1 Vom Problem zum Program

Ein Problem besteht darin, aus einer Menge von Informationen eine weitere (unbekannte) Information zu bestimmen.

mathematisch:
Ein Problem beschreibt eine Funktion $f : E \to A$, mit
$E =$ zulässige Eingaben und $A =$ mögliche Ausgaben.

Beispiele:

- Addition: $f : \mathbb{Q} \times \mathbb{Q} \to \mathbb{Q}$
- Primzahltest: $f : \mathbb{N} \to \{\text{yes}, \text{no}\}$
- Schach: $f : \mathcal{P} \to \mathcal{Z}$, wobei $\mathcal{P}$ die Menge aller Schachpositionen ist, und $f(P)$, der beste Zug in Position P.

1 Vom Problem zum Program

Ein Problem besteht darin, aus einer Menge von Informationen eine weitere (unbekannte) Information zu bestimmen.

mathematisch:
Ein Problem beschreibt eine Funktion $f : E \to A$, mit
$E =$ zulässige Eingaben und $A =$ mögliche Ausgaben.

Beispiele:

- Addition: $f : \mathbb{Q} \times \mathbb{Q} \to \mathbb{Q}$
- Primzahltest: $f : \mathbb{N} \to \{\text{yes}, \text{no}\}$
- Schach: $f : \mathcal{P} \to Z$, wobei $\mathcal{P}$ die Menge aller Schachpositionen ist, und $f(P)$, der beste Zug in Position P.

1 Vom Problem zum Program

Ein Problem besteht darin, aus einer Menge von Informationen eine weitere (unbekannte) Information zu bestimmen.

mathematisch:
Ein Problem beschreibt eine Funktion $f : E \to A$, mit
$E =$ zulässige Eingaben und $A =$ mögliche Ausgaben.

Beispiele:

- Addition: $f : \mathbb{Q} \times \mathbb{Q} \to \mathbb{Q}$
- Primzahltest: $f : \mathbb{N} \to \{\text{yes}, \text{no}\}$
- Schach: $f : \mathcal{P} \to \mathcal{Z}$, wobei $\mathcal{P}$ die Menge aller Schachpositionen ist, und $f(P)$, der beste Zug in Position P.

1 Vom Problem zum Program

Ein Problem besteht darin, aus einer Menge von Informationen eine weitere (unbekannte) Information zu bestimmen.

mathematisch:
Ein Problem beschreibt eine Funktion $f : E \to A$, mit
$E =$ zulässige Eingaben und $A =$ mögliche Ausgaben.

Beispiele:

- Addition: $f : \mathbb{Q} \times \mathbb{Q} \to \mathbb{Q}$
- Primzahltest: $f : \mathbb{N} \to \{\text{yes}, \text{no}\}$
- Schach: $f : \mathcal{P} \to \mathcal{Z}$, wobei $\mathcal{P}$ die Menge aller Schachpositionen ist, und $f(P)$, der beste Zug in Position P.

1 Vom Problem zum Program

Ein Problem besteht darin, aus einer Menge von Informationen eine weitere (unbekannte) Information zu bestimmen.

mathematisch:
Ein Problem beschreibt eine Funktion $f : E \to A$, mit
$E =$ zulässige Eingaben und $A =$ mögliche Ausgaben.

Beispiele:

- Addition: $f : \mathbb{Q} \times \mathbb{Q} \to \mathbb{Q}$
- Primzahltest: $f : \mathbb{N} \to \{\text{yes}, \text{no}\}$
- Schach: $f : \mathcal{P} \to \mathcal{Z}$, wobei $\mathcal{P}$ die Menge aller Schachpositionen ist, und $f(P)$, der beste Zug in Position P.

Algorithmus

Ein Algorithmus ist ein exaktes Verfahren zur Lösung eines Problems, d.h. zur Bestimmung der gewünschten Resultate.

Man sagt auch ein Algorithmus berechnet eine Funktion f.

Ausschnitt aus Briefmarke, Soviet Union 1983
Public Domain

Abu Abdallah Muhamed ibn Musa al-Chwarizmi, ca. 780–835

1 Vom Problem zum Program

Ein Problem besteht darin, aus einer Menge von Informationen eine weitere (unbekannte) Information zu bestimmen.

mathematisch:
Ein Problem beschreibt eine Funktion $f : E \to A$, mit
$E =$ zulässige Eingaben und $A =$ mögliche Ausgaben.

