Publ. I.R.M.A. Strasbourg, 1987, 340/S-15
Actes 15^{e} Séminaire Lotharingien, p. 79–83

Addieren ohne Übertrag

von

Heinz Lüneburg

Es sei D eine Menge, $0 \in D$ und s sei eine Abbildung von D in sich. Das Tripel (D,0,s) heißt Dedekindtripel, falls gilt:

1) $0 \notin s(D)$.

2) s ist injektiv.

3) D ist die einzige Teilalgebra der {0,s}-Algebra D.

Die Bedingung 3) ist nichts anderes als das Induktionsprinzip. Ist nämlich $X \subseteq D$, gilt $0 \in X$ und folgt aus $y \in D$, daß auch $s(y) \in D$ gilt, so ist X eine Teilalgebra und folglich X = D.

An dieser Stelle beweist man nun üblicherweise den Rekursionssatz, den wir jedoch nicht einmal formulieren werden, da er allgemein bekannt ist. Mit Hilfe des Rekursionssatzes beweist man nun, daß die folgenden Definitionen in der Tat eine Addition + und eine Multiplikation * auf D definieren.

a + b := if b = 0 then a else s(a) + p(b);

a*b := if b = 0 then 0 else a*p(b) + a;

Dabei ist p(b) der Vorgänger von b, dh., das eindeutig bestimmte Element c von D mit s(c) = b. Man beweist dann, daß + und * alle gewünschten Eigenschaften haben.

Alle Dedekindtripel sind isomorph. Also kann man versuchen, günstige Darstellungen von Dedekindtripeln zu finden, in denen es weniger primitive Algorithmen für die Addition und die Multiplikation gibt als die mit den obigen Definitionen gegebenen. So lernt jeder Europäer schon in frühester Kindheit die Darstellung eines Dedekindtripels im Dezimalsystem und die damit gegebenen, sehr viel schnelleren Additions= und Multiplikationsver= fahren.

Es sei (D,0,s) ein Dedekindtripel. Wir definieren eine Abbildung f des vierfachen cartesischen Produktes von D mit sich selbst in D durch

f(k,a,b,r) := (2**k)*(a + b) + r.

Dann hat f die folgenden Eigenschaften:

a) f(0,s,b,0) = a + b.

b) f(k,0,b,r) = (2**k)*b + r.

c) f(k,a,b,r) = f(k + 1,a/2,b/2,r), falls a und b gerade.

d) f(k,a,b,r) = f(k + 1,(a - 1)/2,(b + 1)/2,r), falls a und b ungerade.

e) f(k,a,b,r) = f(k + 1,(a - 1)/2,b/2,2**k + r), falls a ungerade und b gerade.

f) f(k,a,b,r) = f(k + 1,a/2,(b - 1)/2,2**k + r), falls a gerade und b ungerade.

Die Addition läßt sich also auf folgende Operationen zurückführen: Bestimmung des Nachfolgers und Vorgängers, das Halbieren, die Bestimmung der Parität und das Addieren von 2**k bzw. (2**k)*b zu r: Man scheint vom Regen in die Traufe gekommen zu sein. Dem ist aber nicht so. Notieren wir zunächst einen Algorithmus.

Algorithmus Addition.

Input: Elemente a und b eines Dedekindtripels D.

Output: r ∈ D mit r = a + b.

```
var A, B: D;

begin A := a; B := b; k := 0; r := 0;

      {(2**k)*(A + B) + r = a + b, 0 <= r < 2**k}

      while A <> 0 do

      {(2**k)*(A + B) + r = a + b, 0 <= r < 2**k}

      case par(A) = par(B) of

       true: k := k + 1;
             if par(A) = 1 then B := B + 1;
             A := A div 2;
             B := B div 2;
             endtrue;

      false: r := 2**k + r;
             k := k + 1;
             A := A div 2;
             B := B div 2;
             endfalse;

      endcase;
      endwhile;
```

```
    {(2**k)*B + r = a + b, 0 <= r < 2**k}

    r := (2**k)*B + r;

    {r = a + b}

end;
```

Es geht nun darum, diesen Algorithmus mit Leben zu erfüllen.

