Abschnitt 4: Daten und Algorithmen

4. Daten und Algorithmen

Roadmap

- Wir nehmen unsere Diskussion über den Algorithmus-Begriff wieder auf und konkretisieren einige wichtige Aspekte.
- Wir beschäftigen uns zunächst mit dem Aspekt der eindeutigen Darstellung der zu verarbeitenden Daten
- Anschließend diskutieren wir anhand einem Beispiel wesentliche Eigenschaften von Algorithmen und unterschiedliche Herangensweisen in der Algorithmenentwicklung.

Überblick

Daten

- Wir betrachten zunächst die Daten (Objekte), die durch Algorithmen verarbeitet werden sollen.
- Wir wollen insbesondere klären, wie diese Daten dargestellt werden.
- Typische Daten sind Zahlen, z.B. die Zahl “drei”, die wie folgt dargestellt werden kann:
  - 3
  - DREI
  - III
  - drei ausgestreckte Finger einer Hand
  - ...

Datendarstellung

- Wir unterscheiden bei einem Objekt
  - die Darstellung, (*Syntax*, “Bezeichnung”),
  - seine Bedeutung, (*Semantik*, “Information”).
- Einige Datendarstellungen sind für maschinelle Verarbeitung nicht geeignet.
- Alle geeigneten Datendarstellungen beruhen auf dem Grundprinzip der *Zeichenreihe*, die wir im folgenden formal definieren.

Alphabet

- Ein *Alphabet* $\mathcal{A}$ ist eine endliche Menge, deren Elemente *Zeichen* genannt werden.
- Beispiele:
  - Menge der Großbuchstaben: $\{$A,B,C, . . .,Z$\}$
  - Menge der Dezimalziffern: $\{$1,2,3, . . .,9$\}$
  - Menge der Vorzeichen: $\{+, -\}$
  - Menge der Richtungszeiger eines Lifts: $\{\uparrow, \downarrow\}$
- Alphabete, die genau zwei Zeichen enthalten heißen *binär*.
- Ein wichtiges binäres Alphabet besteht aus den *Binärziffern* (*Bits*) $\{0, 1\}$.

Zeichenreihe

- Eine *Zeichenreihe* über einem Alphabet $\mathcal{A}$ ist eine (endliche) Folge von Zeichen aus $\mathcal{A}$.
- Formal ist auch die leere Folge eine Zeichenreihe.
- Wir schreiben Zeichenreihen/Folgen $(x_1, x_2, \ldots, x_n)$ auch als $x_1 x_2 \ldots x_n$.
- Beispiele:
  - Sei $\mathcal{A}_1 = \{$A,B,C,...,Z$\}$
    - Die Folge INFORMATIK ist eine Zeichenreihe über $\mathcal{A}_1$.
    - Die Folge (S,C,H,M,I,D) ist eine Zeichenreihe über $\mathcal{A}_1$.
    - Die Folgen Kröger und (Z,Ü,F,L,E) sind *keine* Zeichenreihen über $\mathcal{A}_1$.
      Warum?
  - Sei $\mathcal{A}_2 = \{0, 1\}$
    - Die Folge 0 ist eine Zeichenreihe über $\mathcal{A}_2$.
    - Die Folge 1 ist eine Zeichenreihe über $\mathcal{A}_2$.
    - Die Folge 01 ist eine Zeichenreihe über $\mathcal{A}_2$.
    - Die Folge 10 ist eine Zeichenreihe über $\mathcal{A}_2$.
    - Die Folge 11 ist eine Zeichenreihe über $\mathcal{A}_2$.
    - Die Folge 00 ist eine Zeichenreihe über $\mathcal{A}_2$.
    - ...

