3.2 Relationen und Funktionen

Das *ungeordnete Paar aus a und b* ist die Menge $\{a, b\}$. Da nach dem Extensionalitätsaxiom $\{a, b\} = \{b, a\}$ ist, kommt es auf die Reihenfolge oder Anordnung, in der man die Elemente der Menge angibt, nicht an. Oft ist aber eine solche Reihenfolge wichtig: z.B. wenn man Listen von Objekten braucht, die man „der Reihe nach“ bearbeitet.[5]

Definition 3.24 Das *geordnete Paar* (oder: Tupel) *aus a und b* ist die Menge[6]

$$\langle a, b\rangle := \{\{a\}, \{a, b\}\}.$$

Man nennt a die *erste Komponente* und b *die zweite Komponente* von $\langle a, b\rangle$.

Proposition 3.25 Für alle a, b, c, d gilt:

$$\langle a, b\rangle = \langle c, d\rangle \quad \Longleftrightarrow \quad a = c \wedge b = d.$$

Beweis: $\Rightarrow$: Nach der Definition ist $\{a\} \in \langle a, b\rangle = \langle c, d\rangle = \{\{c\}, \{c, d\}\}$. Falls $\{a\} = \{c\}$ ist, so muß $a = c$ sein, und falls $\{a\} = \{c, d\}$ ist, so muß $c \in \{a\}$ sein, also auch $a = c$.

Außerdem ist $\{a, b\} \in \langle a, b\rangle = \langle c, d\rangle$. Falls $\{a, b\} = \{c\}$ ist, so muß $b = c = a$ sein, und daher $\langle a, b\rangle = \{\{a\}, \{a, b\}\} = \{\{a\}\}$. Wegen $\langle c, d\rangle = \langle a, b\rangle$ muß dann auch $\{c, d\} \in \{\{a\}\}$ sein, daher $d = a$, insgesamt also auch $b = d$.

Falls aber $\{a, b\} = \{c, d\}$ ist, so ist entweder $b = d$, also nichts mehr zu zeigen, oder $b = c$. Im letzten Fall ist wegen $a = c$ aber $b = a$, und daraus folgt $b = d$ wie vorhin.

$\Leftarrow$: Mit $a = c$ ist schon mal $\{a\} = \{c\}$, und wenn auch $b = d$ ist, folgt daraus $\{a, b\} = \{c, d\}$. Damit wird $\langle a, b\rangle = \{\{a\}, \{a, b\}\} = \{\{c\}, \{c, d\}\} = \langle c, d\rangle$. □

Es folgt insbesondere, daß für $a \neq b$ auch $\langle a, b\rangle \neq \langle b, a\rangle$ ist, also die Anordnung der Komponenten eine Rolle spielt.

Definition 3.26 Zu Mengen A und B sei

$$A \times B := \{\, x \mid \exists a \exists b (a \in A \wedge b \in B \wedge x = \langle a, b\rangle) \,\}.$$

Man überlege sich, daß dies eine Menge ist (Übungsaufgabe!). Wir schreiben sie auch als

$$A \times B = \{\, \langle a, b\rangle \mid a \in A \wedge b \in B \,\}.$$

[5] Normalerweise wird man eine endliche Menge im Rechner durch eine Liste darstellen, also eine eigentlich unerwünschte Anordnung ins Spiel bringen, von der der Benutzer gar nichts wissen sollte.

[6] Mit $a, b \in X$ sind $\{a\}$ und $\{a, b\}$ Elemente der Menge 2^X, also $\langle a, b\rangle$ eine Teilmenge von 2^X und damit eine Menge.

Damit können wir definieren, was man zur Interpretation von transitiven Verben braucht:

Definition 3.27 Eine *2-stellige Relation* zwischen den Mengen A und B ist eine Menge $R \subseteq A \times B$. Ist $a \in A$, so heißt jedes $b \in B$ mit $\langle a, b\rangle$ ein *Bild von a unter der Relation R.*

Eine *partielle Funktion von A nach B* ist eine Relation $R \subseteq A \times B$ mit

$$\forall a \forall b \forall b' (\langle a, b\rangle \in R \wedge \langle a, b'\rangle \in R \rightarrow b = b'),$$

d.h. eine Relation zwischen A und B, in deren Tupel die zweite Komponente durch die erste Komponente bestimmt ist.