Beispiele:

- Addition: $f : \mathbb{Q} \times \mathbb{Q} \to \mathbb{Q}$
- Primzahltest: $f : \mathbb{N} \to \{\text{yes}, \text{no}\}$
- Schach: $f : \mathcal{P} \to \mathcal{Z}$, wobei $\mathcal{P}$ die Menge aller Schachpositionen ist, und $f(P)$, der beste Zug in Position P.

Algorithmus

Beobachtung:
Nicht jedes Problem läßt sich durch einen Algorithmus lösen (↑Berechenbarkeitstheorie).

Beweisidee: (↑Diskrete Strukturen)

- es gibt überabzählbar unendlich viele Probleme
- es gibt abzählbar unendlich viele Algorithmen

Algorithmus

Ein Algorithmus ist ein exaktes Verfahren zur Lösung eines Problems, d.h. zur Bestimmung der gewünschten Resultate.

Man sagt auch ein Algorithmus berechnet eine Funktion f.

Ausschnitt aus Briefmarke, Soviet Union 1983
Public Domain

Abu Abdallah Muhamed ibn Musa al-Chwarizmi, ca. 780–835

Algorithmus

Beobachtung:
Nicht jedes Problem läßt sich durch einen Algorithmus lösen (↑Berechenbarkeitstheorie).

Beweisidee: (↑Diskrete Strukturen)

- es gibt überabzählbar unendlich viele Probleme
- es gibt abzählbar unendlich viele Algorithmen

Algorithmus

Ein Algorithmus ist ein exaktes Verfahren zur Lösung eines Problems, d.h. zur Bestimmung der gewünschten Resultate.

Man sagt auch ein Algorithmus berechnet eine Funktion f.

Ausschnitt aus Briefmarke, Soviet Union 1983
Public Domain

Abu Abdallah Muhamed ibn Musa al-Chwarizmi, ca. 780–835

Algorithmus

Das exakte Verfahren besteht i.a. darin, eine Abfolge von elementaren Einzelschritten der Verarbeitung festzulegen.

Beispiel: Alltagsalgorithmen

Resultat	*Algorithmus*	*Einzelschritte*
Pullover	Strickmuster	eine links, eine rechts, eine fallen lassen
Kuchen	Rezept	nimm 3 Eier . . .
Konzert	Partitur	Noten

Algorithmus

Beobachtung:
Nicht jedes Problem läßt sich durch einen Algorithmus lösen (↑Berechenbarkeitstheorie).

Beweisidee: (↑Diskrete Strukturen)

- es gibt überabzählbar unendlich viele Probleme
- es gibt abzählbar unendlich viele Algorithmen

Beispiel: Euklidischer Algorithmus

Problem: geg. $a, b \in \mathbb{N}, a, b \neq 0$. Bestimme $\mathrm{ggT}(a, b)$.

Algorithmus:

1. Falls $a = b$ brich Berechnung ab. Es gilt $\mathrm{ggT}(a, b) = a$. Ansonsten gehe zu Schritt 2.
2. Falls $a > b$, ersetze a durch $a - b$ und setze Berechnung in Schritt 1 fort. Ansonsten gehe zu Schritt 3.
3. Es gilt $a < b$. Ersetze b durch $b - a$ und setze Berechnung in Schritt 1 fort.

Algorithmus

Das exakte Verfahren besteht i.a. darin, eine Abfolge von elementaren Einzelschritten der Verarbeitung festzulegen.

Beispiel: Alltagsalgorithmen

Resultat	*Algorithmus*	*Einzelschritte*
Pullover	Strickmuster	eine links, eine rechts, eine fallen lassen
Kuchen	Rezept	nimm 3 Eier ...
Konzert	Partitur	Noten

Beispiel: Euklidischer Algorithmus

Warum geht das?
Wir zeigen, für $a > b$: $\mathrm{ggT}(a,b) = \mathrm{ggT}(a-b,b)$.

