Skriptum - Mengenlehre (Grundlagen) (v 1.1)

Dieses (lücken)-Skriptum begleitet die Vorlesung Diskrete Mathematik (Mathematik für Informatiker) an der Fachhochschule Vorarlberg. Es ist jedoch nicht vollständig und eignet sich daher (ohne Ergänzungen) nicht zur Vorbereitung auf die Prüfung. Beispiele, Beweise und zum Teil auch textuelle Ergänzungen werden während der Vorlesung gemeinsam erarbeiten.

Die Vorlesung folgt in weiten Teilen dem Buch *Konkrete Mathematik (nicht nur) für Informatiker* von Edmund Weitz [1]. Dieses Buch verfolgt den Ansatz, dass Informatiker*innen in Ihrem späteren Berufsleben mehr angewandte (konkrete) Mathematik benötigen anstelle von „reiner" Mathematik, welche Mathematik der Mathematik wegen betreibt und in der Lehre oft abstrakt und ohne nähere Betrachtung von Anwendungen bleibt. Unser Anwendungswerkzeug der Wahl: Programmieren! Sie lernen in unserer Vorlesung die mathematischen Begriffe, Methoden und vor allem deren Anwendungen, indem Sie diese in Computerprogramme „übersetzen". Wir werden dafür (wie auch Herr Weitz) Python und JupyterLab verwenden.

Das Skriptum enthält mit hoher Wahrscheinlichkeit auch Fehler (typographisch sowie inhaltlich). Bitte melden Sie jeden Fehler der Ihnen auffällt. Ihre Kommilitonen (und natürlich auch ich) werden es Ihnen danken!

Eine Menge ist die grundlegendste Struktur der Mathematik. Alle „mathematischen Objekte“, insbesondere Zahlen und Funktionen, lassen sich durch Mengen repräsentieren.

Die Mengenlehre ist (beispielhaft) wichtig

- für die mathematische Fachsprache (z.B. in wissenschaftlichen Artikeln (auch im Bereich der Informatik), in weiterführenden Vorlesungen, ...)
- als Grundlage der theoretische Informatik
- als grundlegende Datenstruktur für die Programmierung
- ...

Wir machen in unserer Vorlesung „naive Mengenlehre“ == eine Anschauliche Einführung ohne viel Formalismen und keine „axiomatische Mengenlehre“.

Sie sollen hier die grundlegenden Begrifflichkeiten, z.B. den Begriff der Menge, sowie einfache Operationen auf Mengen kennen, verstehen und auch anwenden können. Und auch in diesem Abschnitt werden wir PYTHON einsetzen, um die Theorie mit Leben zu füllen, indem wir einige Dinge selber in kleine Programme umsetzen.

> Eine Menge ist eine Ansammlung von Objekten

z.B.

- Menge Eier im Kühlschrank
- Menge der Informatikstudierenden an der FHV
- Menge an Vorlesungsräumen an der FHV
- Menge von abstrakten Dingen wie z.B. Zahlen
- ...

Bei überschaubaren Mengen werden die Objekten in {} geschrieben umd mit „, “ getrennt.

Welche Objekte gehören zur Menge A?

Jedes Element das man sich vorstellen kann ist entweder Element von A oder eben nicht.

Dies ist die einzige relevante, und somit definierende Eigenschaft der Mengenlehre!

$\Rightarrow$ die Identität einer Menge ergibt sich aus ihren Elementen.

$\Rightarrow$ Die Reihenfolge der Elemente spielt keine Rolle

$\Rightarrow$ Aus

== Die definierten Mengen sind identisch
== es gibt sie nur einmal, wir haben ihr lediglich zwei unterschiedlichen Namen gegeben.

Weiters gilt für

da es das Objekt 3 nur einmal gibt

== entweder ist $3 \in C$ oder nicht.

Achtung:

Sei

Jedoch ist jedes Element von X auch in Y enthalten. Das bedeutet, daß X eine **Teilmenge** von Y ist und man schreibt

wobei Y dann die **Obermenge** von X ist.

