

3.1 Datendarstellung durch Zeichenreihen

3.2 Syntaxdefinitionen

3.3 Algorithmen

3.3 Algorithmen

- Nachdem wir bisher in Kapitel 3 die Eigenschaft “eindeutige Datendarstellung” diskutiert haben, widmen wir uns im folgenden den restlichen drei Eigenschaften von Algorithmen.
- Neben diesen Forderungen gibt es noch weitere interessante Eigenschaften, die Algorithmen haben können.
- Wir werden all diese Eigenschaften an einem Beispiel erläutern.

Zur Erinnerung: neben der eindeutigen Darstellung der Daten sind folgende Eigenschaften von Algorithmen ebenfalls wichtig

1. Präzise, endliche Beschreibung.
2. Effektiver, elementare Verarbeitungsschritte.
3. Ein Algorithmus heißt *terminierend*, wenn er bei jeder Anwendung nach endlich vielen Verarbeitungsschritten zum Ende kommt.
4. Ein Algorithmus heißt *deterministisch*, wenn die Wirkung und die Reihenfolge der Einzelschritte eindeutig festgelegt ist, andernfalls *nicht-deterministisch*.
5. Ein Algorithmus heißt *determiniert*, wenn das Ergebnis der Verarbeitung für jede einzelne Anwendung eindeutig bestimmt ist, andernfalls *nicht-determiniert*.

Desweiteren ist natürlich die möglicherweise wichtigste Eigenschaft eines Algorithmus die *Korrektheit*, d.h. informell, “der Algorithmus tut, was er tun soll”.

1. Ein Algorithmus heißt *partiell korrekt*, wenn für alle gültigen Eingaben das Resultat der Spezifikation des Algorithmus entspricht.
2. Ein Algorithmus heißt *(total) korrekt*, wenn der Algorithmus partiell korrekt ist, und für alle gültigen Eingaben terminiert.

Wir werden uns später noch genauer mit der Frage beschäftigen, wie man sicher sein kann, dass ein Algorithmus partiell bzw. total korrekt ist.

Runnig Example:

Wir betrachten folgende Aufgabe:

Ein Kunde kauft Waren für $1 \leq r \leq 100$ EUR und bezahlt mit einem 100 EUR Schein (r sei ein voller EUR Betrag ohne Cent-Anteil). Gesucht ist ein Algorithmus, der zum Rechnungsbetrag r das Wechselgeld w bestimmt. Zur Vereinfachung nehmen wir an, dass w nur aus 1 EUR oder 2 EUR Münzen oder 5 EUR Scheinen bestehen soll. Es sollen möglichst wenige Münzen / Scheine ausgegeben werden (also ein 5 EUR Schein statt fünf 1 EUR Münzen).

3.3 Algorithmen

- Zunächst müssen wir zur Lösung dieser Aufgabe die Darstellung (Modellierung) der relevanten Daten festlegen.
- Für den Rechnungsbetrag r ist dies trivial, denn offensichtlich ist $r \in \mathbb{N}$. Wir nehmen an, dass r in Dezimaldarstellung gegeben ist.
- Das Wechselgeld w kann auf verschiedene Weise modelliert werden, z.B. als Folge oder Multimenge von Wechselgeldmünzen. Ein aus zwei 1-EUR-Münzen, einer 2-EUR-Münze und zwei 5-EUR-Scheinen bestehendes Wechselgeld könnte als Folge $(1, 1, 2, 5, 5)$ dargestellt sein.

3.3 Algorithmen

- Wir legen folgende Datendarstellung fest:
 - r : als natürliche Zahl in Dezimaldarstellung.
 - w : als Folge von Werten 1, 2 oder 5.
- Wir benutzen dabei die Bezeichnung $()$ für die leere Folge und die Funktion \circ zur Konkatination zweier Folgen wie in Kap. 2 eingeführt. Desweiteren sagen wir auch "nimm x zu w hinzu" für die Operation $w \circ x$.