Bezeichnung von Daten

- Wir verwenden ausschließlich Zeichenreihen zur Bezeichnung von Daten.
- Im folgenden betrachten wir als Beispiel die Darstellung von natürlichen Zahlen, also Elementen der Menge $\mathbb{N}_0$.
- Die Zahl “dreizehn” lässt sich u.a. durch folgende Zeichenreihen bezeichnen:

13	$(\mathcal{A} = \{0, 1, 2, \ldots, 9\})$,
DREIZEHN	$(\mathcal{A} = \{\text{A,B,}\ldots\text{,Z}\})$,
||||||||||||||	$(\mathcal{A} = \{\|\})$.

- Nicht alle diese Darstellungen sind für den praktischen Gebrauch (z.B. Rechnen) geeignet.

Zifferndarstellung / p-adische Zahlendarstellung

- Am besten geeignet ist die *Zifferndarstellung*, z.B. die allgemein gebräuchliche *Dezimaldarstellung* über dem Alphabet $\{0, 1, 2, \ldots, 9\}$.
- Das allgemeine Prinzip der Zifferndarstellung ist wie folgt definiert:
  - Sei $p \in \mathbb{N}$, $p \geq 2$ und $\mathcal{A}_p = \{z_0, z_1 \ldots, z_{p-1}\}$ ein Alphabet mit p Zeichen (genannt *Ziffern*) $z_0, z_1 \ldots, z_{p-1}$.
  - Die Funktion $Z : \mathcal{A}_p \to \mathbb{N}_0$ bildet jedes Zeichen aus $\mathcal{A}_p$ auf eine natürliche Zahl wie folgt ab:
    $$Z(z_i) = i \qquad \text{für } i = 0, \ldots, p-1.$$
  - Eine Zeichenreihe $x = x_n x_{n-1} \ldots x_1 x_0$ (der Länge $(n+1)$) über $\mathcal{A}_p$ (d.h. $x_i \in \mathcal{A}_p$ für $0 \leq i \leq n$) bezeichnet die Zahl
    $$\mathcal{Z}(x) = p^n \cdot Z(x_n) + p^{n-1} \cdot Z(x_{n-1}) + \ldots + p \cdot Z(x_1) + Z(x_0).$$
- Zur Verdeutlichung schreiben wir auch x_p. x_p heißt *p-adische Zahlendarstellung* der Zahl $\mathcal{Z}(x_p) \in \mathbb{N}_0$.

Zifferndarstellung / p-adische Zahlendarstellung

- Nochmal die Formel

$$\mathcal{Z}(x) = p^n \cdot Z(x_n) + p^{n-1} \cdot Z(x_{n-1}) + \ldots + p \cdot Z(x_1) + Z(x_0).$$

- Beispiele:
  - Mit $p = 10$ und $\mathcal{A}_{10} = \{z_0, z_1 \ldots, z_9\}$ erhält man die Dezimaldarstellung wenn man statt z_i gleich $Z(z_i)$ schreibt (also z.B. statt z_3 schreibe $Z(z_3) = 3$):

    $\mathcal{Z}(983_{10}) = 10^2 \cdot 9 + 10^1 \cdot 8 + 10^0 \cdot 3 =$ “neunhundertdreiundachtzig”.

    (Wir schreiben direkt $\mathcal{A}_{10} = \{0, 1 \ldots, 9\}$)

Zifferndarstellung / p-adische Zahlendarstellung

- Nochmal die Formel

$$\mathcal{Z}(x) = p^n \cdot Z(x_n) + p^{n-1} \cdot Z(x_{n-1}) + \ldots + p \cdot Z(x_1) + Z(x_0).$$

