LMU LUDWIG-MAXIMILIANS-UNIVERSITY MUNICH
DEPARTMENT INSTITUTE FOR INFORMATICS
DATABASE SYSTEMS GROUP

Kapitel 3

Daten und Algorithmen

Skript zur Vorlesung
Einführung in die Programmierung
im Wintersemester 2012/13
Ludwig-Maximilians-Universität München

Übersicht

Übersicht

3.1 Datendarstellung durch Zeichenreihen

Wir betrachten zunächst die Daten (Objekte), die durch Algorithmen verarbeitet werden sollen.

Typische Daten sind Zahlen, z.B. die Zahl “drei”, die wie folgt dargestellt werden kann:

- 3
- DREI
- III
- Drei ausgestreckte Finger einer Hand
- …

3.1 Datendarstellung durch Zeichenreihen

- 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*.

3.1 Datendarstellung durch Zeichenreihen

- Ein *Alphabet* 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: {↑, ↓}
  - Alphabete, die genau zwei Zeichen enthalten heißen *binär*.
  - Ein wichtiges binäres Alphabet besteht aus den *Binärziffern* (*Bits*): {0, 1}

3.1 Datendarstellung durch Zeichenreihen

- Eine *Zeichenreihe* über einem Alphabet A ist eine (endliche) Folge von Zeichen aus A. Wir schreiben Zeichenreihen $(x_1, x_2, \ldots, x_n)$ auch als $x_1x_2 \ldots x_n$.
- Beispiele:
  - Sei $A_1 = \{A, B, C, \ldots, Z\}$
    - Die Folge INFORMATIK ist eine Zeichenreihe über A_1.
    - Die Folge KRÖGER ist *keine* Zeichenreihe über A_1. **Warum?**
  - Sei $A_2 = \{0, 1\}$
    - Die Folge 0 ist eine Zeichenreihe über A_2.
    - Die Folge 1 ist eine Zeichenreihe über A_2.
    - Die Folge 01 ist eine Zeichenreihe über A_2.
    - Die Folge 10 ist eine Zeichenreihe über A_2.
    - Die Folge 11 ist eine Zeichenreihe über A_2.
    - Die Folge 00 ist eine Zeichenreihe über A_2.
    - …

3.1 Datendarstellung durch Zeichenreihen

- 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 ($A = \{0, 1, 2, \ldots, 9\}$),
  - DREIZEHN ($A = \{A, B, \ldots, Z\}$),
  - IIIIIIIIIIIII ($A = \{I\}$).

3.1 Datendarstellung durch Zeichenreihen

- Nicht alle diese Darstellungen sind für den praktischen Gebrauch (z.B. Rechnen) geeignet.
- Am besten geeignet ist die *Zifferndarstellung*, z.B. die allgemein gebräuchliche *Dezimaldarstellung* über dem Alphabet {0, 1, 2, . . . , 9}.

3.1 Datendarstellung durch Zeichenreihen

- Das allgemeine Prinzip der Zifferndarstellung ist wie folgt definiert:
  Sei $p \in \mathbb{N}$, $p \geq 2$ und $A_p = \{z_0, z_1, \ldots, z_{p-1}\}$ ein Alphabet mit p Zeichen (*Ziffern*) $z_0, z_1 \ldots, z_{p-1}$.
  Die Funktion $Z : A_p \rightarrow \mathbb{N}_0$ bildet jedes Zeichen aus A_p auf eine natürliche Zahl wie folgt ab:

$$Z(z_i) = i \qquad \text{für } i = 0, \ldots, p-1.$$

  Eine Zeichenreihe $x = x_n x_{n-1} \ldots x_1 x_0$ über A_p (d.h. $x_i \in A_p$ für $0 \leq i \leq n$) bezeichnet die Zahl

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

- Zur Verdeutlichung schreiben wir auch x_p.
  x_p heißt *p-adische Zahlendarstellung* der Zahl $\boldsymbol{Z}(x_p) \in \mathbb{N}_0$

3.1 Datendarstellung durch Zeichenreihen

- Nochmal die Formel: $\boldsymbol{Z}(x) = p^n \cdot Z(x_n) + p^{n-1} \cdot Z(x_n{-}1) + \ldots + p \cdot Z(x_1) + Z(x_0)$
- Beispiele:
  - Mit $p = 10$ und $A_{10} = \{z_0, z_1 \ldots, z_9\}$ erhält man die Dezimaldarstellung wenn man statt z_i gleich $Z(z_i)$ schreibt (also z.B. statt z_3 schreibe $Z(z_3) = 3$):

    $\boldsymbol{Z}(983_{10}) = 10^2 \cdot 9 + 10^1 \cdot 8 + 10^0 \cdot 3 =$ "**neunhundertdreiundachtzig**".