3.3 Algorithmen

- Idee: Ausgehend von r sukzessive um 1, 2 und 5 hochzählen (unter Hinzunahme der entsprechenden Münzen / Scheine) bis man bei 100 angekommen ist. Dabei möglichst schnell eine durch 5 teilbare Zahl erreichen, um möglichst wenige Münzen / Scheine auszugeben.
- Algorithmus *Wechselgeld 1*
Führe folgende Schritte der Reihe nach aus:
 1. Setze $w = ()$.
 2. Falls die letzte Ziffer von r eine 2, 4, 7 oder 9 ist, dann erhöhe r um 1 und nimm 1 zu w hinzu.
 3. Falls die letzte Ziffer von r eine 1 oder 6 ist, dann erhöhe r um 2 und nimm 2 zu w hinzu.
 4. Falls die letzte Ziffer von r eine 3 oder 8 ist, dann erhöhe r um 2 und nimm 2 zu w hinzu.
 5. Solange $r < 100$: Erhöhe r um 5 und nimm 5 zu w hinzu.

3.3 Algorithmen

- Ablaufbeispiel: Sei $r = 81$.
 - $r = 81$ (Ausgangssituation)
 - Schritt 1 $r = 81$ $w = ()$
 - Schritt 2 (keine Änderung, da die letzte Ziffer keine 2, 4, 7, 9 ist)
 - Schritt 3 $r = 83$ $w = (2)$
 - Schritt 4 $r = 85$ $w = (2, 2)$
 - Schritt 5 $r = 90$ $w = (2, 2, 5)$
 - $r = 95$ $w = (2, 2, 5, 5)$
 - $r = 100$ $w = (2, 2, 5, 5, 5)$
- Eigenschaften:
 - Endliche Aufschreibung: OK.
 - Effektive und elementare Einzelschritte: ?
 - Der Algorithmus ist terminierend, deterministisch, determiniert und partiell korrekt (und damit auch total korrekt).

3.3 Algorithmen

- Idee: Fasse ähnliche Schritte zusammen.
- Algorithmus *Wechselgeld 2*
Führe folgende Schritte der Reihe nach aus:
 1. Setze $w = ()$.
 2. Solange $r < 100$: Führe jeweils (wahlweise) einen der folgenden Schritte aus:
 1. Falls die letzte Ziffer von r eine 2, 4, 7 oder 9 ist, dann erhöhe r um 1 und nimm 1 zu w hinzu.
 2. Falls die letzte Ziffer von r eine 1, 2, 3, 6, 7 oder 8 ist, dann erhöhe r um 2 und nimm 2 zu w hinzu.
 3. Falls die letzte Ziffer von r eine 0, 1, 2, 3, 4 oder 5 ist, dann erhöhe r um 5 und nimm 5 zu w hinzu.

3.3 Algorithmen

- Ablaufbeispiel:

	$r = 81$	(Ausgangssituation)
Schritt 1	$r = 81$	$w = ()$
Schritt 2b	$r = 83$	$w = (2)$
Schritt 2b	$r = 85$	$w = (2, 2)$
Schritt 2c	$r = 90$	$w = (2, 2, 5)$
Schritt 2c	$r = 95$	$w = (2, 2, 5, 5)$
Schritt 2c	$r = 100$	$w = (2, 2, 5, 5, 5)$

- Alternativer Ablauf:

	$r = 81$	(Ausgangssituation)
Schritt 1	$r = 81$	$w = ()$
Schritt 2b	$r = 83$	$w = (2)$
Schritt 2c	$r = 88$	$w = (2, 5)$
Schritt 2b	$r = 90$	$w = (2, 5, 2)$
Schritt 2c	$r = 95$	$w = (2, 5, 2, 5)$
Schritt 2c	$r = 100$	$w = (2, 5, 2, 5, 5)$

- Eigenschaften:
 - Algorithmus 2 arbeitet nach dem selben Grundprinzip wie Algorithmus 1, lässt aber gewisse "Freiheiten" bei Schritt 2.
=> **Algorithmus 2 ist nicht-deterministisch.**
 - Algorithmus 2 ist nicht determiniert (siehe alternativer Ablauf)
 - Bemerkung: Wenn wir statt Folgen Multimengen (Reihenfolge der Elemente spielt keine Rolle) bei der Darstellung von w verwendet hätten, wären die Ergebnisse $w = \{2, 2, 5, 5, 5\}$ und $w = \{2, 5, 2, 5, 5\}$ gleich, d.h. in diesem Fall wäre Algorithmus 2 determiniert (aber nach wie vor nicht-deterministisch).
- Die Wahl der Datendarstellung kann also Einfluss auf Eigenschaften von Algorithmen haben.
- Bemerkung: Würden wir r nicht in Dezimal- sondern z.B. in Binärdarstellung modellieren, wären beide Algorithmen in der angegebenen Form sinnlos.