- Beispiele (cont.):
  - $p = 2$ und $\mathcal{A}_2 = \{0, 1\}$ (*Binärdarstellung*):
    $\mathcal{Z}(1111010111_2) = 2^9 \cdot 1 + 2^8 \cdot 1 + 2^7 \cdot 1 + 2^6 \cdot 1 + 2^5 \cdot 0 + 2^4 \cdot 1 + 2^3 \cdot 0 + 2^2 \cdot 1 + 2 \cdot 1 + 1 =$
    "neunhundertdreiundachtzig".
  - $p = 8$ und $\mathcal{A}_8 = \{0, 1, 2, 3, 4, 5, 6, 7\}$ (*Oktaldarstellung*):
    $\mathcal{Z}(1727_8) = 8^3 \cdot 1 + 8^2 \cdot 7 + 8 \cdot 2 + 7 =$ "neunhundertdreiundachtzig".
  - $p = 16$ und $\mathcal{A}_{16} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}$
    (*Hexadezimaldarstellung*):
    $\mathcal{Z}(3D7_{16}) = 16^2 \cdot 3 + 16 \cdot 13 + 7 =$ "neunhundertdreiundachtzig".

Zifferndarstellung / p-adische Zahlendarstellung

- Führende Nullen sind in der Definition der p-adischen Zahldarstellung zugelassen, z.B. ist die Zeichenreihe

  000983

  eine zulässige Dezimaldarstellung und bezeichnet die gleiche Zahl wie die Zeichenreihe 983.
- Offensichtlich können führende Nullen (d.h. führende Ziffern 0, also die Ziffer z_0) immer weggelassen werden, außer bei der Bezeichnung “0” für die Zahl “null”.
- Für jede Zahl aus $\mathbb{N}_0$ gibt es für beliebiges $p \geq 2$ eine p-adische Darstellung.
- Betrachtet man nur Darstellungen ohne führende Nullen, so ist (zu festem $p \geq 2$) die p-adische Zahldarstellung eindeutig.

Menschengerechte Darstellung von Daten

- Zur Entwicklung von Algorithmen werden wir typischerweise die Dezimaldarstellung der natürlichen Zahlen verwenden.
- Allgemein gibt es für die meisten Daten eine *Standarddarstellung* bei denen die Lesbarkeit der Darstellung für den menschlichen Benutzer im Vordergrund steht.
- Wir verwenden hier folgende Standardbezeichnungen wie sie in den üblichen höheren Programmiersprachen gebräuchlich sind:

Natürliche Zahlen	$\mathbb{N}_0$	Dezimaldarstellung (ohne führende Nullen)
Ganze Zahlen	$\mathbb{Z}$	wie natürliche Zahlen, ggf. mit Vorzeichen "-", z.B. 3,-3.
Reelle Zahlen	$\mathbb{R}$	*Gleitpunktdarstellung*, siehe später.
Wahrheitswerte	$\mathbb{B}$	*TRUE* und *FALSE* für "wahr" bzw. "falsch".

Darstellung von Zeichen und Texten

- Eine Dezimaldarstellung (z.B. 983) stellt eine natürliche Zahl dar, die verarbeitet werden kann.
- Die Zeichenreihe selbst (und nicht die dargestellte Zahl) könnte aber auch Gegenstand der Verarbeitung sein.
- Zeichenreihen können also nicht nur Darstellungen von Objekten sein, sondern auch selbst Objekte, die dargestellt werden müssen.
- Zeichenreihen heißen in diesem Zusammenhang *Texte*.
- In der Praxis wird zur Bildung von Texten häufig das sog. *ASCII-Alphabet* benützt.
- Das ASCII-Alphabet repräsentiert eine Menge von Zeichen, die wir im folgenden als $CHAR$ bezeichnen.
- Die Elemente von $CHAR$ finden Sie in allen gängigen Lehrbüchern.

Maschinengerechte Darstellung von Daten

- Aus technischen Gründen kann die kleinste Speichereinheit eines Computers (das *Bit*) nur zwei Zustände speichern:
  - Zustand 1: es liegt (elektr.) Spannung an.
  - Zustand 0: es liegt keine Spannung an.
- Daher werden Werte (Daten/Objekte) als Bitmuster (Zeichenreihe über dem Alphabet $\mathcal{A}_2 = \{0, 1\}$) codiert gespeichert (auch Darstellungen über $\mathcal{A}_8$ und $\mathcal{A}_{16}$ sind mit diesen technischen Gegebenheiten gut vereinbar).
- Intern kann der Computer also z.B. die natürlichen Zahlen in Binärcodierung repräsentieren.
- Ganze Zahlen können intern ebenfalls leicht als Zeichenkette über dem Alphabet $\mathcal{A}_2 = \{0, 1\}$ codiert werden (z.B. mit einem führenden Bit für das Vorzeichen; Genaueres darüber werden Sie in der Vorlesung “Rechnerarchitektur” lernen).

