Technische Universität München
Institut für Informatik
Prof. Tobias Nipkow, Ph.D.
Dr. Stefan Berghofer

WS 2008/09
Lösungsvorschläge zu Blatt 5
19. 11. 2008

Vorlesung Perlen der Informatik

Aufgabe 1 Textersetzungssysteme

(Lösungsvorschlag)

Der Algorithmus verwendet die Zeichen `n` bzw. `c`, um den ersten Summanden vom zweiten Summanden zu trennen. Hierbei bedeutet `n`, daß kein Übertrag zu berücksichtigen ist und `c`, daß ein Übertrag berücksichtigt werden muß. Um den zweiten Summanden vom Ergebnis zu trennen, wird das Zeichen `*` verwendet. Das Ergebnis steht also rechts vom zweiten Summanden. Nach dem Ende einer Iteration erhält man ein Wort der Form

$$a_l \cdots a_{i+1}\, a_i\, \texttt{n}\, b_m \cdots b_{i+1}\, b_i\, \texttt{*}\, c_{i-1} \cdots c_1\, c_0$$

oder

$$a_l \cdots a_{i+1}\, a_i\, \texttt{c}\, b_m \cdots b_{i+1}\, b_i\, \texttt{*}\, c_{i-1} \cdots c_1\, c_0$$

wobei $a_l \cdots a_{i+1}\, a_i$ und $b_m \cdots b_{i+1}\, b_i$ die noch nicht abgearbeiteten Bits der Summanden und $c_{i-1} \cdots c_1\, c_0$ die bereits berechneten Bits des Ergebnisses sind.

Wir geben nun die einzelnen Regeln des Algorithmus an, wobei, falls nichts anderes angegeben ist, die Konvention gelten soll, daß zuerst genannte Regeln Vorrang vor später genannten Regeln haben und jede Regel so weit links wie möglich im aktuellen Wort angewandt wird.

1. Zu Beginn wird das Zeichen `+` durch `n` sowie ein weiteres Hilfszeichen `#` ersetzt. Dieses Hilfszeichen wandert dann so weit wie möglich nach rechts und wird dann schließlich durch das Zeichen `*` ersetzt, das das Ende des zweiten Summanden markiert. Das Zeichen `n` signalisiert, daß zu Beginn noch kein Übertrag zu berücksichtigen ist:

   ```
   +   ⟶  n#
   #0  ⟶  0#
   #1  ⟶  1#
   #   ⟶  *
   ```

2. Danach wird das unterste Bit des zweiten Summanden, d.h. das erste Bit links vom Trennzeichen `*`, nach links geschoben, bis es auf das unterste Bit des ersten Summanden trifft. Hierfür wird links vom Bit, das direkt links vom Trennzeichen `*` steht, das Hilfszeichen `<` eingefügt. Das Hilfszeichen sowie das direkt rechts davon stehende Bit wird dann nach links verschoben:

   ```
   0*   ⟶  <0*
   1*   ⟶  <1*

   0<0  ⟶  <00
   1<0  ⟶  <01
   0<1  ⟶  <10
   1<1  ⟶  <11
   ```

   Hierbei ist wichtig, daß die ersten beiden Regeln die niedrigste Priorität von *allen* Regeln haben, müssen, da sonst unendlich viele `<` produziert werden würden.

3. Wenn das nach links geschobene unterste Bit des zweiten Summanden auf das unterste Bit des ersten Summanden sowie die Markierung `n` oder `c` trifft, werden die beiden Bits unter Berücksichtigung des Übertrags addiert. Die Markierung wird dann entsprechend aktualisiert und das Ergebnis der Addition wird nach rechts bis direkt hinter das Zeichen

`*` geschoben, das den Beginn des Ergebnisses markiert. Zum Verschieben wird das Hilfszeichen `>` analog zum Zeichen `<` eingeführt. Zunächst die Regeln für den Fall, daß kein Übertrag zu berücksichtigen ist:

```
0n<0  ⟶  n0>
1n<0  ⟶  n1>
0n<1  ⟶  n1>
1n<1  ⟶  c0>
```

Ist ein Übertrag zu berücksichtigen, so sehen die Regeln wie folgt aus:

```
0c<0  ⟶  n1>
1c<0  ⟶  c0>
0c<1  ⟶  c0>
1c<1  ⟶  c1>
```

4. Das Hilfszeichen `>` sowie das direkt links davon stehende Bit wird dann nach rechts verschoben. Trifft man auf das Zeichen `*`, so wird das Bit direkt hinter dieses Zeichen geschoben und das Zeichen `>` gelöscht. Danach beginnt die nächste Iteration des Algorithmus wieder bei (2).

```
0>0  ⟶  00>
0>1  ⟶  10>
1>0  ⟶  01>
1>1  ⟶  11>

1>*  ⟶  *1
0>*  ⟶  *0
```

5. Es sind nun noch zwei Randfälle zu betrachten. Ist der erste Summand kürzer als der zweite Summand, so wird das nach links geschobene Bit des zweiten Summanden nur mit dem evtl. vorhandenen Übertrag verknüpft, die Markierung `n` bzw. `c` aktualisiert und dann das Ergebnis wieder nach rechts geschoben. Die Regeln sind analog zu (3):

```
n<0  ⟶  n0
n<1  ⟶  n1>
c<0  ⟶  n1>
c<1  ⟶  c0>
```

Ist der zweite Summand kürzer als der erste Summand, so wird lediglich das unterste Bit des ersten Summanden mit dem evtl. vorhandenen Übertrag verknüpft und das Ergebnis direkt hinter das Zeichen `*` geschoben:

```
0n*  ⟶  n*0
1n*  ⟶  n*1
0c*  ⟶  n*1
1c*  ⟶  c*0
```

6. Sind alle Bits der beiden Summanden abgearbeitet, so wird das Zeichen `*` schließlich gelöscht. Ist noch ein Übertrag zu berücksichtigen, so wird vorne an das Ergebnis noch eine `1` angehängt.

```
n*  ⟶
c*  ⟶  1
```

Das folgende Beispiel zeigt den Ablauf des Algorithmus bei der Addition von `111` und `11`:

```
111+11   ⟶  111n#11   ⟶  111n1#1  ⟶  111n11#  ⟶  111n11*  ⟶
111n1<1*  ⟶  111n<11*  ⟶  11c0>1*  ⟶  11c10>*  ⟶  11c1*0  ⟶
11c<1*0  ⟶  1c1>*0   ⟶  1c*10   ⟶  c*010   ⟶  1010
```