Eine DNA-Arithmetik auf der Basis von Chomsky-Grammatiken

Thomas Hinze, Monika Sturm

Arbeitsgemeinschaft „Rechnen mit Molekülen“ an der Technischen Universität Dresden

email: {hinze,sturm}@tcs.inf.tu-dresden.de

www: http://wwwtcs.inf.tu-dresden.de/molecules/

Gliederung

1. **DNA-Computing** als ein innovatives Rechenkonzept
2. **Chomsky-Grammatiken** als ein universelles Berechnungsmodell
3. Einsatz von Chomsky-Grammatiken zur **Algorithmennotation**
4. Darstellung und Erzeugung **natürlicher Zahlen** mittels Chomsky-Grammatiken
5. Chomsky-Grammatiken für **arithmetische Grundoperationen**
6. Grammatikbasierte Sprachgenerierung im DNA-Computing mittels **Splicing-Systemen**
7. Zusammenfassung und **Ausblick**

DNA-Computing

Idee

- Nutzung des Erbmoleküls **DNA** als Datenträger und Speichermedium
- Realisierung von **Operationen** auf DNA durch geeignete **molekularbiologische** und **biochemische Prozesse**, die laborpraktisch **in vitro** (Reagenzglas) oder **in vivo** (lebende Zelle) ablaufen können
- Kodierung der **Eingabedaten** durch spezifische DNA-Moleküle (Synthese)
- Auslesen von **Ergebnisdaten** durch Analyse der aus der Berechnung hervorgegangenen Moleküle

Potenzial

- massive Datenparallelität, hohe Speicherkapazität und -dichte
- Energieeffizienz, Biokompatibilität, Persistenz, Verschleißfreiheit
- Recycelbarkeit, Regenerierbarkeit, Selbstreparaturfähigkeit

$\Longrightarrow$ inzwischen beachtliche Erfolge im praktischen Einsatz

Ziel

- Konstruktion, Aufbau und Anwendung universeller DNA-Computer

$\Longrightarrow$ **Frage nach der Programmierbarkeit universeller DNA-Computer**

Chomsky-Grammatiken als Berechnungsmodell

$\Longrightarrow$ Erzeugungssysteme für **formale Sprachen**

Definition Chomsky Grammatik vom Typ 0 (allgemeinste Form)

$G = (V, \Sigma, P, S)$

V: Alphabet der Nichtterminalsymbole (Variablen)

Σ: Alphabet der Terminalsymbole $\quad V \cap \Sigma = \emptyset$

$P \subseteq ((V \cup \Sigma)^* \cdot V \cdot (V \cup \Sigma)^*) \times ((V \cup \Sigma)^*)$: Regelmenge (Produktionen)

$S \in V$: Startsymbol

- **Ableitungsschritt** (Regelanwendg.): Relation $\vdash_G \subseteq ((V \cup \Sigma)^*) \times ((V \cup \Sigma)^*)$
  $x \vdash_G y = \{(x, y) \mid x = lur \wedge y = lvr \wedge \exists l, r \in (V \cup \Sigma)^*.(u \to v \in P)\}$
- **durch G erzeugte Sprache**: $L(G) = \{x \in \Sigma^* \mid S \vdash_G^* x\}$,
  wobei $\vdash_G^*$ reflexive transitive Hülle von $\vdash_G$
- jede Chomsky-Grammatik konstruktiv in **Kuroda-Normalform** überführbar
- **Berechnung** (Sprachgenerierung) beruht auf einer
  – durch Regeln gesteuerten fortlaufenden Ersetzung von Wortfragmenten
  – ausgehend vom Startsymbol S
- jede **rekursiv aufzählbare Sprache** mittels einer Chomsky-Grammatik vom Typ 0 generierbar (Sprachklasse RE ausgeschöpft, **universell**)

Chomsky-Grammatiken als Berechnungsmodell

Universalität (Turing-Berechenbarkeit)

- universelle Berechnungsmodelle können sich gegenseitig simulieren
- bekannte **Transformationen** universeller Berechnungsmodelle (Auswahl):

- Nachteil der Transformation: **Effizienzverlust** bei Algorithmenabarbeitung

Eigenschaften von Chomsky-Grammatiken (Typ 0) als Berechnungsmodell

- Notationssystem für Algorithmen
- freie Programmierbarkeit durch Universalität gewährleistet
- potentiell unendlich großer Langzeitspeicher unterstellt
- innewohnender Nichtdeterminismus ($\vdash_G$ i.Allg. keine Funktion)
- innewohnende (massive) Datenparallelität