Eine *totale Funktion von A nach B* ist eine partielle Funktion R von A nach B mit

$$\forall a \exists b \ \langle a, b\rangle \in R,$$

d.h. eine partielle Funktion, bei der jedes $a \in A$ ein Bild b unter R hat.

Man schreibt $f : A \rightarrow B$ für „f ist eine Funktion von A nach B“ und $f(a)$ für das (eindeutig bestimmte) Bild von a unter f.

Da Relationen und Funktionen spezielle Mengen sind, erhalten wir mit der Mengengleichheit auch einen Gleichheitsbegriff für Relationen und Funktionen:

Proposition 3.28 Zwei Relationen R und S zwischen A und B sind genau dann gleich, wenn für alle $a \in A$ und $b \in B$ gilt:

$$\langle a, b\rangle \in R \quad \Longleftrightarrow \quad \langle a, b\rangle \in S.$$

Zwei (totale) Funktionen f und g von A nach B sind genau dann gleich, wenn für alle $a \in A$ gilt:

$$f(a) = g(a)$$

3.2.1 Diagonalverfahren

Wir zeigen an zwei Beispielen eine interessante Konstruktion, mit der man ein Objekt konstruieren kann, das nicht zu einer vorgegebenen „Liste“ von Objekten gehört.

Satz 3.29 Die Potenzmenge 2^A einer Menge A hat mehr Elemente als A.

Beweis: Angenommen, 2^A habe nicht mehr Elemente als A. Dann könnten wir jede Teilmenge $B \subseteq A$ mit einem Element a von A benennen, etwa B_a, so daß $2^A = \{ B_a \mid a \in A \}$ wäre. Wir stellen uns eine i.a. unendlich große Tabelle vor, in deren Zeilen wir für jedes Element von A durch ein Bit festhalten, in welchen der Teilmengen es vorkommt:

	B_a	B_b	B_c	B_d	$\cdots$
a					
b			i		
c					
d					
$\vdots$					

Falls z.B. das Element b in der Teilmenge B_c vorkommt, soll das Bit $i = 1$ sein, sonst 0. Sei allgemein das Feld in Zeile x und Spalte y belegt durch

$$f(x, y) := \begin{cases} 1, & \text{falls } x \in B_y, \\ 0, & \text{sonst.} \end{cases}$$

Nun betrachte die Menge

$$B := \{\, x \mid x \in A \wedge f(x, x) = 0 \,\}.$$

Wenn diese Teilmenge von A in der Liste vorkäme, hätte sie einen Namen, etwa c, und $B = B_c$. Dann wäre aber einerseits

$$c \in B \iff f(c, c) = 0,$$

nach der Definition von B, und andererseits

$$c \in B_c \iff f(c, c) = 1,$$

nach der Definition von f, der Tabelle. Aber wegen $B = B_c$ hieße das

$$f(c, c) = 0 \iff f(c, c) = 1,$$

was unmöglich ist. Also kommt B nicht in $\{\, B_a \mid a \in A \,\}$ vor. Daher enthält 2^A mindestens eine Menge „mehr“ als A Elemente hat. □

Wir haben auch für unendliche A gezeigt, daß es keine Funktion $g : A \to 2^A$ gibt, die *jedem* $B \subseteq A$ einen Namen $c \in A$ gibt.

Das zweite Beispiel soll deutlich machen, daß Funktionen und Programme nicht dasselbe sind. Das ist natürlich schon deshalb klar, weil ein Programm festlegt, *wie man vom Argument a zum Wert kommt*, während eine Funktion nur sagt, *was* der Wert $f(a)$ zum Argument a ist.

Wir wollen zeigen, daß man nicht jede Funktion durch ein Programm berechnen kann. Es reicht, wenn wir Funktionen $f : \mathbb{N} \to \mathbb{N}$ betrachten.

