Prof. Dr. Hans-Jörg Kreowski **Juni 2007**

Formale Sprachen: DNA Computing

Beispiel: *insdel*-System für $\{a^n b^n c^n \mid n \in I\!N\}$

Die bekannte nicht-kontextfreie, aber kontextsensitive Sprache $\{a^n b^n c^n \mid n \in I\!N\}$ lässt sich durch folgendes *insdel*-System erzeugen:

$$\gamma_{xmpl} = (\{a, b, c, S, B, C, D, K\}, \{a, b, c\}, \{aSbcD, \lambda, abc\}, R_{xmpl})$$

wobei die Regeln in folgender Tabelle stehen:

Nr.	u	α	β	v
1	S	λ	$KaSbc$	bc
2	S	λ	$KabBc$	bc
3	λ	SK	λ	λ
4	cb	λ	Kbc	x
5	λ	cbK	λ	λ
6	Bb	λ	KbB	x
7	λ	BbK	λ	λ
8	B	λ	KC	x
9	λ	BK	λ	λ
10	Cc	λ	KcC	x
11	λ	CcK	λ	λ
12	λ	CD	λ	λ

Das x in den Regeln des Typs 4,6,8 und 10 ist ein von K verschiedenes Zeichen.

Die beiden Startwörter λ und abc sind terminal, sie liefern die gesuchten Wörter für $n = 0$ und $n = 1$. Auf sie kann aber keine Regel angewendet werden. Auf das Startwort $aSbcD$ kann die erste Regel angewendet werden und, da sie die Teilsequenzen Sbc einfügt, kann das beliebig oft wiederholt werden:

$$aSbcD \overset{n}{\underset{(1)}{\Longrightarrow}} a(SKa)^n Sbc(bc)^n D.$$

Die einzige Alternative ist die Einfügung von $KabBc$ zwischen S und bc. Außerdem können – auch schon zwischendurch – alle SK gelöscht werden:

$$aSbcD \overset{n}{\underset{(1)}{\Longrightarrow}} a(SKa)^n Sbc(bc)^n D \overset{n}{\underset{(2)}{\Longrightarrow}} a(SKa)^{n+1} bBc(bc)^{n+1} D \overset{n+1}{\underset{(3)}{\Longrightarrow}} a^{n+2} bBc(bc)^{n+1} D.$$

Mit den Regeln 4 und 5 kann ein Teilwort cb zu bc gedreht werden:

$$ucbv \underset{(4)}{\Longrightarrow} ucbKbcv \underset{(5)}{\Longrightarrow} ubcv.$$

Dabei muss allerdings v mit einem von K verschiedenen Zeichen beginnen. Diese Bedingung ist wichtig, weil sonst die Einfügung mehrmals ausführbar wäre. Insbesondere lässt sich die Vertauschung, mit der Zeichenkette oben beginnend, wiederholen, bis alle b vor allen c stehen:

$$a^{n+2}bBc(bc)^{n+1}D \underset{(4),(5)}{\overset{*}{\Longrightarrow}} a^{n+2}bBb^{n+1}c^{n+2}D.$$

Wie man von Sortieralgorithmen her weiß, sind dafür höchstens quadratisch viele Vertauschungen nötig. Die restlichen Regeln dienen der Überprüfung, dass alle Vertauschungen vorgenommen wurden. Die Regeln 6 und 7 vertauschen Bb zu bB, die Regeln 8 und 9 machen aus B ein C und die Regeln 10 und 11 vertauschen Cc zu cC. Zum Schluss kann man CD mit Regel 12 löschen. Man erhält:

$$\begin{aligned} a^{n+2}bBb^{n+1}c^{n+2}D &\underset{(6),(7)}{\overset{n+1}{\Longrightarrow}} a^{n+2}b^{n+2}Bc^{n+2}D \\ &\underset{(8),(9)}{\Longrightarrow} a^{n+2}b^{n+2}Cc^{n+2}D \\ &\underset{(10),(11)}{\overset{n+2}{\Longrightarrow}} a^{n+2}b^{n+2}c^{n+2}CD \\ &\underset{(12)}{\Longrightarrow} a^{n+2}b^{n+2}c^{n+2}. \end{aligned}$$

Insgesamt lassen sich also auch die weiteren gewünschten Wörter mit Exponenten 2 und größer erzeugen.

Es lässt sich aber auch nicht mehr erzeugen. Denn jede erfolgreiche Ableitung kann in die betrachtete umgeformt werden. Werden beispielsweise Regeln 6 und 7 vor den Regeln 4 und 5 benutzt, kann das auch umgekehrt passieren. Entsprechend kann man die Regeln 6 und 7 vor den Regeln 8 und 9 sowie diese vor den Regeln 10 und 11 anwenden. Die Regel 12 kann nur ganz zum Schluss angewendet werden, weil C erst dann neben D steht. Das B lässt sich nur an den b vorbei nach rechts schieben. Ändert man B zu C, wenn rechts noch ein b kommt, kann das C da nicht vorbeigeschoben werden. Insbesondere kann in einer erfolgreichen Ableitung zum Schluss kein Teilwort cb mehr vorkommen, so dass alle erzeugbaren Wörter die Form $a^m b^n c^p$ haben. Dass $m = n = p$ gilt, liegt an der gleichzeitigen Erzeugung von je einem a, b und c mit den Regeln 1 und 2 und dem Startwort mit je einem a, b und c.