Beziehungen von Mengen lassen sich (in einfachen Fällen) durch **Mengendiagramme** darstellen:

Für

schreibt man entsprechend

Mit $\{\}$ bezeichnet man die **leere Menge** (Menge ohne Element) und es gilt

Achtung: in PYTHON wird $\emptyset$ mit `set()`, nicht mit $\{\}$ erzeugt.

Nachdem wir nun Mengen grundlegend definiert haben, können wir uns den Operationen auf Mengen widmen.

Vereinigung $A \cup B$:

Menge, die aus allen Elementen besteht, die in mindestens einer der beiden Mengen A oder B enthalten sind.

z.B.:

Durchschnitt $A \cap B$:

Menge, die aus allen Elementen besteht, die in beiden Mengen A und B enthalten sind.

z.B.:

(mengentheoretische) Differenz $A \setminus B$:

Menge, die aus allen Elementen besteht von A besteht, die nicht Element von B sind.

z.B.:

Zwei Mengen A und B heißen **disjunkt**, falls

== sie haben kein gemeinsames Element

Python: `Mengen` - Definition und Operationen

Folgende Aussagen gelten für alle Mengen A, B und C:

Aus $A \subseteq A = A$ folgt, daß mit $\subseteq$ bei $A \subseteq B$ A und B identisch sein können.

$A \subsetneq B$ (oder auch $A \subset B$) bedeutet hingegen, daß A eine **echte Teilmenge** von B ist, also dass es Elemente in B gibt, die es nicht in A gibt.

Bisher haben wir uns nur mit „kleinen“ Mengen beschäftigt. Für große Mengen ist die Definition durch Aufzählen aller Elemente jedoch nicht praktikabel.

üblich (aber leider etwas unpräzise) ist die Verwendung von Auslassungspunkten $(\ldots)$.

z.B.:

Man hofft hierbei, dass der Leser die Gesetzmäßigkeit erkennt.

Auf die Art kann man mit Mengenoperationen auch „komplizierte“ Mengen bilden:

- ▷ Menge aller ungeraden natürlichen Zahlen kleiner 1000:

- ▷ $\mathbb{N}^+$: Menge aller natürlichen Zahlen ohne 0:

Präziser wie die Definition einer Menge mittels Auslassungspunkten ist die Definition über Eigenschaften.

Die Menge

kann über Eigenschaften eindeutig so definiert werden:

B sei die Menge aller Objekte, die positive natürliche Zahlen sind und außerdem gerade und nicht größer als 80.

Also

Weiteres Beispiel:

X bestehe aus allen Primzahlen, deren Quadrat kleiner als 10000 ist:

Python: `mengen` - Subset und Mengendefinition

$\{x : \ldots\}$ ist also die Menge aller x, die die Eigenschaft rechts vom „:" erfüllt.

== „Menge aller x mit Eigenschaft …"

Statt dem „:" kann auch ein „|" stehen und auch links vom „:" kann ein Teil der Bedingung stehen.

z.B.:

So kann man auch mehr als ein Parameter in der Bedingung links vom „:“ angeben:

$\Rightarrow$ n und d sind „unabhängig voneinander“.
$\Rightarrow$ in A ist jede **Kombination** von n und d enthalten
== auch hier spielt die Reihenfolge wiederum keine Rolle.

Python: Mengen mit zwei Parametern

Mit Mengen können wir nun auch $\mathbb{Z}$ mit Hilfe von $\mathbb{N}$ definieren:

und auch die rationalen Zahlen $\mathbb{Q}$ über die Mengen $\mathbb{Z}$ und $\mathbb{N}^{+}$:

(Siehe Weitz für eine ähnliche Definition von $\mathbb{R}$)

Somit können wir unsere in Abschnitt 1 definierten Zahlen als „Turm“ von Mengen darstellen:

Die Anzahl der Elemente einer (endlichen) Menge A nennt man **Mächtigkeit** (oder **Kardinalität** von A und schreibt dafür $|A|$.

z.B.:

Das **kartesische Produkt** oder **Mengenprodukt** (auch Kreuzprodukt obwohl verwirrend) von zwei Mengen A und B ist die Menge aller **geordneter Paare**, deren erste Komponente $\in A$ und deren zweite $\in B$ ist:

Das geordnete Paar (a, b) nennt man ein **2-Tupel**
$\Rightarrow$ die Reihenfolge bei Tupeln spielt eine Rolle
$\Rightarrow (a, b) \neq (b, a)$ (wenn $a \neq b$)

Beispiele:

In einem n-Tupel „kombiniert“ man Elemente aus n Mengen um eine neue Menge zu konstruieren:

Potenzmenge:

Menge aller Teilmengen von Ausgangsmenge:

z.B.:

Allgemein gilt:

Unendliche Mengen:

Was ist im Universum unendlich?

$\Rightarrow$ Vermutlich nix!!

Unendlichkeit ist allerdings wesentliches Denkkonzept der Mathematik! Zum Beispiel ist die Integral- und somit auch die Differential-Rechnung ohne das Konzept der Unendlichkeit nicht möglich.

Mathematisches Symbol für Unendlich ∞.

Denken Sie über folgende Fragen nach:

- ▷ Wie lange (Zeit) benötigen Sie, um an einen unendlich weit entfernten Ort zu reisen, wenn Sie mit Lichtgeschwindigkeit unterwegs sind?
- ▷ Wie lange benötigen Sie, um ans Ende des Universums zu reisen, wenn Sie unendlich schnell unterwegs sind?
- ▷ Wie lange an einen unendlich weit entfernten Ort, wenn Sie unendlich schnell unterwegs sind?

$\Rightarrow$ ∞ ist keine Zahl!!
$\Rightarrow$ $\infty - \infty$ muß nicht (ist nicht) 0 sein.
$\Rightarrow$ $\frac{\infty}{\infty} \neq 1$ (auch hier: kann, muß aber nicht).

Jedoch gilt

und

(Achtung: eigentlich „nur“ im Grenzwert → siehe zweites Semester)

Ist $\infty \in \mathbb{R}$?

aber

Denkexperiment: Hilberts Hotel.

PYTHON: `infset`

Unser Generator `Nat()` liefert jede natürliche Zahl.

⇒ $\mathbb{N}$ ist **rekursiv aufzählbar**
== es existiert eine Funktion, welche $\mathbb{N}$ „liefert“.

> Allgemein gilt, daß eine Menge rekursiv aufzählbar ist, wenn es eine Funktion gibt, welche die Menge aufzählt.

Ist $\mathbb{Q}$ rekursiv aufzählbar?

Auf den ersten Blick nicht, da ja zwischen zwei rationalen Zahlen wieder unendlich viele andere liegen.

⇒ wie können wir sicherstellen, daß jeder Bruch vorkommt?

Wir nehmen eine Tabelle mit ∞ vielen Zeilen und Spalten und nummerieren sie mit $\mathbb{N}^+$ durch:

Zelle in n-ter Zeile und m-ter Spalte bekommt $\frac{n}{m}$ als Wert.
⇒ Es kommen alle positiven Brüche vor.

Wenn wir diese Tabelle um 45° drehen und die Zellen von oben nach unter und von links nach rechts durchlaufen (== Cantors erstes Diagonalargument), können wir eine PYTHON-Funktion schreiben, welche alle Brüche „aufzählt“

$\Rightarrow \mathbb{Q}$ ist rekursiv aufzählbar

PYTHON: `Rat()`

$\mathbb{R}$ ist hingegen nicht rekursiv aufzählbar und daher „mächtiger“ als $\mathbb{N}$ und $\mathbb{Q}$.

Um dies zu zeigen, benötigen wir jedoch die Definition der Bijektivität.

$\Rightarrow$ wir fahren mit dem Abschnitt Relationen und Abbildungen fort und kehren danach wieder zur Mengenlehre zurück.

Literatur

[1] E. Weitz, Konkrete Mathematik (nicht nur) für Informatiker. 2021, https://www.springer.com/de/book/9783662626177