Grundlagen der Mathematik

Inhaltsverzeichnis

1 Formale Sprachen

1.1 Abstrakte Syntaxbäume

Wenn ein mathematischer Ausdruck syntaktisch analysiert wurde, kann das Ergebnis dieser Analyse als ein sogenannter *abstrakter Syntaxbaum*, kurz *AST* für engl. *abstract syntax tree* dargestellt werden.

Dem Ausdruck »$2x$« ordnen wir z. B. den AST

```
(*,2,x)
```

zu und »$2x+4$« ist dementsprechend

```
(+,(*,2,x),4).
```

Hier ein paar Beispiele:

1. $f(x)$ wird zu `(f,x)`,
2. $f(x,y)$ wird zu `(f,x,y)`,
3. x^2+a wird zu `(+,(^,x,2),a)`,
4. $2(a+b)$ wird zu `(*,2,(+,a,b))`,
5. $a+b+c$ wird zu `(+,(+,a,b),c)`,
6. $a+b+c$ wird zu `(+,a,b,c)`,
7. $x^2=x$ wird zu `(=,(^,x,2),x)`,
8. $A\wedge B$ wird zu `(and,A,B)`.

Bei der Darstellung von abstrakten Syntaxbäumen im Computer gibt es zunächst zwei Varianten. In Lisp wird der AST zum Ausdruck »$f(x,y)$« als verkettete Liste

```
(f x y)
```

dargestellt. Hier kann nämlich mit `first` (auch `car` genannt) der Funktionsbezeichner `f` erhalten werden und mit `rest` (auch `cdr` genannt) die Liste der Argumente.

Bei der Verwendung von dynamischen Feldern anstelle von verketteten Listen sind die Operationen `first` und `rest` eventuell etwas ineffizienter. Hier ist die Darstellung

```
["f", ["x", "y"]]
```

sinnvoller. In einer effizienten statisch typisierten Programmiersprache ließe sich natürlich ein Datentyp für AST-Knoten formulieren, der auf extra die Bedürfnisse, welche sich ergeben, zugeschnitten ist.

2 Lambda-Kalkül

2.1 Variablensubstitution

Angenommen, bei `t` und `u` handelt es sich um Ausdrücke bzw. abstrakte Syntaxbäume. Dann bedeutet die Schreibweise

```
t [x:=u]
```

die Ersetzung jedes Vorkommens der Variable `x` im Baum `t` durch den Baum `u`. Verwendet man keine abstrakten Syntaxbäume, so müssen dabei jedoch im Zusammenhang mit Vorrangregeln die Regeln der Klammersetzung beachtet werden. Z. B. ist

$$a*b\,[b:=x+y]=a*(x+y) \tag{2.1}$$

und nicht $a*x+y$.

Eine solche Variablensubstitution soll außerdem nur für Variablen durchgeführt werden, welche *frei*, d. h. ungebunden vorkommen. Man spricht daher von einer *freien Variablensubstitution*. Variablen können ausschließlich durch lambda-Ausdrücke gebunden werden. Was ein lambda-Ausdruck ist, wird später erklärt.

2.2 Lambda-Ausdrücke

Einer Liste wie `[a,b,c,d]` ordnen wir den AST

```
([],a,b,c,d)
```

zu. Somit gehört zur einelementigen Liste `[x]` der Baum `([],x)`. Zur leeren Liste `[]` gehört der Baum `([])`, wobei die runden Klammern obligatorisch sind. Der Begriff *Liste* ist gleichbedeutend mit *Tupel*.

Einen AST der Form

```
(lambda, ([], x), t)
```

bzw. einen Ausdruck

$$\lambda([x],t) \tag{2.2}$$

wobei `t` ein beliebiger AST ist, wollen wir als lambda-Ausdruck bezeichnen. Anstelle von »$\lambda([x],t)$« schreiben wir kürzer »$|x|\ t$«. Alternative Schreibweisen sind »$\lambda x.t$« oder »$x\mapsto t$«.

Dabei verlangt man nun, dass $|x|$ schwächer bindet als alle anderen Operationen und rechtsassoziativ

ist. Rechtsassoziativ bedeutet, dass die Klammerung immer auf der rechten Seite gesetzt wird. Z. B. ist

$$|x|\ |y|\ x+y = |x|\ (|y|\ x+y). \tag{2.3}$$

Ein lambda-Ausdruck lässt sich nun auf einen AST *applizieren*. Man definiert

$$(|x|\ t)(u)\ =\ t\ [x := u]. \tag{2.4}$$

Z. B. ist

$$(|x|\ x^2)(4) = 4^2 = 16 \tag{2.5}$$

und

$$\begin{aligned}&(|x,y|\ x\cdot y)(a+b,c+d)\\ &= x\cdot y\ [x := a+b,\ y := c+d]\\ &= (a+b)(c+d).\end{aligned} \tag{2.6}$$

Applikation ist linksassoziativ. D. h. es gilt

$$f(x)(y) = (f(x))(y). \tag{2.7}$$