$\Longrightarrow$ kommt der formal-abstrakten Beschreibung der Verarbeitung von DNA im DNA-Computing sehr nahe

$\Longrightarrow$ geeignet zur Konstruktion und Notation DNA-basierter Algorithmen

$\Longrightarrow$ **mögliche Programmiersprache für DNA-Computer**

Chomsky-Grammatiken zur Algorithmennotation

Aspekte zum Einsatz von Chomsky-Grammatiken als Programmiersprache

- ergänzend zum imperativen, funktionalen und logischen Paradigma entsteht ein auf Ersetzungsregeln basierendes **Programmierkonzept**
- Auslotung im Hinblick auf die Konstruktion **zeiteffizienter** Algorithmen, insbesondere für das DNA-Computing, lohnenswert
- Einbringen neuer Herangehensweisen und unkonventioneller Ideen in die **Algorithmenkonstruktion** erscheint vielversprechend
- Erarbeitung einer **Algorithmensammlung**, die praxisrelevante rechen- und /oder speicherintensive Problemklassen der Informatik abdeckt (z.B. kombinatorische Suchprobleme, diskrete Optimierungsaufgaben, semientscheidbare Probleme)
- **numerische Aufgaben** häufig Bestandteil komplexerer Problemstellungen

$\Longrightarrow$ **DNA-Arithmetik für die Grundrechenarten auf natürlichen Zahlen als einfaches Anwendungsbeispiel**

Darstellung und Erzeugung natürlicher Zahlen

Kodierung und Dekodierung

Menge $\mathbb{N}$ der nat. Zahlen $\leftrightarrow$ reguläre Menge $\{a\}^*$ über $\Sigma = \{a\}$

nat. **Zahl** $n \in \mathbb{N}$ entspricht **Wort** $a^n = \underbrace{a \ldots a}_{n-\text{mal}}$, wobei $a^0 = \varepsilon$ (leeres Wort)

Erzeugung einer natürlichen Zahl n

$G_{\#n} = (\{S\}, \{a\}, P_{\#n}, S)$ mit $P_{\#n} = \{S \to a^n\}$

$L(G_{\#n}) = \{a^n\}$

Vereinigung zahlenkodierender Sprachen

- Erzeuge $G_1 = (V_1, \{a\}, P_1, S_1)$, $S_1 \in V_1$ mit $L(G_1) = \{a^x \mid x \in X \wedge X \subseteq \mathbb{N}\}$ eine **beliebige Teilmenge** $X \subseteq \mathbb{N}$ der natürlichen Zahlen.
- Erzeuge $G_2 = (V_2, \{a\}, P_2, S_2)$, $S_2 \in V_2$ mit $L(G_2) = \{a^y \mid y \in Y \wedge Y \subseteq \mathbb{N}\}$ eine **beliebige Teilmenge** $Y \subseteq \mathbb{N}$ der nat. Zahlen (o.B.d.A. $V_1 \cap V_2 = \emptyset$).
- Dann erzeugt $G_\cup = (V_1 \cup V_2 \cup \{S\}, \{a\}, P_1 \cup P_2 \cup \{S \to S_1, S \to S_2\}, S)$ mit dem neuen Symbol $S \notin V_1 \cup V_2$ die Sprache $L(G_\cup) = L(G_1) \cup L(G_2)$ und mithin die **Vereinigung** $X \cup Y$.

$\Longrightarrow$ **auch unendliche Zahlenfolgen durch entsprechende Chomsky-Grammatiken generierbar**

z.B. Menge aller Quadratzahlen, Zweierpotenzen, Primzahlen, ...

Addition

Gegebene Grammatiken G_1 und G_2 zur Operandenerzeugung

- Erzeuge $G_1 = (V_1, \{a\}, P_1, S_1)$, $S_1 \in V_1$ mit $L(G_1) = \{a^x \mid x \in X \wedge X \subseteq \mathbb{N}\}$ eine beliebige Teilmenge $X \subseteq \mathbb{N}$ der natürlichen Zahlen.
- Erzeuge $G_2 = (V_2, \{a\}, P_2, S_2)$, $S_2 \in V_2$ mit $L(G_2) = \{a^y \mid y \in Y \wedge Y \subseteq \mathbb{N}\}$ eine beliebige Teilmenge $Y \subseteq \mathbb{N}$ der nat. Zahlen (o.B.d.A. $V_1 \cap V_2 = \emptyset$).