Maschinengerechte Darstellung von Daten

- Die Binärdarstellung der reellen Zahlen ist etwas komplizierter.
- Die *Gleitpunktdarstellung* einer Zahl z ist

$$z = m \cdot 2^e,$$

  wobei sowohl m (*Mantisse*) als auch e (*Exponent*) binär repräsentiert werden (können).
- In vielen Programmiersprachen (z.B. Java) wird eine Zahl dargestellt durch $z = m \cdot 10^e$, z.B. 3.14, -7.45, $1.33E-2$ (für $1.33 \cdot 10^{-2}$).
- Eine genaue Spezifikation lernen wir im nächsten Abschnitt kennen.
- Wichtig: Für viele reelle Zahlen gibt es gar keine derartige Darstellung (z.B. für $\sqrt{2}$). Die darstellbaren Zahlen heißen auch *Gleitpunktzahlen*.
- Dieser Aspekt der maschinengerechten Darstellung von Daten ist bei der Entwicklung von Algorithmen möglicherweise wichtig! Warum?

Maschinengerechte Darstellung von Daten

- Ein weiterer Aspekt der maschinengerechten Darstellung ist, dass die Ressourcen (Speicherzellen) einer Rechenanlagen begrenzt sind.
- Es stehen daher für die Darstellung von Daten immer nur endlich viele Bits zur Verfügung.
- D.h. zur Darstellung einer beliebigen natürlichen Zahl stehen nur Zeichenketten mit fixer *Länge* zur Verfügung (notfalls mit führenden Nullen).
- Werte, deren Darstellung mehr Zeichen (Bits) benötigt würden, können somit nicht dargestellt werden.
- Die Länge der zur Verfügung stehenden Zeichenketten hat damit offenbar Einfluss auf den Wertebereich der darstellbaren Objekte.
- Dies ist bei der Entwicklung von Algorithmen möglicherweise ebenfalls wichtig! Warum?

Überblick

Syntax

- Die Gestalt einer Datendarstellung nennt man *Syntax*.
- Die Bedeutung der dargestellten Objekte heißt *Semantik*.
- Die Syntax von Darstellungen von Daten/Objekten kann ohne Bezugnahme auf die Semantik definiert werden.

Bsp.: Syntaxdefinition von natürlichen Zahlen

- Die Menge der natürlichen Zahlen in Dezimaldarstellung kann durch folgende Regeln definiert werden:
  1. 0 ist eine $\langle$*Dezimalzahl*$\rangle$.
  2. Jede Ziffer $x \in \mathcal{A}_{10} \setminus \{0\}$ ist eine $\langle$*Nichtnulldarstellung*$\rangle$.
  3. Ist a eine $\langle$*Nichtnulldarstellung*$\rangle$ und $y \in \mathcal{A}_{10}$ so ist $a \circ y$ eine $\langle$*Nichtnulldarstellung*$\rangle$.
  4. Jede $\langle$*Nichtnulldarstellung*$\rangle$ ist eine $\langle$*Dezimalzahl*$\rangle$.
- Beispiel:
  Die Zeichenreihe 308 ist eine Dezimalzahl gemäß folgender Anwendungen der Regeln:

  3 ist $\langle$*Nichtnulldarstellung*$\rangle$ nach Regel 2
  30 ist $\langle$*Nichtnulldarstellung*$\rangle$ nach Regel 3
  308 ist $\langle$*Nichtnulldarstellung*$\rangle$ nach Regel 3
  308 ist $\langle$*Dezimalzahl*$\rangle$ nach Regel 4

