Skriptum - Relationen und Funktionen (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!

Relationen setzen mehrere Mengen miteinander in Beziehung.

> Eine Relation zwischen zwei Mengen M und N ist eine Teilmenge des Produkts $M \times N$.

$\Rightarrow$

Für $x \in M$ und $y \in N$ sagt man

„x steht (bezüglich R) in Relation zu y “

falls $(x, y) \in R$ und schreibt dafür xRy.

$x \neg Ry$ bedeutet hingegen, daß „x nicht in Relation zu y steht“.

Ist $M = N$ nennt man R eine Relation auf M.

Relationen werden für unterschiedliche Dinge verwendet:

- Vergleich von Zahlen:

- Abbilden einer Menge auf eine andere:

- Zusammenfassen von Elementen einer Menge nach bestimmten Kriterien zu Klassen (== Äquivalenzrelationen $\rightarrow$ später)

Beispiele von Relationen:

1) Menge M der eingeschriebenen FHV-Studierenden, N Menge aller Vorlesungen. Dann ist

   Relation zwischen M und N welche alle geordneten Tupel mit Studierenden und ihren „eingeschriebenen“ Vorlesungen enthält.

2) $A = \{\text{Anton}, \text{Jakob}, \text{Katja}\}$ = „Menge an Software-Entwickler*innen einer Firma“. $B = \{\text{C\#}, \text{Java}, \text{Python}\}$ = „Menge der eingesetzten Programmiersprachen“. Anton beherrscht Java, Katja C# und Java, Jakob Python. Dann gilt

   eine Relation von der man „ablesen“ kann, welchen Mitarbeiter man für welche Programmiersprache einsetzen kann.

3) Sei $A = \{\text{Eier}, \text{Milch}, \text{Honig}\}$, $B = \{\text{Huhn}, \text{Kuh}, \text{Biene}\}$, dann ist

   die Relation „erzeugt von“.

4) Sei $A = \{1, 2\}$, $B = \{1, 2, 3\}$, dann ist

   eine Relation und es gilt

<u>Funktionen</u> (oder Abbildungen) sind Relationen zwischen zwei Mengen M und N, bezüglich der jedes Element $x \in M$ zu genau einem Element $y \in N$ in Relation steht.

Das kennen sie schon aus der Schule!

Jedoch ist diese übliche Schreibweise leider etwas „ungenau“.

Besser wäre

da f ja eine Relation (und somit eine Menge von **geordneten** Paaren) ist.

Und formal ganz korrekt und eindeutig ist

$f(x)$ ist dann ein Funktionswert und eben keine Funktion!

z.B. ist

f hingegen die Funktion selbst in der alle Information steckt.

Funktionen kann man mit Hilfe von Mengendiagrammen visualisieren:

z.B.

Funktionen sind also **rechtseindeutig**:
Es gibt immer nur ein Paar (2-Tupel) (x, y) pro $x \in M$
== es darf keine verschiedenen Paare mit identischer ersten Komponente geben.

PYTHON: `functions`

Eine Funktion f ist **injektiv**, wenn je zwei verschiedene Objekte aus dem Definitionsbereich auch immer verschiedene Funktionswerte haben.

Eine Funktion f ist **surjektiv**, wenn der gesamte Wertebereich „ausgeschöpft“ wird
== alle Werte des Wertebereichs erreicht werden.

Eine Funktion f ist **bijektiv**, wenn sie injektiv und surjektiv ist.

Injektive Funktionen können **invertiert** (umgekehrt) werden.
== die Werte jedes Tupels werden „vertauscht“.

Aus

kann man so

bilden, oder allgemein:

wobei f^{-1} Umkehrfunktion von f heißt.

Nicht injektive Funktionen sind nicht invertierbar, da das Resultat nicht mehr rechtseindeutig wäre
== keine Funktion mehr ist.

Funktionen, die von $\mathbb{R}$ auf $\mathbb{R}$ abbilden kann man durch **Funktionsgraphen** darstellen.

z.B.: Geradengleichung:

Quadratische Gleichung:

Einheitskreis (Kreis mit Mittelpunkt (0,0) und Radius 1):

Wird ein Funktionsgraph von einer horizontaler Linie mehrmals „getroffen“ ist die Funktion nicht injektiv!

Wird ein Funktionsgraph von einer vertikalen Linie mehrmals „getroffen“ ist die Funktion nicht rechtseindeutig, also gar keine Funktion!

Liegt für eine Funktion eine Rechenvorschrift vor, so kann man versuchen, eine Rechenvorschrift für die Umkehrfunktion anzugeben.