Konstruktion von G_{add}

- G_1 und G_2 bleiben unverändert
- $L(G_{\text{add}}) = \{a^{x+y} \mid x \in X \wedge y \in Y \wedge X, Y \subseteq \mathbb{N}\}$ erzeugt durch
- $G_{\text{add}} = (V_1 \cup V_2 \cup \{S\}, \{a\}, P_1 \cup P_2 \cup \{S \to S_1 S_2\}, S)$ mit dem neu hinzugenommenen Startsymbol $S \notin V_1 \cup V_2$

Beispiel

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

$\Longrightarrow$ **nebenläufige Auswertung aller Operandenkombinationen**

$\Longrightarrow$ **Vorteil der massiven Datenparallelität bei mehrelementigen Sprachen**

$\Longrightarrow$ **induktiver Aufbau von G_{add} aus G_1 und G_2**

Nichtnegative Subtraktion

Gegebene Grammatiken G_1 und G_2 zur Operandenerzeugung

- Erzeuge $G_1 = (V_1, \{a\}, P_1, S_1)$, $S_1 \in V_1$ mit $L(G_1) = \{a^x \mid x \in X \wedge X \subseteq \mathbb{N}\}$ eine beliebige Teilmenge $X \subseteq \mathbb{N}$ der natürlichen Zahlen.
- Erzeuge $G_2 = (V_2, \{a\}, P_2, S_2)$, $S_2 \in V_2$ mit $L(G_2) = \{a^y \mid y \in Y \wedge Y \subseteq \mathbb{N}\}$ eine beliebige Teilmenge $Y \subseteq \mathbb{N}$ der nat. Zahlen (o.B.d.A. $V_1 \cap V_2 = \emptyset$).

Konstruktion von G_{nsub}

- in P_1 jedes Vorkommen von a durch $A \notin V_1$ ersetzt ($\Rightarrow$ Regelmenge P_1')
- in P_2 jedes Vorkommen von a durch $B \notin V_2$ ersetzt ($\Rightarrow$ Regelmenge P_2')
- $L(G_{\text{nsub}}) = \{a^{x \dot{-} y} \mid x \in X \wedge y \in Y \wedge X, Y \subseteq \mathbb{N}\}$ erzeugt durch
- $G_{\text{nsub}} = (V_1 \cup V_2 \cup \{A, B, L, S\}, \{a\}, P_{\text{nsub}}, S)$,
  $P_{\text{nsub}} = P_1' \cup P_2' \cup \{S \to LS_1S_2, AB \to \varepsilon, LB \to L, L \to \varepsilon, A \to a\}$
  mit den neu hinzugenommenen Symbolen $A, B, L, S \notin V_1 \cup V_2$

Beispiele

- „$3 \dot{-} 2$“: $S \vdash_G^* LAAABB \vdash_G LAAB \vdash_G LA \vdash_G^* a$
- „$2 \dot{-} 3$“: $S \vdash_G^* LAABBB \vdash_G LABB \vdash_G LB \vdash_G^* \varepsilon$
- Aus G_1 mit $L(G_1) = \{a^2, a^5, a^9\}$ und G_2 mit $L(G_2) = \{a^4, a^7\}$ entsteht G_{nsub} mit $L(G_{\text{nsub}}) = \{\varepsilon, a, a^2, a^5\}$.

Multiplikation

Gegebene Grammatiken G_1 und G_2 zur Operandenerzeugung

- Erzeuge $G_1 = (V_1, \{a\}, P_1, S_1)$, $S_1 \in V_1$ mit $L(G_1) = \{a^x \mid x \in X \wedge X \subseteq \mathbb{N}\}$ eine beliebige Teilmenge $X \subseteq \mathbb{N}$ der natürlichen Zahlen.
- Erzeuge $G_2 = (V_2, \{a\}, P_2, S_2)$, $S_2 \in V_2$ mit $L(G_2) = \{a^y \mid y \in Y \wedge Y \subseteq \mathbb{N}\}$ eine beliebige Teilmenge $Y \subseteq \mathbb{N}$ der nat. Zahlen (o.B.d.A. $V_1 \cap V_2 = \emptyset$).