3.3 Algorithmen

- Idee: Wir lösen uns von der Abhängigkeit von der Datendarstellung, indem wir r nicht mehr hochzählen, sondern die Differenz $100 - r$ berechnen und diese in möglichst wenige Teile der Größen 1, 2 und 5 aufteilen. Wir gehen dabei davon aus, dass die Differenz “-” für beliebige Zifferndarstellungen (dezimal, binär, oktal, etc.) definiert ist. Die Zahl 100 müsste in die selbe Darstellung gebracht werden, die auch für r benutzt wird.
- Algorithmus *Wechselgeld 3*
Führe folgende Schritte der Reihe nach aus:
 1. Berechne $d = 100 - r$ und setze $w = ()$.
 2. Solange $d \geq 5$: Vermindere d um 5 und nimm 5 zu w hinzu.
 3. Falls $d \geq 2$, dann vermindere d um 2 und nimm 2 zu w hinzu.
 4. Falls $d \geq 2$, dann vermindere d um 2 und nimm 2 zu w hinzu.
 5. Falls $d \geq 1$, dann vermindere d um 1 und nimm 1 zu w hinzu.

3.3 Algorithmen

- Ablaufbeispiel:

	$r = 81$	(Ausgangssituation)
Schritt 1	$d = 19$	$w = ()$
Schritt 2	$d = 14$	$w = (5)$
	$d = 9$	$w = (5, 5)$
	$d = 4$	$w = (5, 5, 5)$
Schritt 3	$d = 2$	$w = (5, 5, 5, 2)$
Schritt 4	$d = 0$	$w = (5, 5, 5, 2, 2)$
Schritt 5	(keine Änderung, da $d \geq 1$ nicht gilt)	

- Diskussion
 - Alle drei Varianten weisen eine gemeinsame “operative” Auffassung auf: Die Berechnung des Rückgelds wird durch eine Folge von “Handlungen” vollzogen.
 - Diese Handlungen sind charakterisiert durch Veränderungen der Größen r bzw. d und w .
 - Die Abfolge dieser “Aktionen” wird durch typische Konstruktionen gesteuert:
 - *Fallunterscheidung*: Falls . . . , dann . . . ermöglicht, die Ausführungen von Aktionen vom Erfülltsein gewisser Bedingungen abhängig sein zu lassen.
 - *Iteration*: Solange . . . : . . . steuert die Wiederholung von Aktionen in Abhängigkeit gewisser Bedingungen.

3.3 Algorithmen

- Neben der strikten operativen Auffassung gibt es eine Sichtweise von Algorithmen, die durch eine rigorose mathematische Abstraktion gewonnen wird.
- In unserem Beispiel ist die Zuordnung eines Wechselgelds w zu einem Rechnungsbetrag r mathematisch nichts anderes als eine Abbildung $h : r \rightarrow$ "herauszugebendes Wechselgeld"
- Die Aufgabe ist dann, eine Darstellung der Abbildung h zu finden, die "maschinell auswertbar" ist.
- Triviale Lösung: Auflistung aller möglichen Werte für r mit zugehörigem Wechselgeld $w = h(r)$. **Probleme?**
- Allgemein wird man eine kompakte Darstellung suchen, in der die zu bestimmende Abbildung in geeigneter Weise aus einfachen (elementar auswertbaren) Abbildungen zusammengesetzt ist.

3.3 Algorithmen

- Für eine solche Darstellung der Abbildung h benötigen wir zwei “Hilfsabbildungen”, die in der Informatik oft gebräuchlich sind:

$$\begin{array}{l} DIV : \mathbb{N}_0 \times \mathbb{N} \rightarrow \mathbb{N}_0 \\ MOD : \mathbb{N}_0 \times \mathbb{N} \rightarrow \mathbb{N}_0 \end{array}$$

