Theoretische Informatik II – 2. Übung

2. Aufgabe

Skizzieren Sie eine Typ-1-Grammatik für die folgende Sprache.

$$L = \{a + b = c \mid a, b, c \in \{0,1\}^n, \quad n \geq 1, \quad \begin{array}{l}\text{Als Binärzahlen interpretiert gilt:}\\ c \text{ ist die Summe aus } a \text{ und } b.\}\end{array}$$

Zunächst einmal ein Beispiel:

	0	1	0	1	0	1	0	a
+	0	0	1	0	1	1	1	b
	1	1	1	1	1	0	0	Übertrag
=	1	0	0	0	0	0	1	c

Das Wort „$101010 + 010111 = 1000001$“ gehört damit zur obigen Sprache.

Idee:

1. Platzhalter für die a, b, c erzeugen,
2. Platzhalter an die richtigen Stellen schieben,
3. Werte für das Ergebnis „berechnen“,
4. $0, 1$ einsetzen.

Problem: Die „Ergebnisbits“ hängen von den Bits ab, die weiter rechts stehen. Je nachdem, ob die Bits von „links nach rechts“ oder von „rechts nach links“ erzeugt werden, sind diese Werte aber noch nicht bekannt oder man muß sich irgendwie merken, was bereits erzeugt wurde.

Betrachten wir noch einmal eine einzelne Bitstelle. Wenn kein Übertrag von rechts kommt, ist das Ergebnis bekannt. Der interessante Fall ist also, wenn ein einlaufender Übertrag auftritt. Hier gibt es drei Fälle:

a_i	b_i	
0	0	Der Übertrag wird „verschluckt“. *(kill)*
0 (1)	1 (0)	Der Übertrag wird „durchgereicht“. *(propagate)*
1	1	Die Stelle erzeugt selbst einen Übertrag. *(generate)*

Das Eingangsbeispiel sieht dann so aus:

	0	1	0	1	0	1	0	a
+	0	0	1	0	1	1	1	b
	k	p	p	p	p	g	p	

Damit können die Bits in 3er-Gruppen erzeugt werden. Das erste aus dem a, b bzw. c Bereich ist speziell markiert. Wir benutzen die folgenden *Nichtterminalsymbole*.

Symbol	Bedeutung
A, a	a-Teil, wird zu 1 bzw. 0
B, b	b-Teil, wird zu 1 bzw. 0
g, p, k	generate, propagate, kill
C, G	wird zu 1
c	wird zu 0
S	Startsymbol
T	Induktion

Platzhalter erzeugen:

$$
\begin{array}{rcl|l|l|}
S & \to & +=TA_\#b_\#C_\# & +=A_\#b_\#C_\# & \\
 & & +=Ta_\#B_\#C_\# & +=a_\#B_\#C_\# & \\
 & & +=Ta_\#b_\#c_\# & +=a_\#b_\#c_\# & \\
 & & +=TA_\#B_\#g_\# & +=A_\#B_\#g_\# &
\end{array}
$$

$$
\begin{array}{rcl|l|}
T & \to & Tabk & abk & \\
 & & TaBp & aBp & \\
 & & TAbp & Abp & \\
 & & TABg & ABg &
\end{array}
$$

Reihenfolge: Die a und die b nach vorn tauschen, + und = dazwischen.

$$
\begin{array}{lcl|lcl|lcl|lcl}
ka & \to & ak & kb & \to & bk & & & & +a & \to & a+ \\
kA & \to & Ak & kB & \to & Bk & & & & +A & \to & A+ \\
pa & \to & ap & pb & \to & bp & ba & \to & ab & +a_\# & \to & a_\#+ \\
kA & \to & Ap & kB & \to & Bp & Ba & \to & aB & +A_\# & \to & A_\#+ \\
ga & \to & ag & gb & \to & bg & bA & \to & Ab & =a & \to & a= \\
gA & \to & Ag & gB & \to & Bg & BA & \to & AB & =A & \to & A= \\
ka_\# & \to & a_\#k & kb_\# & \to & b_\#k & ba_\# & \to & a_\#b & =a_\# & \to & a_\#= \\
kA_\# & \to & A_\#k & kB_\# & \to & B_\#k & Ba_\# & \to & a_\#B & =A_\# & \to & A_\#= \\
pa_\# & \to & a_\#p & pb_\# & \to & b_\#p & bA_\# & \to & A_\#b & =b & \to & b= \\
pA_\# & \to & A_\#p & pB_\# & \to & B_\#p & BA_\# & \to & A_\#B & =B & \to & B= \\
ga_\# & \to & a_\#g & gb_\# & \to & b_\#g & & & & =b_\# & \to & b_\#= \\
gA_\# & \to & A_\#g & gB_\# & \to & B_\#g & & & & =B_\# & \to & B_\#=
\end{array}
$$