Seien $g = \mathrm{ggT}(a,b)$, $g' = \mathrm{ggT}(a-b,b)$.
Dann gilt:

Beispiel: Euklidischer Algorithmus

Problem: geg. $a,b \in \mathbb{N}, a,b \neq 0$. Bestimme $\mathrm{ggT}(a,b)$.

Algorithmus:

1. Falls $a = b$ brich Berechnung ab. Es gilt $\mathrm{ggT}(a,b) = a$. Ansonsten gehe zu Schritt 2.
2. Falls $a > b$, ersetze a durch $a - b$ und setze Berechnung in Schritt 1 fort. Ansonsten gehe zu Schritt 3.
3. Es gilt $a < b$. Ersetze b durch $b - a$ und setze Berechnung in Schritt 1 fort.

Beispiel: Euklidischer Algorithmus

Warum geht das?

Wir zeigen, für $a > b$: $\mathrm{ggT}(a,b) = \mathrm{ggT}(a-b,b)$.

Seien $g = \mathrm{ggT}(a,b)$, $g' = \mathrm{ggT}(a-b,b)$.
Dann gilt:

$$\begin{aligned} a &= q_a \cdot g \\ b &= q_b \cdot g \end{aligned} \quad \text{und} \quad \begin{aligned} a-b &= q'_{a-b} \cdot g' \\ b &= q'_b \cdot g' \end{aligned}$$

Beispiel: Euklidischer Algorithmus

Problem: geg. $a, b \in \mathbb{N}, a, b \neq 0$. Bestimme $\mathrm{ggT}(a,b)$.

Algorithmus:

1. Falls $a = b$ brich Berechnung ab. Es gilt $\mathrm{ggT}(a,b) = a$. Ansonsten gehe zu Schritt 2.
2. Falls $a > b$, ersetze a durch $a - b$ und setze Berechnung in Schritt 1 fort. Ansonsten gehe zu Schritt 3.
3. Es gilt $a < b$. Ersetze b durch $b - a$ und setze Berechnung in Schritt 1 fort.

Beispiel: Euklidischer Algorithmus

Warum geht das?
Wir zeigen, für $a > b$: $\text{ggT}(a, b) = \text{ggT}(a - b, b)$.

Seien $g = \text{ggT}(a, b)$, $g' = \text{ggT}(a - b, b)$.
Dann gilt:

$$\begin{aligned} a &= q_a \cdot g \\ b &= q_b \cdot g \end{aligned} \quad \text{und} \quad \begin{aligned} a - b &= q'_{a-b} \cdot g' \\ b &= q'_b \cdot g' \end{aligned}$$

$$\begin{aligned} a - b &= (q_a - q_b) \cdot g \\ b &= q_b \cdot g \end{aligned} \quad \text{und} \quad \begin{aligned} a &= (q'_{a-b} + q'_b) \cdot g' \\ b &= q'_b \cdot g' \end{aligned}$$

Beispiel: Euklidischer Algorithmus

Problem: geg. $a, b \in \mathbb{N}, a, b \neq 0$. Bestimme $\text{ggT}(a, b)$.

Algorithmus:

1. Falls $a = b$ brich Berechnung ab. Es gilt $\text{ggT}(a, b) = a$. Ansonsten gehe zu Schritt 2.
2. Falls $a > b$, ersetze a durch $a - b$ und setze Berechnung in Schritt 1 fort. Ansonsten gehe zu Schritt 3.
3. Es gilt $a < b$. Ersetze b durch $b - a$ und setze Berechnung in Schritt 1 fort.

Beispiel: Euklidischer Algorithmus

Warum geht das?
Wir zeigen, für $a > b$: $\mathrm{ggT}(a, b) = \mathrm{ggT}(a - b, b)$.

Seien $g = \mathrm{ggT}(a, b)$, $g' = \mathrm{ggT}(a - b, b)$.
Dann gilt:

$$\begin{aligned} a &= q_a \cdot g \\ b &= q_b \cdot g \end{aligned} \quad \text{und} \quad \begin{aligned} a - b &= q'_{a-b} \cdot g' \\ b &= q'_b \cdot g' \end{aligned}$$

$$\begin{aligned} a - b &= (q_a - q_b) \cdot g \\ b &= q_b \cdot g \end{aligned} \quad \text{und} \quad \begin{aligned} a &= (q'_{a-b} + q'_b) \cdot g' \\ b &= q'_b \cdot g' \end{aligned}$$

Das heißt g ist Teiler von $a - b, b$ und g' ist Teiler von a, b.