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

3.1 Datendarstellung durch Zeichenreihen

- Nochmal die Formel: $\boldsymbol{Z}(x) = p^n \cdot Z(x_n) + p^{n-1} \cdot Z(x_n-1) + \ldots + p \cdot Z(x_1) + Z(x_0)$
- Weitere Beispiele:
  - $p = 2$ und $A_2 = \{0, 1\}$ (*Binärdarstellung*):
    $\boldsymbol{Z}(1111010111_2) =$
    $2^9 \cdot 1 + 2^8 \cdot 1 + 2^7 \cdot 1 + 2^6 \cdot 1 + 2^5 \cdot 0 + 2^4 \cdot 1 + 2^3 \cdot 0 + 2^2 \cdot 1 + 2 \cdot 1 + 1 =$
    "**neunhundertdreiundachtzig**".
  - $p = 8$ und $A_8 = \{0, 1, 2, 3, 4, 5, 6, 7\}$ (*Oktaldarstellung*):
    $\boldsymbol{Z}(1727_8) = 8^3 \cdot 1 + 8^2 \cdot 7 + 8 \cdot 2 + 7 =$ "**neunhundertdreiundachtzig**".
  - $p = 16$ und $A_{16} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}$ (*Hexadezimaldarstellung*):
    $\boldsymbol{Z}(3D7_{16}) = 16^2 \cdot 3 + 16 \cdot 13 + 7 =$ "**neunhundertdreiundachtzig**".

3.1 Datendarstellung durch Zeichenreihen

- 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 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.

3.1 Datendarstellung durch Zeichenreihen

- 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.

3.1 Datendarstellung durch Zeichenreihen

- 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 "-"
Reelle Zahlen	$\mathbb{R}$	*Gleitpunktdarstellung* (s. später)
Wahrheitswerte	$\mathbb{B}$	*TRUE* und *FALSE* für "wahr" bzw. "falsch"

3.1 Datendarstellung durch Zeichenreihen

- 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)

3.1 Datendarstellung durch Zeichenreihen

- Auf Rechenanlagen wird meist eine andere Darstellung der Daten gewählt (typischerweise Zeichenreihen über den oben benannten Alphabeten A_2, A_8 und A_{16}).
- Aus technischen Gründen kann die kleinste Speichereinheit eines Computers (ein sog. *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 $A_2 = \{0, 1\}$) codiert gespeichert.

– Intern kann der Computer also z.B. die natürlichen Zahlen in Binärcodierung repräsentieren.

– Ganze Zahlen können intern ebenfalls leicht als Zeichenkette über dem Alphabet $A_2 = \{0, 1\}$ codiert werden (Genaueres darüber werden Sie in der Vorlesung "Rechnerarchitektur" lernen).

3.1 Datendarstellung durch Zeichenreihen

- Binärdarstellung reeller Zahlen ist i.d.R. etwas komplizierter.
- Üblicherweise verwenden sowohl Rechner als auch höhere Programmiersprachen die sog. *Gleitpunktdarstellung*.
- Eine Zahl z wird hier z.B. folgendermaßen dargestellt:

  $z = m \cdot 2^e$,

  wobei sowohl m (*Mantisse*) als auch e (*Exponent*) wiederum binär repräsentiert werden (können).

3.1 Datendarstellung durch Zeichenreihen

- In den meisten höheren 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 folgt im nächsten Abschnitt.
- **Wichtig**: Für viele reelle Zahlen gibt es ***keine*** solche Darstellung (z.B. $\sqrt{2}$). Die darstellbaren Zahlen heißen auch *Gleitpunktzahlen*.
- Die maschinengerechte Darstellung von Daten ist bei der Entwicklung von Algorithmen möglicherweise wichtig! **Warum?**

Übersicht

3.2 Syntaxdefinitionen

- Nocheinmal:
  - 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.

3.2 Syntaxdefinitionen

- Die Menge der natürlichen Zahlen in Dezimaldarstellung kann z.B. durch folgende Regeln induktiv definiert werden:
  1. 0 ist eine *<Dezimalzahl>*.
  2. Jede Ziffer $x \in A_{10} \backslash \{0\}$ ist eine *<Nichtnulldarstellung>*.
  3. Ist *a* eine *<Nichtnulldarstellung>* und $y \in A_{10}$ so ist $a \circ y$ eine *<Nichtnulldarstellung>*.
  4. Jede *<Nichtnulldarstellung>* ist eine *<Dezimalzahl>*.