Rechnen: Durch die bisherigen Regeln können nur Satzformen erzeugt werden, die ein $c_\#, C_\#$ oder $g_\#$ am Ende stehen haben.

Da das letzte Bit keinen einlaufenden Übertrag hat, ist das Ergebnis bereits bestimmt. Diesen Wert kann man jetzt „hinschreiben“. Gleichzeitig ergibt sich damit der Wert für die Stelle davor. In Abhängigkeit davon, ob diese als p, g oder k erzeugt wurde.

- Ein k kommt aus „$0+0$“. Mit Übertrag von rechts entsteht eine 1, sonst eine 0.
- Ein p kommt aus „$0+1$“ oder „$1+0$“. Mit Übertrag entsteht eine 0 und ein neuer Übertrag, sonst eine 1.
- Ein g kommt aus „$1+1$“. Mit Übertrag entsteht eine 1, sonst eine 0. Außerdem ensteht auf jeden Fall ein neuer Übertrag.

Das ganze läßt sich jetzt für die übrigen Bits fortsetzen.

$$\begin{array}{llll} kc_\# & \to & c_\#0 & pc_\# \to C_\#0 \quad gc_\# \to g_\#0 \\ kC_\# & \to & c_\#1 & pC_\# \to C_\#1 \quad gC_\# \to g_\#1 \\ kg_\# & \to & C_\#0 & pg_\# \to g_\#0 \quad gg_\# \to G_\#0 \\ kG_\# & \to & C_\#1 & pG_\# \to g_\#1 \quad gG_\# \to G_\#1 \end{array}$$

Ersetzen. Zum Schluß müssen noch die restlichen Bits ersetzt werden.

Beachte: Es gibt nur Regeln, die als letztes ein $c_\#$ bzw. $C_\#$ ersetzen. Es kann am Ende also kein weiterer Übertrag entstehen und die a, b und c Teile sind genau gleich lang.

$$\begin{array}{llll} & & & a0 \to 00 \\ =c_\# & \to & =0 & a1 \to 01 \\ =C_\# & \to & =1 & A0 \to 10 \\ b_\#= & \to & 0= & A1 \to 11 \\ B_\#= & \to & 1= & b0 \to 00 \\ a_\#+ & \to & 0+ & b1 \to 01 \\ A_\#+ & \to & 1+ & B0 \to 10 \\ & & & B1 \to 11 \end{array}$$

Ein abschließendes Beispiel.

$$a = 0101010\,, \quad b = 0010111\,, \quad c = 1000001$$

$$\begin{array}{rll} S & \Rightarrow^* & +=\ abk\,Abp\,aBp\,Abp\,aBp\,ABg\,a_\#B_\#C_\# \\ & \Rightarrow^* & +=\ aAaAaAa_\#\,bbBbBBB_\#\,kppppgC_\# \\ & \Rightarrow^* & aAaAaAa_\# + bbBbBBB_\# = kppppgC_\# \\ & \Rightarrow & aAaAaAa_\# + bbBbBBB_\# = kpppg_\#1 \\ & \Rightarrow & aAaAaAa_\# + bbBbBBB_\# = kppg_\#01 \\ & \Rightarrow & aAaAaAa_\# + bbBbBBB_\# = kpg_\#001 \\ & \Rightarrow & aAaAaAa_\# + bbBbBBB_\# = kg_\#0001 \\ & \Rightarrow & aAaAaAa_\# + bbBbBBB_\# = C_\#00001 \\ & \Rightarrow & aAaAaAa_\# + bbBbBBB_\# = 1000001 \\ & \Rightarrow^* & 0101010 + 0010111 = 1000001 \end{array}$$