Daraus folgt $g \le g'$ und $g' \le g$, also $g = g'$.

Beispiel: Euklidischer Algorithmus

Problem: geg. $a, b \in \mathbb{N}, a, b \neq 0$. Bestimme $\mathrm{ggT}(a, b)$.

Algorithmus:

1. Falls $a = b$ brich Berechnung ab. Es gilt $\mathrm{ggT}(a, b) = a$. Ansonsten gehe zu Schritt 2.
2. Falls $a > b$, ersetze a durch $a - b$ und setze Berechnung in Schritt 1 fort. Ansonsten gehe zu Schritt 3.
3. Es gilt $a < b$. Ersetze b durch $b - a$ und setze Berechnung in Schritt 1 fort.

Eigenschaften

(statische) Finitheit. Die Beschreibung des Algorithmus besitzt endliche Länge. (↑nichtuniforme Algorithmen)

(dynamische) Finitheit. Die bei Abarbeitung entstehenden Zwischenergebnisse sind endlich.

Terminiertheit. Algorithmen, die nach endlich vielen Schritten ein Resultat liefern, heißen terminierend. (↑Betriebssysteme, ↑reaktive Systeme)

Determiniertheit. Bei gleichen Eingabedaten gibt ein Algorithmus das gleiche Ergebnis aus. (↑randomisierte Algorithmen, ↑nicht-deterministische Algorithmen)

Determinismus. Der nächste anzuwendende Schritt im Verfahren ist stets eindeutig definiert. (↑randomisierte Algorithmen, ↑nicht-deterministische Algorithmen)

Beispiel: Euklidischer Algorithmus

Warum geht das?
Wir zeigen, für $a > b$: $\mathrm{ggT}(a,b) = \mathrm{ggT}(a-b,b)$.

Seien $g = \mathrm{ggT}(a,b)$, $g' = \mathrm{ggT}(a-b,b)$.
Dann gilt:

$$\begin{aligned} a &= q_a \cdot g \\ b &= q_b \cdot g \end{aligned} \quad \text{und} \quad \begin{aligned} a-b &= q'_{a-b} \cdot g' \\ b &= q'_b \cdot g' \end{aligned}$$

$$\begin{aligned} a-b &= (q_a - q_b) \cdot g \\ b &= q_b \cdot g \end{aligned} \quad \text{und} \quad \begin{aligned} a &= (q'_{a-b} + q'_b) \cdot g' \\ b &= q'_b \cdot g' \end{aligned}$$

Das heißt g ist Teiler von $a-b, b$ und g' ist Teiler von a, b.

Daraus folgt $g \le g'$ und $g' \le g$, also $g = g'$.

Eigenschaften

(statische) Finitheit. Die Beschreibung des Algorithmus besitzt endliche Länge. (↑nichtuniforme Algorithmen)

(dynamische) Finitheit. Die bei Abarbeitung entstehenden Zwischenergebnisse sind endlich.

Terminiertheit. Algorithmen, die nach endlich vielen Schritten ein Resultat liefern, heißen terminierend. (↑Betriebssysteme, ↑reaktive Systeme)

Determiniertheit. Bei gleichen Eingabedaten gibt ein Algorithmus das gleiche Ergebnis aus. (↑randomisierte Algorithmen, ↑nicht-deterministische Algorithmen)

Determinismus. Der nächste anzuwendende Schritt im Verfahren ist stets eindeutig definiert. (↑randomisierte Algorithmen, ↑nicht-deterministische Algorithmen)

Beispiel: Euklidischer Algorithmus

Warum geht das?
Wir zeigen, für $a > b$: $\mathrm{ggT}(a,b) = \mathrm{ggT}(a-b,b)$.