Satz 1. Es sei (D,0,s) ein Dedekindtripel. Wir definieren rekursiv eine Abbildung G von D in die Menge FIN(D) der endlichen Teilmengen von D durch: G(0) := {} und

$$G(x) := \{n\} \cup G(2^{**}(n+1) - 1 - x)$$

für $2^{**}n <= x < 2^{**}(n+1)$. Dann ist G eine Bijektion von D auf FIN(D) und es gilt: $G(2*x) \# G(2*x + 1) = \{0\}$ und $G(2*x + 1) \# G(2*x + 2) = \{b + 1\}$ für alle $x \in D$. Dabei ist $b := \min(G(2*x + 1))$ und # bezeichne die symmetrische Differenz zweier Mengen.

Die in Satz 1 definierte Abbildung G ist nichts anderes als der binäre gespiegelte Graycode. Interpretiert man ihn für die charakteristischen Funktionen der endlichen Teilmengen von D, so beginnt die Liste wie folgt:

```
 0:  0
 1:  1
 2:  1  1
 3:  0  1
 4:  0  1  1
 5:  1  1  1
 6:  1  0  1
 7:  0  0  1
 8:  0  0  1  1
 9:  1  0  1  1
10:  1  1  1  1
11:  0  1  1  1
12:  0  1  0  1
13:  1  1  0  1
14:  1  0  0  1
15:  0  0  0  1
```

Satz 1 besagt nun, daß die Addition von 1 niemals einen Übertrag erzeugt. Ferner folgt aus ihm unmittelbar das nächste

Korollar. $|G(n)| = n \bmod 2$ für alle $n \in D$.

Satz 2. Ist $0 < n \in D$, so gilt:

a) Ist n ungerade, so ist $G(2*n) = \{x + 1 \mid x \in G(n)\}$.

b) Ist n gerade, so ist $G(2*n) = \{0\} \cup \{ x + 1 \mid x \in G(n)\}$.

Dieser Satz zeigt, daß das Verdoppeln und dann auch das Halbieren keine Probleme bereitet. Man sieht auch, daß es günstig ist, sich die Parität zu merken, sich Zahlen also durch

$$par\,|\,e_0 \ldots e_t$$

darzustellen mit $e_i \in \{0,1\}$ und $par \in \{0,1\}$, wobei stets

$$par = \Sigma_{i:=0 \text{ to } t}\, e_i \bmod 2$$

gelte. Dann ist also

$$2*(par\,|\,e_0 \ldots e_t) = 0\,|\,par\ e_0 \ldots e_t,$$

bzw.

$$(0\,|\,e_0 \ldots e_t)/2 = e_0\,|\,e_1 \ldots e_t.$$

Schließlich gilt

Satz 3. Sind k, m, n $\in$ D und ist $m < 2**k$, so ist

$$G(m + (2**k)*n) = G(m) \# G((2**k)*n).$$

Hier zwei Beispiele: Es ist 5 = 1|111. Es folgt

$$\begin{aligned} 2*5 &= 0|1111 \\ 4*5 &= 0|01111 \\ 8*5 &= 0|001111 \\ 16*5 &= 0|0001111. \end{aligned}$$

Die Addition von 10 und 16*5 ergibt also 0|1110111.

Es sei a = 9 = 1|1011 und b = 14 = 0|1001. Die folgende Tabelle gibt dann wieder, was bei der Ausführung des Algorithmus' Addition geschieht.

A	B	r	k
1|1011	0|1001	0|0	0
0|1011	1|001	1|1	1
0|111	1|01	1|01	2
1|1	1|1	1|001	3
0|0	1|1	1|001	4

Also ist a + b = 11001 + (2**4)*111 = 11001 + 0100011 = 1100111.

Man wird natürlich k nicht wirklich notieren, sondern an r eine 0 anhängen, wenn par(A) = par(B) ist. Im vorliegenden Falle sähe die letzte Zeile also so aus:

010 111 110010

und die letzte Addition käme dann zustande, indem man 111 und 110010 wie folgt untereinanderschriebe

```
110010
   11
```

und dann die sich überlappenden Komponenten modulo 2 addierte und den verbleibenden Rest von (2**k)*B vorne anhängte.

Es zeigt sich insgesamt, daß das Addieren keinen Übertrag erfordert. Hinzukommt, daß die Bestimmung des Vorgängers nur für ungerade Zahlen erforderlich ist. Dies bedarf nach Satz 1 keiner Suche. Nur in dem Falle, daß A und B beide ungerade sind, ist eine Suche erforderlich, um den Nachfolger von B zu bestimmen.

Auch die anderen Grundoperationen wie Vergleich, Subtraktion, Multiplikation und Division mit Rest, sowie das Umwandeln in dezimal und umgekehrt, lassen sich sehr einfach implementieren.