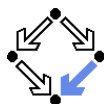


## Formale Grundlagen der Informatik 2 Algorithmen

Wolfgang Schreiner  
Wolfgang.Schreiner@risc.uni-linz.ac.at

Research Institute for Symbolic Computation (RISC)  
Johannes Kepler University, Linz, Austria  
<http://www.risc.uni-linz.ac.at>



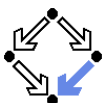
## Was ist ein Algorithmus?

al-Khwarizmi (um 825): *Über das Rechnen mit indischen Ziffern.*

- **Algorithmus:** eine genau definierte Handlungsvorschrift zur Lösung eines bestimmten Problems.
  - Nimmt eine Eingabe.
  - Führt eine endliche Anzahl von Schritten aus.
  - Liefert eine Ausgabe.
- Charakteristische Eigenschaften eines Algorithmus:
  - Ist durch eine endliche Menge von Anweisungen beschrieben.
  - Jede Anweisung ist durch einen "Rechner" effektiv ausführbar.
  - Die nächste auszuführende Anweisung ist klar definiert.
  - Der Algorithmus liefert für jede Eingabe ein Ergebnis.
  - Gleiche Eingaben führen zu gleichem Ergebnis.
  - ...

Leider nur ein informelles Konzept auf rein intuitiver Basis.

## Der Euklidische Algorithmus

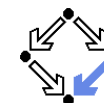


Euklid (um 300 v.Chr.): *Die Elemente.*

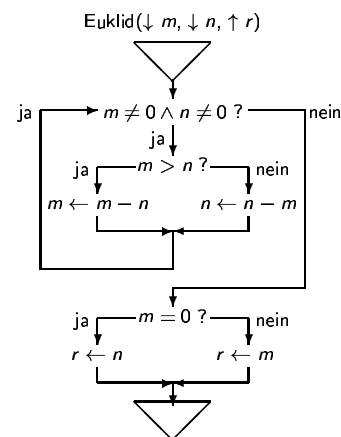
- Klassische Formulierung (geometrisch)
  - Problem: Suche nach dem gemeinsamen "Maß" der Längen zweier Linien AB und CD.  
... Wenn CD aber AB nicht mißt, und man nimmt bei AB, CD abwechselnd immer das kleinere vom größeren weg, dann muss (schließlich) eine Zahl übrig bleiben, die die vorangehende misst.
- Moderne Formulierung (arithmetisch)
  - Problem: Suche nach dem größten gemeinsamen Teiler zweier natürlicher Zahlen  $m$  und  $n$ , die nicht beide Null sind.
    1. Wenn  $m = 0$ , dann ist das Ergebnis  $n$ .
    2. Wenn  $n = 0$ , dann ist das Ergebnis  $m$ .
    3. Wenn  $m > n$ , setze  $m$  auf  $m - n$  und fahre mit Schritt 1 fort.
    4. Ansonsten setze  $n$  auf  $n - m$  und fahre mit Schritt 1 fort.

Siehe Skriptum für eine optimierte Variante.

## Der Euklidische Algorithmus



Darstellung durch Ablaufdiagramm bzw. strukturiertes Programm.

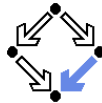


Euklid( $\downarrow m, \downarrow n, \uparrow r$ ):

```

while m ≠ 0 ∧ n ≠ 0 do
  if m > n
  then m ← m - n
  else n ← n - m
if m = 0
then r ← n
else r ← m
end Euklid.
  
```

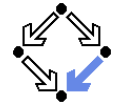
## Algorithmus und Funktion



- **Algorithmus:** eine eindeutige Rechenvorschrift.
  - Zum Beispiel der *Euklidische Algorithmus*.
    - Eingabe: zwei natürliche Zahlen  $m$  und  $n$ .
    - Ausgabe: eine natürliche Zahl  $r$ .
    - Vorschrift: ...
  - Dieser Algorithmus berechnet eine *mathematische Funktion*.
- **Funktion:** eine eindeutige Abbildung von Eingaben auf Ausgaben.
  - Zum Beispiel der *größte gemeinsame Teiler*.
$$\text{ggT} : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$$
$$\text{ggT}(m, n) = r$$
wobei  $r$  die größte natürliche Zahl ist, die  $m$  und  $n$  teilt.
  - Der größte gemeinsame Teiler wird z.B. durch den Euklidischen Algorithmus berechnet.

Verschiedene Algorithmen können die gleiche Funktion berechnen.

## Das Problem des Algorithmus-Begriffs

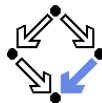


Für manche Fragestellungen ist der Begriff "Algorithmus" zu schwammig.

- Gibt es *irgendeinen* Algorithmus zur Lösung eines bestimmten Problems (bzw. zur Berechnung einer bestimmten Funktion)?
  - Problem: entscheide, ob ein beliebiges Programm terminiert.
    - *Halte-Problem*: unlösbar!
  - Problem: entscheide, ob zwei beliebige Programme das gleiche Ein/Ausgabe-Verhalten haben.
    - Unlösbar!
  - Problem: entscheide, ob eine beliebige Aussage über den natürlichen Zahlen gültig ist.
    - Gödels *Unvollständigkeitssatz* (1931): unlösbar!

In den 1930ern wurden verschiedene formale Modelle entwickelt, um den Begriff "Algorithmus" genauer zu fassen.

## Die weitere Landkarte



- **Endliche Automaten**
  - Spielen eine wichtige Rolle in der Informatik.
    - Lexikalische Analyse, Model Checking, ...
  - Erkennen *reguläre Sprachen*.
- **Random Access Machines (RAMs)**
  - Mächtiger als endliche Automaten.
  - Gleichmächtig der *Random Access Stored Program Machine (RASP)*.
    - Beschreibt in formaler Form den Aufbau eines Computers.
- **Turing-Maschinen**
  - Gleiche Mächtigkeit wie RAMs, aber einfacher.
  - Erkennt *rekursiv aufzählbare Sprachen*.
- **Rekursive Funktionen**
  - Gleiche Mächtigkeit wie Turing-Maschinen, aber rein mathematisch.
    - Gleiche Mächtigkeit wie Programme mit while-Schleifen.
  - Hat als Unterklasse die *primitiv rekursiven Funktionen*.
    - Gleiche Mächtigkeit wie Programme mit Zählschleifen.

Church'sche These: ein Algorithmus ist, was als eine Turing-Maschine (oder RAM/RASP oder rekursive Funktion) beschreibbar ist.