Seien $g = \mathrm{ggT}(a,b)$, $g' = \mathrm{ggT}(a-b,b)$.
Dann gilt:

$$\begin{aligned} a &= q_a \cdot g \\ b &= q_b \cdot g \end{aligned} \quad \text{und} \quad \begin{aligned} a-b &= q'_{a-b} \cdot g' \\ b &= q'_b \cdot g' \end{aligned}$$

$$\begin{aligned} a-b &= (q_a - q_b) \cdot g \\ b &= q_b \cdot g \end{aligned} \quad \text{und} \quad \begin{aligned} a &= (q'_{a-b} + q'_b) \cdot g' \\ b &= q'_b \cdot g' \end{aligned}$$

Das heißt g ist Teiler von $a-b, b$ und g' ist Teiler von a, b.

Daraus folgt $g \le g'$ und $g' \le g$, also $g = g'$.

Eigenschaften

(statische) Finitheit. Die Beschreibung des Algorithmus besitzt endliche Länge. (↑nichtuniforme Algorithmen)

(dynamische) Finitheit. Die bei Abarbeitung entstehenden Zwischenergebnisse sind endlich.

Terminiertheit. Algorithmen, die nach endlich vielen Schritten ein Resultat liefern, heißen terminierend. (↑Betriebssysteme, ↑reaktive Systeme)

Determiniertheit. Bei gleichen Eingabedaten gibt ein Algorithmus das gleiche Ergebnis aus. (↑randomisierte Algorithmen, ↑nicht-deterministische Algorithmen)

Determinismus. Der nächste anzuwendende Schritt im Verfahren ist stets eindeutig definiert. (↑randomisierte Algorithmen, ↑nicht-deterministische Algorithmen)

Beispiel: Euklidischer Algorithmus

Warum geht das?
Wir zeigen, für $a > b$: $\mathrm{ggT}(a,b) = \mathrm{ggT}(a-b,b)$.

Seien $g = \mathrm{ggT}(a,b)$, $g' = \mathrm{ggT}(a-b,b)$.
Dann gilt:

$$\begin{array}{rcl} a & = & q_a \cdot g \\ b & = & q_b \cdot g \end{array} \quad \text{und} \quad \begin{array}{rcl} a-b & = & q'_{a-b} \cdot g' \\ b & = & q'_b \cdot g' \end{array}$$

$$\begin{array}{rcl} a-b & = & (q_a - q_b) \cdot g \\ b & = & q_b \cdot g \end{array} \quad \text{und} \quad \begin{array}{rcl} a & = & (q'_{a-b} + q'_b) \cdot g' \\ b & = & q'_b \cdot g' \end{array}$$

Das heißt g ist Teiler von $a-b, b$ und g' ist Teiler von a, b.

Daraus folgt $g \le g'$ und $g' \le g$, also $g = g'$.

Eigenschaften

(statische) Finitheit. Die Beschreibung des Algorithmus besitzt endliche Länge. (↑nichtuniforme Algorithmen)

(dynamische) Finitheit. Die bei Abarbeitung entstehenden Zwischenergebnisse sind endlich.

Terminiertheit. Algorithmen, die nach endlich vielen Schritten ein Resultat liefern, heißen terminierend. (↑Betriebssysteme, ↑reaktive Systeme)

Determiniertheit. Bei gleichen Eingabedaten gibt ein Algorithmus das gleiche Ergebnis aus. (↑randomisierte Algorithmen, ↑nicht-deterministische Algorithmen)

Determinismus. Der nächste anzuwendende Schritt im Verfahren ist stets eindeutig definiert. (↑randomisierte Algorithmen, ↑nicht-deterministische Algorithmen)

Beispiel: Euklidischer Algorithmus

Warum geht das?
Wir zeigen, für $a > b$: $\mathrm{ggT}(a,b) = \mathrm{ggT}(a-b,b)$.

Seien $g = \mathrm{ggT}(a,b)$, $g' = \mathrm{ggT}(a-b,b)$.
Dann gilt:

$$\begin{array}{rcl} a & = & q_a \cdot g \\ b & = & q_b \cdot g \end{array} \quad \text{und} \quad \begin{array}{rcl} a-b & = & q'_{a-b} \cdot g' \\ b & = & q'_b \cdot g' \end{array}$$

$$\begin{array}{rcr} a-b & = & (q_a - q_b) \cdot g \\ b & = & q_b \cdot g \end{array} \quad \text{und} \quad \begin{array}{rcr} a & = & (q'_{a-b} + q'_b) \cdot g' \\ b & = & q'_b \cdot g' \end{array}$$

Das heißt g ist Teiler von $a-b, b$ und g' ist Teiler von a, b.