Bsp.: Syntaxdefinition von natürlichen Zahlen

- Die Begriffe ⟨*Dezimalzahl*⟩ und ⟨*Nichtnulldarstellung*⟩ in den Regeln werden *syntaktische Variablen* genannt. Sie kennzeichnen den zu definierenden Begriff sowie einen Hilfsbegriff[3].
- Allgemein können in Syntaxdefinitionen noch mehr syntaktische Variablen vorkommen. Wir heben sie durch kursive Schrift und die eckigen Klammern hervor.
- Die Zeichen des vorgegeben Alphabets heißen *Terminalzeichen*.

[3] Der Hilfsbegriff ⟨*Nichtnulldarstellung*⟩ ist im Übrigen induktiv definiert.

Backus-Naur-Form

- Eine formale, häufig gebrauchte Form von Syntaxdefinitionen ist die *Backus-Naur-Form (BNF)*
- Eine Syntaxdefinition in BNF is gegeben durch eine endliche Anzahl von *Syntaxregeln* der Form

$$\alpha ::= \beta$$

- Dabei ist α eine syntaktische Variable und β eine *BNF-Satzform* (induktive Definition auf der nächste Folie).
- Eine ausgezeichnete syntaktische Variable ist das *Startsymbol* α_S.

BNF-Satzform

- Syntaktische Variablen und Terminalzeichen sind BNF-Satzformen.
- Auswahl (Alternative):
  Sind $\gamma_1, \ldots, \gamma_n$ BNF-Satzformen, dann auch $\gamma_1 | \ldots | \gamma_n$.
- Verkettung (Konkatenation):
  Sind $\gamma_1, \ldots, \gamma_n$ BNF-Satzformen, dann auch $\gamma_1 \ldots \gamma_n$.
- Klammerung:
  Ist γ eine BNF-Satzform, dann auch (γ).
- Iteration:
  Ist γ eine BNF-Satzform, dann auch $(\gamma)^*$.
- nichtleere Iteration:
  Ist γ eine BNF-Satzform, dann auch $(\gamma)^+$.
- Option:
  Ist γ eine BNF-Satzform, dann auch $[\gamma]$.

Bedeutung (Semantik) der BNF

- Eine Zeichenreihe w ist *herleitbar* mit einer BNF Syntaxdefinition, wenn man w aus α_S durch eine oder mehrere der folgenden Ersetzungsoperationen (siehe nächste Folie) erhalten kann
- Wir schreiben auch $\alpha_S \rightarrow w$.

Ersetzungsoperationen der BNF

Eine syntaktische Variable α, für die die Regel $\alpha ::= \beta$ vorhanden ist, darf wie folgt ersetzt werden (die erlaubten Ersetzungen von α sind rekursiv über die Gestalt von β definiert):

- Ist β eine syntaktische Variable δ, so darf α mit δ ersetzt werden.
- Ist β ein Terminalzeichen $x \in \mathcal{A}$, so darf α mit x ersetzt werden.
- Hat β die Form $\gamma_1 | \ldots | \gamma_n$ (Auswahl), so darf α durch eines der γ_i $(1 \leq i \leq n)$ ersetzt werden.
- Hat β die Form $\gamma_1 \ldots \gamma_n$ (Verkettung), so darf α durch $\gamma_1 \ldots \gamma_n$ ersetzt werden.

Ersetzungsoperationen der BNF

(Fortsetzung)

- Hat β die Form (γ) (Klammerung), so darf α durch γ ersetzt werden.
- Hat β die Form $(\gamma)^*$ (Iteration), so darf α durch $\underbrace{(\gamma)(\gamma)\ldots(\gamma)}_{n-\text{mal}}$ mit einem beliebigen ($n \geq 0$) ersetzt werden.
- Hat β die Form $(\gamma)^+$ (nicht-leere Iteration), so darf α durch $\underbrace{(\gamma)(\gamma)\ldots(\gamma)}_{n-\text{mal}}$ mit einem beliebigen ($n > 0$) ersetzt werden.
- Hat β die Form $[\gamma]$ (Option), so darf α durch (γ) ersetzt oder ersatzlos gestrichen werden.