Bezeichnet man den zu x gehörenden Funktionswert mit y, so wird die Funktion f durch die Gleichung $y = f(x)$ beschrieben.

z.B.: $y = f(x) = 2 \cdot x - 1$

und es muss gelten

z.B.: $x = 2$ für $y = f(x) = 2 \cdot x - 1$

Dies führt uns zur **Komposition** von Funktionen.
== „nacheinander“ ausführen von zwei Funktionen.

z.B.:

Das geht natürlich auch mit Funktionen die über Mengen definiert sind.

z.B.:

„Nur“ weil g injektiv ist, muß dies nicht zwangsläufig auch für $f \circ g$ gelten. Auch Definitions- und Wertebereich von f, g und $g \circ f$ sind „normalerweise“ unterschiedlich.

Jedoch: sind f und g bijektiv, so ist auch $g \circ f$ bijektiv.

Da jede Menge Definitionsbereich einer Funktion sein kann, können wir auch „mehrstellige“ Funktionen definieren.

z.B.:

Ein Funktionswert ist dann z.B.

$+$ und $\cdot$ auf R sind somit auch zweistellige Funktionen:

Äquivalenzrelation und Äquivalenzklassen

Eine Beziehung, die die Gleichwertigkeit zwischen Objekten unter bestimmten Aspekten ausdrückt, wird **Äquivalenzrelation** genannt. Diese werden verwendet, um Mengen in disjunkte Teilmengen zu unterteilen und finden z.B. ihre Anwendung in der Analyse großer Datenmengen.

> Eine Relation R auf der Menge A heißt Äquivalenzrelation, wenn sie
>
> - ▷ reflexiv
> - ▷ symmetrisch
> - ▷ und transitiv
>
> ist.

Falls aRb mit $a \in A$ und $b \in A$ gilt (also a und b in Relation R stehen), sagt man auch „einfach", dass a (bezüglich R) zu b **äquivalent** ist.

Durch eine Äquivalenzrelation auf einer Menge A erhält man eine **Partition** oder **Klasseneinteilung** von A. Die Menge

bezeichnet man dann als **Äquivalenzklassen** von A bezüglich R wobei a ein beliebiges Element (Vertreter) aus dieser Klasse ist.

Beispiele:

- ▷ a wohnt im selben Ort wie b

- ▷ gerade und ungerade natürliche Zahlen

- ▷ analoge Uhr

Eine Äquivalenzrelation ist also (so wie auch ein Funktion) eine spezielle Relation die man verwenden kann, um eine Menge in Äquivalenzklassen zu „zerlegen“.

Dies sollte als eine sehr kurze Einführung in das Thema Relationen und Funktionen reichen um mit Mengen fortfahren zu können. Lassen Sie uns aber zuerst noch einen Zusammenhang zwischen Relationen und Datenbanken herstellen, damit Sie auch sehen, wie Relationen in der Informatik zum Einsatz kommen.

Relationen und Datenbanken

Relationen bilden auch die Grundlage von relationalen Datenbankmodellen (z.B. verwendet von SQL-Server, PostgreSQL, SQLite, ...), welche immer dann zum Einsatz kommen, wenn Daten („uninterpretierte Information“) in einer bestimmten „Beziehung“ zueinander stehen. Wir haben schon gesehen, dass Relationen auf mehreren Mengen „operieren“ können (z.B. + auf R). Auf die gleiche Art können wir auch eine n-stellige Relation definieren:

Ein Element einer n-stelligen Relation ist dann ein n-Tupel.
In relationalen Datenbanken stellt man solche Relationen in Form von Tabellen dar, wobei ein n-Tupel der Relation dann einer Zeile in der Datenbank entspricht:

Die einzelnen Spalten der Tabelle gehören dabei zu gewissen Mengen (Attribute genannt) (z.B. p.Name, p.Ort, o.Id, ...).

Eine Zeile entspricht einem Element der **Produktmenge**

z.B. (konkrete Zeile ist Element von R_P)

Wenn wir eine zweite Tabelle `Ort` definieren

in der jede Zeile Element der Menge

ist, können wir Relationen auf bzw. zwischen den beiden Relationen R_P und R_O bilden und so bestimmte Daten Abfragen. Die Menge von Operationen zur Manipulation von solchen „Datenbank-Relationen" (Filtern, Verknüpfen, Aggregieren,...) bilden eine **relationale Algebra** oder **Relationenalgebra** und werden z.B. von der „Structured Query Language" (SQL) implementiert.

z.B. ist das `select` (Abfragen von Daten welche eine bestimmte Bedingung erfüllen) folgende Relation:

Auch eine `join`-Operation ist eine Relation, allerdings auf zwei Relationen (== zwei Tabellen):

Für weitere Operationen und Details sei hier auf die einschlägigen Spezialvorlesungen verwiesen.

Literatur

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