Daraus folgt $g \le g'$ und $g' \le g$, also $g = g'$.

Eigenschaften

(statische) Finitheit. Die Beschreibung des Algorithmus besitzt endliche Länge. (↑nichtuniforme Algorithmen)

(dynamische) Finitheit. Die bei Abarbeitung entstehenden Zwischenergebnisse sind endlich.

Terminiertheit. Algorithmen, die nach endlich vielen Schritten ein Resultat liefern, heißen terminierend. (↑Betriebssysteme, ↑reaktive Systeme)

Determiniertheit. Bei gleichen Eingabedaten gibt ein Algorithmus das gleiche Ergebnis aus. (↑randomisierte Algorithmen, ↑nicht-deterministische Algorithmen)

Determinismus. Der nächste anzuwendende Schritt im Verfahren ist stets eindeutig definiert. (↑randomisierte Algorithmen, ↑nicht-deterministische Algorithmen)

Beispiel: Euklidischer Algorithmus

Warum geht das?
Wir zeigen, für $a > b$: $\mathrm{ggT}(a,b) = \mathrm{ggT}(a-b,b)$.

Seien $g = \mathrm{ggT}(a,b)$, $g' = \mathrm{ggT}(a-b,b)$.
Dann gilt:

$$\begin{aligned} a &= q_a \cdot g \\ b &= q_b \cdot g \end{aligned} \quad \text{und} \quad \begin{aligned} a-b &= q'_{a-b} \cdot g' \\ b &= q'_b \cdot g' \end{aligned}$$

$$\begin{aligned} a-b &= (q_a - q_b) \cdot g \\ b &= q_b \cdot g \end{aligned} \quad \text{und} \quad \begin{aligned} a &= (q'_{a-b} + q'_b) \cdot g' \\ b &= q'_b \cdot g' \end{aligned}$$

Das heißt g ist Teiler von $a-b, b$ und g' ist Teiler von a, b.

Daraus folgt $g \le g'$ und $g' \le g$, also $g = g'$.

Programm

Ein Programm ist die Formulierung eines Algorithmus in einer Programmiersprache.

Die Formulierung gestattet (hoffentlich) eine maschinelle Ausführung.

- Ein Programmsystem berechnet i.a. nicht nur eine Funktion, sondern immer wieder Funktionen in Interaktion mit Benutzerinnen und/oder der Umgebung.
- Es gibt viele Programmmiersprachen: Java, C, Prolog, Fortran, TeX, PostScript, . . .

Eigenschaften

(statische) Finitheit. Die Beschreibung des Algorithmus besitzt endliche Länge. (↑nichtuniforme Algorithmen)

(dynamische) Finitheit. Die bei Abarbeitung entstehenden Zwischenergebnisse sind endlich.

Terminiertheit. Algorithmen, die nach endlich vielen Schritten ein Resultat liefern, heißen terminierend. (↑Betriebssysteme, ↑reaktive Systeme)

Determiniertheit. Bei gleichen Eingabedaten gibt ein Algorithmus das gleiche Ergebnis aus. (↑randomisierte Algorithmen, ↑nicht-deterministische Algorithmen)

Determinismus. Der nächste anzuwendende Schritt im Verfahren ist stets eindeutig definiert. (↑randomisierte Algorithmen, ↑nicht-deterministische Algorithmen)

Programm

Ein Programm ist gut, wenn

- die Programmiererin in ihr algorithmische Ideen natürlich beschreiben kann, insbesondere später noch versteht was das Programm tut (oder nicht tut);
- ein Computer das Programm leicht verstehen und effizient ausführen kann.

Programm

Ein Programm ist die Formulierung eines Algorithmus in einer Programmiersprache.

Die Formulierung gestattet (hoffentlich) eine maschinelle Ausführung.

- Ein Programmsystem berechnet i.a. nicht nur eine Funktion, sondern immer wieder Funktionen in Interaktion mit Benutzerinnen und/oder der Umgebung.
- Es gibt viele Programmmiersprachen: Java, C, Prolog, Fortran, TeX, PostScript, . . .