Satz 3.30 Nicht für jede Funktion $f : \mathbb{N} \to \mathbb{N}$ gibt es ein Programm P, das für jedes $n \in \mathbb{N}$ bei Eingabe (eines Namens) von n nach endlich vielen Schritten hält und (einen Namen für) den Wert $f(a) \in \mathbb{N}$ ausgibt.

Beweis: Egal, welche Programmiersprache man betrachtet, können wir voraussetzen, daß ein Programm ein endlich langer Text über einem endlichen Alphabet A von Grundzeichen ist. Für jede Länge m gibt es aber nur endlich viele Texte über A mit dieser Länge.

Wir denken uns eine Aufzählung oder Durchnummerierung $\{ P_n \mid n \in \mathbb{N} \}$ aller Programme, wobei z.B. zuerst die endlich vielen Programme der Länge 1, dann die endlich vielen der Länge 2, usw. aufgeführt werden.

Da wir nur an Programmen für totale Funktionen von $\mathbb{N}$ nach $\mathbb{N}$ interessiert sind, können wir annehmen, daß alle Programme bei jeder Eingabe nach endlich vielen Schritten die Ausgabe berechnet haben.

Sei $P_n(k)$ die Zahl, die das Programm P_n bei Eingabe von k schließlich ausgibt. Betrachte die unendliche Tabelle

	P_0	P_1	P_3	P_4	$\cdots$
0					
1			i		
2					
3					
$\vdots$					

bei der im Feld in Zeile k und Spalte n die Ausgabe $P_n(k) \in \mathbb{N}$ steht, die das Programm P_n bei Eingabe von k liefert. Dann hat die durch

$$f(n) := P_n(n) + 1$$

definierte Funktion $f : \mathbb{N} \to \mathbb{N}$ kein Programm. Denn wäre P_k ein Programm, mit dem man f berechnen könnte, so wäre ja

$$P_k(k) = f(k) = P_k(k) + 1,$$

was unmöglich ist. Also kann es zu f kein passendes Programm geben. □

In beiden Beispielen haben wir aus der Annahme, es gebe eine (unendliche) Liste aller uns jeweils interessierenden Objekte –aller Teilmengen einer Menge bzw. aller programmierbaren Funktionen–, ein neues solches Objekt konstruiert, was in der Liste nicht vorkommt.

Dabei wird das neue Objekt mit den Feldern auf der „Diagonale“ der Tabelle konstruiert, den $f(n,n)$ bzw. den $P_n(n)$.

Eine ähnliche Konstruktion, nur mit Hilfe einer endlichen Liste, hat schon Euklid benutzt, um zu zeigen daß es unendlich viele Primzahlen gibt:

Satz 3.31 Es gibt unendlich viele Primzahlen.

Beweis: Angenommen, es gebe nur endlich viele. Sei $p_1, p_2, \ldots, p_n$ eine Liste aller Primzahlen (jede nur einmal). Betrachte $p := p_1 \cdot p_2 \cdot p_3 \cdots p_n + 1$. Da p

größer als jedes p_i ist, kommt es nicht in der Liste vor. Es ist aber eine Primzahl: denn sonst würde es durch eine der Primzahlen, etwa durch p_i geteilt werden können, d.h. es gäbe $k \in \mathbb{N}$ mit $p = k \cdot p_i$ und $k > 1$. Andererseits ist $p-1$ durch p_i teilbar, da $p - 1 = p_1 \cdots p_n$ den Faktor p_i enthält. Also ist auch

$$1 = p - (p-1) = k \cdot p_i - p_1 \cdots p_n = (k - p_1 \cdots p_{i-1} \cdot p_{i+1} \cdots p_n) \cdot p_i$$

durch p_i teilbar, was unmöglich ist. □

Auch hier wird von jedem Objekt der angenommenen „vollständigen“ Liste eine Information benutzt, um ein Objekt derselben Art zu konstruieren, das in der Liste nicht vorkommt.