

Grundlagen der Mathematik

Inhaltsverzeichnis

1	Formale Sprachen	1
1.1	Abstrakte Syntaxbäume	1
2	Lambda-Kalkül	1
2.1	Variablensubstitution	1
2.2	Lambda-Ausdrücke	1

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 Kammersetzung beachtet werden. Z. B. ist

$$(a * b) \ [b := x + y] = a * (x + y) \quad (2.1)$$

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

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

bzw. einen Ausdruck

$$\lambda([x], t) \quad (2.2)$$

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

$$|x| |y| x + y = |x| (|y| x + y). \quad (2.3)$$

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

$$(|x| t)(u) := (t [x := u]). \quad (2.4)$$

Z. B. ist

$$(|x| x^2)(4) = 4^2 = 16 \quad (2.5)$$

und

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

Applikation ist linksassoziativ. D. h. es gilt

$$f(x)(y) = (f(x))(y). \quad (2.7)$$

Man kann nun die freie Variablensubstitution präzise definieren. Für Ausdrücke s, t, u und Variablen x, y gelten folgende Regeln:

- (s1) $x [x := u] = u$.
- (s2) $y [x := u] = y$ wenn $x \neq y$.
- (s3) $s(t) [x := u] = (s[x := u])(t[x := u])$.
- (s4) $(|x| t) [x := u] = |x| t$.
- (s5) $(|y| t) [x := u] = |y| (t[x := u])$ wenn $x \neq y$ und $FV(u)$ disjunkt zu $BV(|y| t)$ ist.

Mit $FV(t)$ bezeichnet man die Menge der *freien Variablen* von t , engl. *free variables*. Dieser Operator ist wie folgt definiert:

- 1. $FV(x) = \{x\}$ für jede Variable x .
- 2. $FV(s(t)) = FV(s) \cup FV(t)$.
- 3. $FV(|x| t) = FV(t) \setminus \{x\}$.

Mit $BV(t)$ bezeichnet man die Menge der *gebundenen Variablen* von t , engl. *bounded variables*. Dieser Operator ist wie folgt definiert:

- 1. $BV(x) = \{\}$ für jede Variable x .
- 2. $BV(s(t)) = BV(s) \cup BV(t)$.
- 3. $BV(|x| t) = BV(t) \cup \{x\}$.

Entsprechend können wir noch $AV(t)$, die Menge aller Variablen von t definieren:

- 1. $AV(x) = \{x\}$ für jede Variable x .
- 2. $AV(s(t)) = AV(s) \cup AV(t)$.
- 3. $AV(|x| t) = AV(t) \cup \{x\}$.

Außerdem definieren wir noch $PV(t)$, die Menge der in Ausdrücken präsenten Variablen, wie folgt:

- 1. $PV(x) = \{x\}$ für jede Variable x .
- 2. $PV(s(t)) = PV(s) \cup PV(t)$.
- 3. $PV(|x| t) = PV(t)$.

[Wir hatten gesagt, eine freie Variable sei eine solche, die nicht gebunden ist. Damit ist natürlich gemeint, dass $AV(t)$ disjunkt in $FV(t)$ und $BV(t)$ zerlegt werden kann.] [und wenn frei und gebunden vorkommt?]

Nun gibt es auch lambda-Ausdrücke mit mehr als einem Argument. Wir können nun

$$(|x, y| t) := (|x| |y| t) \quad (2.8)$$

definieren. Man bezeichnet eine solche Umformung als *Currying* oder *Schönfinkeln*.

Bei mehr als einem Argument muss man in solchen Fällen aufpassen, wo die selben Variablen sowohl frei als auch gebunden vorkommen. Der Ausdruck

$$(|x, y| x + y)(y, x) \quad (2.9)$$

unterscheidet sich z. B. von

$$((x + y)[x := y])[y := x]. \quad (2.10)$$

Hier müssen beide Substitutionen »gleichzeitig« vorgenommen werden. Präziser gesagt müssen die gebundenen Variablen vor der Substitution umbenannt werden. Betrachten wir nämlich den geschönfinkelten Ausdruck

$$(|x| |y| x + y)(y)(x). \quad (2.11)$$

Hier erhält man nach Definition nun

$$((|y| x + y)[x := y])(x), \quad (2.12)$$

aber die Substitution kann wegen Regel (s5) nicht ausgeführt werden, denn $FV(y) = \{y\}$ ist nicht disjunkt zu $BV(|y| x + y) = \{y\}$. Benennt man die gebundene Variable y nun in a um, so ergibt sich

$$\begin{aligned} &((|a| x + a)[x := y])(x) \\ &= (|a| y + a)(x) = y + x. \end{aligned} \quad (2.13)$$