3.2 Syntaxdefinitionen

- Beispiel: Die Zeichenreihe 308 ist eine Dezimalzahl gemäß folgender Anwendungen der Regeln:

  3 ist *<Nichtnulldarstellung>* nach Regel 2
  30 ist *<Nichtnulldarstellung>* nach Regel 3
  308 ist *<Nichtnulldarstellung>* nach Regel 3
  308 ist *<Dezimalzahl>* nach Regel 4

3.2 Syntaxdefinitionen

- Die Begriffe *<Dezimalzahl>* und *<Nichtnulldarstellung>* in den Regeln werden *syntaktische Variablen* genannt. Sie kennzeichnen den zu definierenden Begriff (*<Dezimalzahl>*) sowie einen Hilfsbegriff (*<Nichtnulldarstellung>*).
- 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*.

3.2 Syntaxdefinitionen

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

$$\alpha ::= \beta$$

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

3.2 Syntaxdefinitionen

- Eine BNF-Satzform ist wie folgt induktiv definiert
  - Syntaktaktische Variablen und Terminalzeichen sind BNF-Satzformen.
  - Auswahl:
    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 $[\gamma]$.

3.2 Syntaxdefinitionen

- Eine Zeichenreihe *w* ist *herleitbar* mit einer BNF Syntaxdefinition, wenn man *w* aus α_S durch eine oder mehrere der folgenden Ersetzungsoperationen erhalten kann:
  - Eine syntaktische Variable α, für die die Regel $\alpha ::= \beta$ vorhanden ist, darf durch β ersetzt werden
  - Auswahl: Ein Vorkommen von $\gamma_1 | \ldots | \gamma_n$ oder $(\gamma_1 | \ldots | \gamma_n)$ darf man durch eines der γ_i $(1 \leq i \leq n)$ ersetzen.
  - Klammerung: Ein Vorkommen von (γ) darf man durch γ ersetzen
  - Iteration: Ein Vorkommen von $(\gamma)^*$ darf man durch $\underbrace{(\gamma)(\gamma)\ldots(\gamma)}_{n-\text{mal}}$ mit einem beliebigen $(n \geq 0)$ ersetzen.
  - Nichtleere Iteration: Ein Vorkommen von $(\gamma)^+$ darf man durch $\underbrace{(\gamma)(\gamma)\ldots(\gamma)}_{n-\text{mal}}$ mit einem beliebigen $(n > 0)$ ersetzen.
  - Option: Ein Vorkommen von $[\gamma]$ darf man durch (γ) ersetzen, oder ersatzlos streichen.
- Wir schreiben auch $\alpha_S \rightarrow w$.

3.2 Syntaxdefinitionen

- BNF Syntaxdefinition für natürliche Zahlen:

  <*Dezimalzahl*> ::= 0 | <*Nichtnulldarstellung*>

  <*Nichtnulldarstellung*> ::= <*Nichtnullziffer*> (0 | <*Nichtnullziffer*>)*

  <*Nichtnullziffer*>::= 1|2|3|4|5|6|7|8|9

  Die syntaktische Variable <*Dezimalzahl*> dient als Startsymbol α_S.

- Ableitung der Zeichenreihe 308 aus α_S = <*Dezimalzahl*> ist:

  <*Dezimalzahl* >

  → <*Nichtnulldarstellung*>

  → <*Nichtnullziffer*> (0 | <*Nichtnullziffer*>)*

  → <*Nichtnullziffer*> (0 | <*Nichtnullziffer*>)(0 | <*Nichtnullziffer*>)

  → <*Nichtnullziffer*> 0 <*Nichtnullziffer*>

  → 308

3.2 Syntaxdefinitionen

- BNF Syntaxdefinition für ganze Zahlen ("–" und die Dezimalziffern des Alphabets A_{10} sind Terminalzeichen):

  <*GanzeZahl*> ::= [–] <*Dezimalzahl*>

  Die syntaktische Variable <*GanzeZahl*> dient hier als Startsymbol α_S.
- BNF Syntaxdefinition für Gleitpunktzahlen ("–", "E" und die Dezimalziffern des Alphabets A_{10} sind Terminalzeichen) :

  <Gleitpunktzahl> ::= [–] <Mantisse>(E<GanzeZahl>)

  <Mantisse> ::= <Dezimalzahl>.<Dezimalstellen>

  <Dezimalstellen>::= 0|<Nichtnullstellen>

  <Nichtnullstellen> ::= (0|<Nichtnullziffer>)*<Nichtnullziffer>

  Die syntaktische Variable <*Gleitpunktzahl*> dient hier als Startsymbol α_S.

3.2 Syntaxdefinitionen

- 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 Spracheelemente, 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*.