- DIV berechnet das Ergebnis der ganzzahlige Division zweier natürlicher Zahlen, z.B. $DIV(7, 3) = 2$.
- MOD berechnet den Rest der ganzzahligen Division zweier natürlicher Zahlen, z.B. $MOD(7, 3) = 1$.
- Formal: Sind $k \in \mathbb{N}_0$ und $l \in \mathbb{N}$, so kann man k durch l mit ganzzahligem Ergebnis q teilen, wobei eventuell ein Rest r übrigbleibt, d.h. es gibt $q, r \in \mathbb{N}_0$ mit $r < l$ und $k = q \cdot l + r$.
Dann ist

$$DIV(k, l) = q \quad \text{und} \quad MOD(k, l) = r.$$

3.3 Algorithmen

- Idee: Ähnlich wie Algorithmus 3 teilen wir $100 - r$ durch 5. Der ganzzahlige Quotient $q_1 = DIV(100 - r, 5)$ ist die Anzahl der 5-EUR-Scheine im Wechselgeld $h(r)$. Der Rest $r_1 = MOD(100 - r, 5)$ ist der noch zu verarbeitende Wechselbetrag. Offensichtlich gilt $r_1 < 5$. r_1 muss nun auf 1 und 2 aufgeteilt werden, d.h. analog bilden wir $q_2 = DIV(r_1, 2)$ und $r_2 = MOD(r_1, 2)$. q_2 bestimmt die Anzahl der Elemente 2 und r_2 die Anzahl der Elemente 1 in $h(r)$.
- Algorithmus *Wechselgeld 4*

$$h(r) = (5_1, \dots, 5_{DIV(100-r, 5)}, 2_1, \dots, 2_{DIV(MOD(100-r, 5), 2)}, 1_1, \dots, 1_{MOD(MOD(100-r, 5), 2)})$$

Dabei sind alle Elemente in der Ergebnisfolge durchnummeriert, um die Anzahl der entsprechenden Elemente anzudeuten, d.h. das Element 5_i bezeichnet den i -ten 5-EUR-Schein im Ergebnis.

3.3 Algorithmen

- Bemerkung:
In dieser Schreibweise kann z.B. $(5_1, \dots, 5_0)$ auftreten, was bedeuten soll, dass die Folge kein Element 5 enthält.
- Beispielanwendung $r = 81$:

$$\begin{aligned}
 DIV(100 - r, 5) &= DIV(19, 5) &= 3 \\
 MOD(100 - r, 5) &= MOD(19, 5) &= 4 \\
 DIV(MOD(100 - r, 5), 2) &= DIV(4, 2) &= 2 \\
 MOD(MOD(100 - r, 5), 2) &= MOD(4, 2) &= 0
 \end{aligned}$$

d.h. man erhält $h(81) = (5_1, 5_2, 5_3, 2_1, 2_2, 1_1, 1_0)$ oder, einfacher geschrieben,

$$h(81) = (5, 5, 5, 2, 2).$$

- Eigenschaften:
 - Wenn man von der Bildung der Folgen aus den berechneten Anzahlen der Elemente 1, 2 und 5 absieht, sind nur die Auswertungen der Operationen “-”, *DIV* und *MOD* als Einzelschritte anzusehen. Setzt man diese als elementar voraus, ist die angegebene Definition von *h* eine Beschreibung des gesuchten Algorithmus.
 - Diese Darstellung nimmt keinen Bezug mehr auf eine “Abfolge von Aktionen” sondern beschreibt den “funktionalen Zusammenhang” zwischen *r* und *w* durch ineinander eingesetzte Anwendungen gewisser Basisoperationen.

3.3 Algorithmen

- Idee: Im Fall der Abbildung $h \rightarrow$ "Folgen über 1, 2, 5" kann man sehr leicht eine rekursive Definition finden, die die Operationen *DIV* und *MOD* nicht mehr verwendet.
- Algorithmus *Wechselgeld 5*

$$h(r) = \begin{cases} \text{falls } r = 100, \text{ dann } (), & (1) \\ \text{falls } 100 - r \geq 5, \text{ dann } (5) \circ h(r + 5), & (2) \\ \text{falls } 5 > 100 - r \geq 2, \text{ dann } (2) \circ h(r + 2), & (3) \\ \text{falls } 2 > 100 - r \geq 1, \text{ dann } (1) \circ h(r + 1), & (4) \end{cases}$$

- Hier ist der Basisfall (1) für $h(100)$ definiert und die Rekursionsfälle (2), (3), (4) werden auf die Fälle $h(r + 5)$, $h(r + 2)$ und $h(r + 1)$ zurückgeführt.