Konstruktion von G_{mul} nach Idee des inkrementierenden Durchlaufs

- in P_1 jedes Vorkommen von a durch $A \notin V_1$ ersetzt ($\Rightarrow$ Regelmenge P_1')
- in P_2 jedes Vorkommen von a durch $B \notin V_2$ ersetzt ($\Rightarrow$ Regelmenge P_2')
- $L(G_{\text{mul}}) = \{a^{x \cdot y} \mid x \in X \wedge y \in Y \wedge X, Y \subseteq \mathbb{N}\}$ erzeugt durch
- $G_{\text{mul}} = (V_1 \cup V_2 \cup \{A, B, D, I, L, R, S\}, \{a\}, P_{\text{mul}}, S)$, $P_{\text{mul}} = P_1' \cup P_2' \cup \{S \to LS_1S_2R, AB \to BID, ID \to DI, IB \to BID, DB \to BD, IR \to R, AR \to R, LB \to L, L \to \varepsilon, R \to \varepsilon, D \to a\}$, neue Symbole: $A, B, D, I, L, R, S \notin V_1 \cup V_2$

Beispiele

- „$2 \cdot 3$“: $S \vdash_G^* LA^2B^3R \vdash_G LAB\underline{ID}BBR \vdash_G LABD\underline{IB}BR \vdash_G LABDB\underline{ID}BR \vdash_G^* LA(BD)^3IR \vdash_G^* LAB^3D^3IR \vdash_G \vdash_G^* \vdash_G^* LB^3D^6I^2R \vdash_G^* a^6$ (inkr. Durchl.)
- Aus G_1 mit $L(G_1) = \{a^2, a^5, a^9\}$ und G_2 mit $L(G_2) = \{a^4, a^7\}$ entsteht G_{mul} mit $L(G_{\text{mul}}) = \{a^8, a^{14}, a^{20}, a^{35}, a^{36}, a^{63}\}$.

Division

Gegebene Grammatiken G_1 und G_2 zur Operandenerzeugung

- Erzeuge $G_1 = (V_1, \{a\}, P_1, S_1)$, $S_1 \in V_1$ mit $L(G_1) = \{a^x \mid x \in X \wedge X \subseteq \mathbb{N}\}$ eine beliebige Teilmenge $X \subseteq \mathbb{N}$ der natürlichen Zahlen.
- Erzeuge $G_2 = (V_2, \{a\}, P_2, S_2)$, $S_2 \in V_2$ mit $L(G_2) = \{a^y \mid y \in Y \wedge Y \subseteq \mathbb{N}\}$ eine beliebige Teilmenge $Y \subseteq \mathbb{N}$ der nat. Zahlen (o.B.d.A. $V_1 \cap V_2 = \emptyset$).

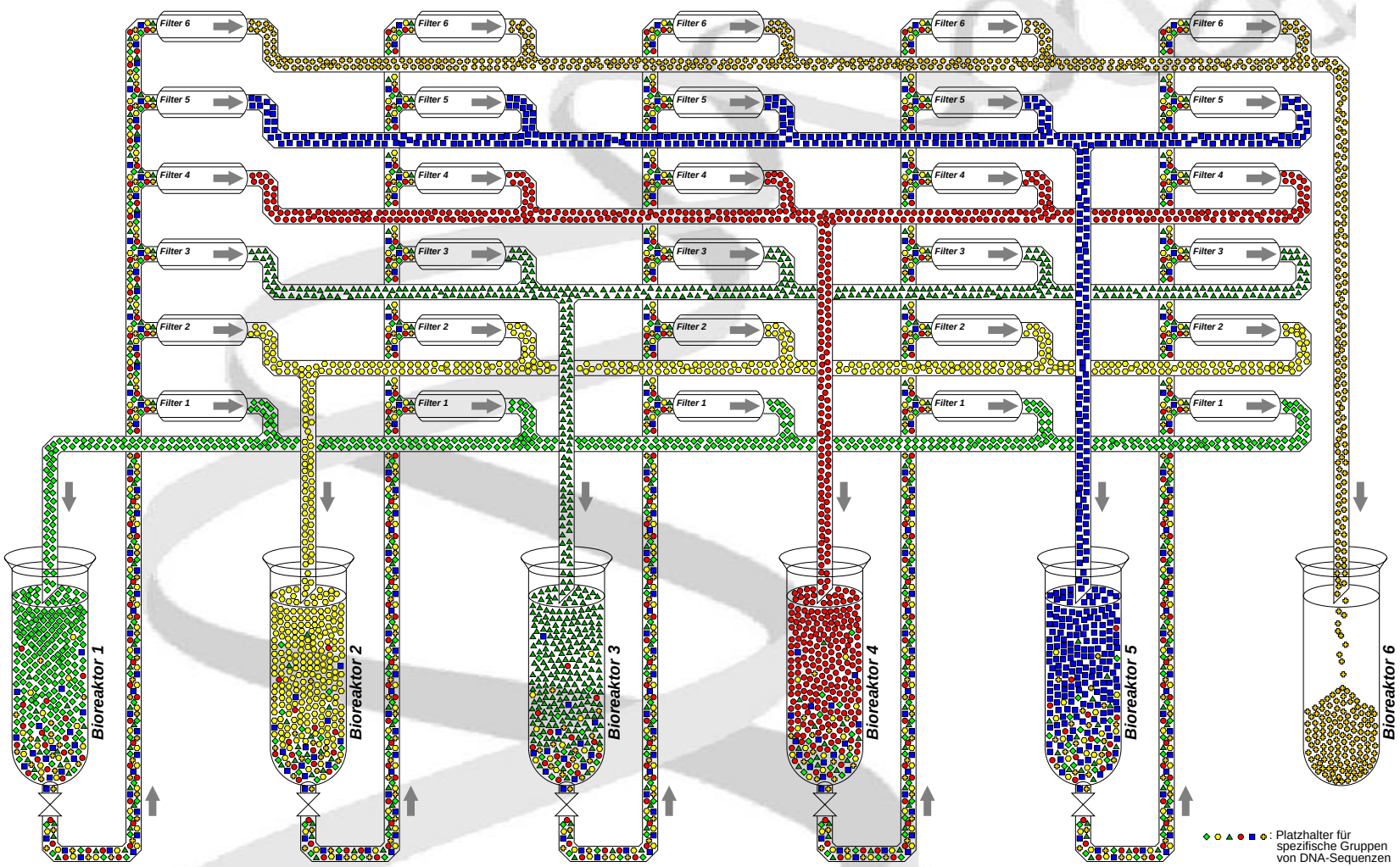
Konstruktion von G_{div} nach Idee der fortlaufenden Subtraktion

