chomsky-Grammatiken beschreiben. So kann das Potenzieren auf eine fortlaufende Multiplikation und das Logarithmieren auf eine fortlaufende Division zurückgeführt werden.

5. Zusammenfassung

Die vorstehenden Beispiele zeigen, wie man Chomsky-Grammatiken als einheitliches Beschreibungsmittel für numerische Algorithmen einsetzen kann. Die Möglichkeit der nebenläufigen Bearbeitung mehrerer Operanden eröffnet den Zugang zu einer massiv datenparallelen Programmierung und Verarbeitung, die insbesondere für das DNA-Computing geeignet ist [4]. Unabhängig von der Kardinalität der Eingangssprachen $L(G_1)$ und $L(G_2)$ gelingt die Generierung jeder Ergebnissprache dann mit linearem (G_{add}, G_{nsub}) bzw. quadratischem (G_{mul}, $G_{\lceil \text{div} \rceil}$) Zeitaufwand.

- überarbeitete Version vom 21.12.2005

Literatur

[1] **R. Freund, C. Martin-Vide, G. Păun.** *From Regulated Rewriting to Computing with Membranes: Collapsing Hierarchies.* Theoretical Computer Science **312:**143–188, 2004

[2] **P. Frisco.** *Parallel Arithmetic with Splicing.* Romanian Journal of Information Science and Technology **3(2):**113–128, 2000

[3] **F. Guarnieri, M. Fliss, C. Bancroft.** *Making DNA Add.* Science **273:**220–223, 1996

[4] **Th. Hinze, M. Sturm.** *Rechnen mit DNA – Eine Einführung in Theorie und Praxis.* Oldenbourg-Wissenschaftsverlag München, 2004

[5] **G. Păun, G. Rozenberg, A. Salomaa.** *DNA Computing – New Computing Paradigms.* Springer Verlag Berlin, Heidelberg, New York, 1998

[6] **G. Păun.** *Membrane Computing – An Introduction.* Springer Verlag Berlin, 2002

[7] **U. Schöning.** *Theoretische Informatik kurzgefasst.* Spektrum Akademischer Verlag Heidelberg, 1999

[8] **R. Sedgewick.** *Algorithmen.* Zweite Auflage, Addison-Wesley Bonn, München, 2002