Beispiele

- BNF Syntaxdefinition für natürliche Zahlen:

  $\langle$*Dezimalzahl*$\rangle$ ::= 0 | $\langle$*Nichtnulldarstellung*$\rangle$

  $\langle$*Nichtnulldarstellung*$\rangle$ ::= $\langle$*Nichtnullziffer*$\rangle$ $(0 \mid \langle$*Nichtnullziffer*$\rangle)^*$

  $\langle$*Nichtnullziffer*$\rangle$ ::= $1|2|3|4|5|6|7|8|9$

  Die syntaktische Variable $\langle$*Dezimalzahl*$\rangle$ dient als Startsymbol α_S.

- Ableitung der Zeichenreihe 308 aus $\alpha_S = \langle$*Dezimalzahl*$\rangle$ ist:

  $\langle$*Dezimalzahl*$\rangle \rightarrow \langle$*Nichtnulldarstellung*$\rangle$
  $\rightarrow \langle$*Nichtnullziffer*$\rangle$ $(0 \mid \langle$*Nichtnullziffer*$\rangle)^*$
  $\rightarrow \langle$*Nichtnullziffer*$\rangle$ $(0 \mid \langle$*Nichtnullziffer*$\rangle)(0 \mid \langle$*Nichtnullziffer*$\rangle)$
  $\rightarrow \langle$*Nichtnullziffer*$\rangle$ 0 $\langle$*Nichtnullziffer*$\rangle$
  $\rightarrow$ 308

Beispiele

- BNF Syntaxdefinition für ganze Zahlen ("$-$" und die Dezimalziffern des Alphabets $\mathcal{A}_{10}$ sind Terminalzeichen):

  $\langle \textit{GanzeZahl} \rangle$ ::= $[-]$ $\langle \textit{Dezimalzahl} \rangle$

  Die syntaktische Variable $\langle \textit{GanzeZahl} \rangle$ dient hier als Startsymbol α_S.

- BNF Syntaxdefinition für Gleitpunktzahlen ("$-$", ".", "E" und die Dezimalziffern des Alphabets $\mathcal{A}_{10}$ sind Terminalzeichen):

  $\langle \textit{Gleitpunktzahl} \rangle$ ::= $[-]$ $\langle \textit{Mantisse} \rangle (E \langle \textit{GanzeZahl} \rangle)$

  $\langle \textit{Mantisse} \rangle$ ::= $\langle \textit{Dezimalzahl} \rangle . \langle \textit{Dezimalstellen} \rangle$

  $\langle \textit{Dezimalstellen} \rangle$ ::= $0 | \langle \textit{Nichtnullstellen} \rangle$

  $\langle \textit{Nichtnullstellen} \rangle$ ::= $(0 | \langle \textit{Nichtnullziffer} \rangle)^* \langle \textit{Nichtnullziffer} \rangle$

  Die syntaktische Variable $\langle \textit{Gleitpunktzahl} \rangle$ dient hier als Startsymbol α_S.

Diskussion

- Mit Hilfe der BNF kann man nicht nur die Syntax (Darstellung) von Daten formal definieren.
- Viele Programmiersprachen benützen BNF Syntaxdefinitionen auch für die eindeutige Definition ihrer Syntax, d.h. der Sprachelemente, die erlaubt sind und der Sprache an sich.
- Manche Eigenschaften von Darstellungen oder Sprachen kann man in BNF allerdings nicht darstellen. Diese werden dann typischerweise zusätzlich “verbal” angegeben, d.h. sie werden als Zusatzanforderungen notiert. Diese zusätzlichen Angaben heißen *Kontextbedingungen*.