- in P_1 jedes Vorkommen von a durch $A \notin V_1$ ersetzt ($\Rightarrow$ Regelmenge P_1')
- in P_2 jedes Vorkommen von a durch $DB \notin V_2^2$ ersetzt ($\Rightarrow$ Regelmenge P_2')
- $L(G_{\text{div}}) = \{a^{\lceil \frac{x}{y} \rceil} \mid x \in X \wedge y \in Y \wedge X, Y \subseteq \mathbb{N}\}$ erzeugt durch
- $G_{\text{div}} = (V_1 \cup V_2 \cup \{A, B, D, L, M, S\}, \{a\}, P_{\text{div}}, S)$, $P_{\text{div}} = P_1' \cup P_2' \cup$ $\{S \to LS_1S_2, BD \to DB, AD \to M, AMD \to M, MB \to DBM, LMD \to LDM,$ $DMD \to DDM, LD \to L, LB \to L, LB \to \varepsilon, M \to a\}$, $A, B, D, L, M, S \notin V_1 \cup V_2$

Beispiele DB-Anordnen, Subtrahieren, inkr. Durchlauf, Aufrunden

- „$\lceil \frac{6}{3} \rceil$“: $S \vdash_G^* LA^6(DB)^3 \vdash_G^* LA^6D^3B^3 \vdash_G^* LA^3MB^3 \vdash_G^* LA^3(DB)^3M \vdash_G^* LA^3D^3B^3M \vdash_G^* LMB^3M \vdash_G^* L(DB)^3MM \vdash_G^* MM \vdash_G^* aa$
- Aus G_1 mit $L(G_1) = \{a^{10}, a^{12}\}$ und G_2 mit $L(G_2) = \{a^2, a^3\}$ entsteht G_{div} mit $L(G_{\text{div}}) = \{a^4, a^5, a^6\}$.

Prinzip eines univ. verteilten Splicing-Systems (TT6)



Zusammenfassung und Ausblick

- durch induktives Konstruktionsprinzip Chomsky-Grammatiken für beliebige arithmetische Terme auf natürlichen Zahlen aufbaubar
- mit den vorgestellten Ideen auch kompliziertere Operationen (z.B. Potenzieren, $\sqrt{\ }$, $\mod$, ...) umsetzbar
- Erstellung einer umfangreichen Algorithmensammlung in Bearbeitung

Weiterführende Informationen zum DNA-Computing

Th. Hinze, M. Sturm.

Rechnen mit DNA

Eine Einführung in Theorie und Praxis.

Oldenbourg Wissenschaftsverlag München,

ISBN 3-486-27530-5, 316 S., 2004

„Das Buch bietet eine umfassende und systematische Einführung in das interdisziplinär geprägte Wissensgebiet des DNA-Computing einschließlich seiner mathematischen wie auch molekularbiologischen Grundlagen. Neben der Vermittlung von Basiswissen zum DNA-Computing werden Modelle, Methoden und Techniken vorgestellt, die eine Realisierung im Labor vorbereiten. Einen Schwerpunkt bildet die labornahe Simulation von Prozessen des DNA-Computing."