3.3 Algorithmen

- Beispielanwendung $r = 81$:

$$\begin{aligned}
 h(81) &= (5) \circ h(86) && \text{(Fall 2)} \\
 &= (5) \circ (5) \circ h(91) && \text{(Fall 2)} \\
 &= (5) \circ (5) \circ (5) \circ h(96) && \text{(Fall 2)} \\
 &= (5) \circ (5) \circ (5) \circ (2) \circ h(98) && \text{(Fall 3)} \\
 &= (5) \circ (5) \circ (5) \circ (2) \circ (2) \circ h(100) && \text{(Fall 3)} \\
 &= (5) \circ (5) \circ (5) \circ (2) \circ (2) \circ () && \text{(Fall 1)} \\
 &= (5, 5, 5, 2, 2)
 \end{aligned}$$

3.3 Algorithmen

- Trotz der rein funktionalen Darstellung der Zuordnung $r \rightarrow w$ erinnert die Auswertung wieder an die Abfolge der Aktionen im entsprechenden Ablaufbeispiel (ähnlich wie in Algorithmus 3).
- Tatsächlich kann das Prinzip der Rekursion auch in der operativen Auffassung der Algorithmen 1-3 eingerichtet werden und dabei einen ähnlichen Effekt wie die Iteration erzielen.
- In einer rekursiven Abbildungsdefinition greift man auf Werte dieser Abbildung für kleinere (oder hier: größere) Argumente zurück.
- Analog kann in einem Algorithmus, der die Abfolge gewisser Aktionen in Abhängigkeit von Eingabewerten steuert, die Ausführung des Algorithmus selbst auch als eine derartige Aktion auftreten.

3.3 Algorithmen

- Algorithmus *Wechselgeld 6*
Setze $w = ()$ und führe anschließend folgenden in Abhängigkeit der Eingabegröße r rekursiv definierten Algorithmus $A(r)$ aus:
 $A(r)$: Führe denjenigen der folgenden Schritte (1) - (3) aus, dessen Bedingung erfüllt ist (ist keine Bedingung erfüllt, ist die Ausführung zu Ende):
 1. Falls $100 - r \geq 5$, dann nimm 5 zu w hinzu und führe anschließend $A(r + 5)$ aus.
 2. Falls $5 > 100 - r \geq 2$, dann nimm 2 zu w hinzu und führe anschließend $A(r + 2)$ aus.
 3. Falls $5 > 100 - r \geq 1$, dann nimm 1 zu w hinzu und führe anschließend $A(r + 1)$ aus.

- Zusammenfassung:

Grundlegende algorithmische Konzepte sind (z.T. abhängig vom verwendeten Programmierparadigma):

- Eingabe- und Ergebnisdaten (insbesondere solche von komplexer Art) müssen geeignet dargestellt werden.
- Die Ausführung gewisser Aktionen bzw. die Auswertung gewisser Operationen wird als in elementaren Schritten durchführbar vorausgesetzt.
- Einzelne Schritte können zusammengesetzt werden (z.B. Nacheinanderausführung, Auswahl von Aktionen, Kompositionen von Abbildungen, etc.)
- Fallunterscheidung, Iteration und Rekursion ermöglichen die Steuerung des durch den Algorithmus beschriebenen Verfahrens.

Paradigmen:

- *Imperative Algorithmen:*
 - Abfolge von Aktionen, die typischerweise bestimmte Größen verändern.
 - Entspricht auch den technischen Möglichkeiten von Rechneranlagen, die Schritt für Schritt bestimmte grundlegende Aktionen ausführen können.
- *Funktionale Algorithmen:*
 - Auswertbare Darstellung des funktionalen Zusammenhangs zwischen Ein- und Ausgabewerten.
 - Höheres Abstraktionsniveau: Obwohl zur Auswertung der entsprechenden Abbildung wiederum bestimmte Schritte ausgeführt werden, ist die eigentliche Spezifikation des Algorithmus als Abbildung praktisch losgelöst von den technischen Möglichkeiten der Rechenanlage.

- *Objektorientierte Algorithmen:*
 - Höhere Abstraktionsebene zur Darstellung von komplexen Eingabe- und Ergebnisdaten sowie Zwischenzuständen während der Berechnung.
 - Eigentlichen Lösung der gegebenen Aufgabe können sowohl über einen funktionalen als auch über einen imperativen Algorithmus erfolgen.