Theoretische Informatik
Kapitel 6 – Berechenbarkeit

Sommersemester 2024

Dozentin: Mareike Mutz

im Wechsel mit

Prof. Dr. M. Leuschel

Prof. Dr. J. Rothe

Einige Fragen

Ziel: Wir suchen Antworten auf die folgenden Fragen:

- Was ist Berechenbarkeit?
  Wie kann man das intuitiv Berechenbare formal fassen?

- Was ist ein Algorithmus?

- Welche Indizien hat man dafür, dass ein formaler Algorithmenbegriff tatsächlich den Begriff des intuitiv Berechenbaren exakt einfängt?

Was ist Berechenbarkeit?

- Intuitiv versteht man unter „Berechenbarkeit“ alles, was sich algorithmisch lösen lässt.

- Wir beschränken uns auf die Berechenbarkeit von Funktionen

$$f : \mathbb{N}^k \to \mathbb{N}$$

bzw. von Wortfunktionen

$$f : \Sigma^* \to \Sigma^*.$$

Was ist Berechenbarkeit?

- Intuitiv versteht man unter „Berechenbarkeit“ alles, was sich algorithmisch lösen lässt.

- Wir beschränken uns auf die Berechenbarkeit von Funktionen

$$f : \mathbb{N}^k \to \mathbb{N}$$

bzw. von Wortfunktionen

$$f : \Sigma^* \to \Sigma^*.$$

- Zahlen $n \in \mathbb{N} = \{0, 1, 2, 3, \ldots\}$ können beispielsweise durch ihre Binärdarstellung als Wörter über dem Alphabet $\Sigma = \{0, 1\}$ dargestellt werden.

Was ist Berechenbarkeit?

- Zu einer natürlichen Zahl n bezeichnen wir mit $\text{bin}(n)$ die *Binärdarstellung von n ohne führende Nullen*, z.B. $\text{bin}(19) = 10011$ und $\text{bin}(5) = 101$.

- Damit kann man durch

$$\text{bin} : \mathbb{N} \mapsto \{0,1\}^*$$

in natürlicher Weise eine Injektion zwischen $\mathbb{N}$ und $\{0,1\}^*$ angeben.

- Die Berechenbarkeit z.B. reellwertiger Funktionen wird hier nicht betrachtet.

Was ist Berechenbarkeit?

Definition

Es sei $f : A \to B$ eine Funktion.

- Die Menge A heißt *Eingabemenge* oder *Urbildbereich* und die Menge B *Ausgabemenge* oder *Bildbereich von f*.
- Der *Definitionsbereich D_f der Funktion f* ist die Teilmenge von A, auf der f definiert ist.
- Der *Wertebereich W_f der Funktion f* ist die Menge $\{f(x) \mid x \in \mathrm{D}_f\}$.
- f ist stets eine *partielle* Funktion.

Was ist Berechenbarkeit?

Definition

Es sei $f : A \to B$ eine Funktion.

- Die Menge A heißt *Eingabemenge* oder *Urbildbereich* und die Menge B *Ausgabemenge* oder *Bildbereich von f*.
- Der *Definitionsbereich D_f der Funktion f* ist die Teilmenge von A, auf der f definiert ist.
- Der *Wertebereich W_f der Funktion f* ist die Menge $\{f(x) \mid x \in \mathrm{D}_f\}$.
- f ist stets eine *partielle* Funktion.
- f heißt *total*, falls sie überall auf ihrer Eingabemenge definiert ist, d.h., falls $A = \mathrm{D}_f$ gilt.

Was ist Berechenbarkeit?

Beispiel:

- Die Funktion $f_1 : \mathbb{N} \to \mathbb{N}$ mit

$$f_1(n) = n + 1$$

ist total (und partiell), da $D_f = \mathbb{N}$.

Was ist Berechenbarkeit?

Beispiel:

- Die Funktion $f_1 : \mathbb{N} \to \mathbb{N}$ mit

$$f_1(n) = n + 1$$

ist total (und partiell), da $D_f = \mathbb{N}$.

- Die Funktion $f_2 : \mathbb{N}^2 \to \mathbb{N}$ mit

$$f_2(n_1, n_2) = n_1 \text{ div } n_2$$

ist nicht-total (und partiell), da $D_f = \mathbb{N}^2 - \{(n, 0) \mid n \in \mathbb{N}\}$.

Was ist ein Algorithmus?

- **Intuitiv:** Ein „Algorithmus“ ist eine endliche Folge von Befehlen oder Anweisungen, die eine Eingabe in eine bestimmte Ausgabe in endlich vielen Rechenschritten transformieren.

Was ist ein Algorithmus?

- **Intuitiv:** Ein „Algorithmus“ ist eine endliche Folge von Befehlen oder Anweisungen, die eine Eingabe in eine bestimmte Ausgabe in endlich vielen Rechenschritten transformieren.

- **Formal:** Ein *Algorithmus / Programm A berechnet eine Funktion* $f : \mathbb{N}^k \to \mathbb{N}$ genau dann, wenn gilt:
  1. Für alle $(n_1, n_2, \ldots, n_k) \in \mathrm{D}_f$ hält der Algorithmus A bei Eingabe $(n_1, n_2, \ldots, n_k)$ nach endlich vielen Schritten mit der Ausgabe $f(n_1, n_2, \ldots, n_k)$ an.

Was ist ein Algorithmus?

- **Intuitiv:** Ein „Algorithmus“ ist eine endliche Folge von Befehlen oder Anweisungen, die eine Eingabe in eine bestimmte Ausgabe in endlich vielen Rechenschritten transformieren.

- **Formal:** Ein *Algorithmus / Programm A berechnet eine Funktion* $f : \mathbb{N}^k \to \mathbb{N}$ genau dann, wenn gilt:
  1. Für alle $(n_1, n_2, \ldots, n_k) \in \mathrm{D}_f$ hält der Algorithmus A bei Eingabe $(n_1, n_2, \ldots, n_k)$ nach endlich vielen Schritten mit der Ausgabe $f(n_1, n_2, \ldots, n_k)$ an.
  2. Für alle $(n_1, n_2, \ldots, n_k) \notin \mathrm{D}_f$ hält A bei Eingabe $(n_1, n_2, \ldots, n_k)$ nie an (ist also z.B. in einer Endlosschleife).
     (Alternativ hierzu könnte man verlangen, dass der Algorithmus zwar hält, aber die Eingabe verwirft.)

Was ist Berechenbarkeit?

- Nur intuitiv zu argumentieren, dass eine bestimmte Funktion f *nicht* berechenbar ist, ist unmöglich.
- Für einen solchen Nachweis ist es unabdingbar, dass man den Algorithmenbegriff im mathematischen Sinne formalisiert und zeigt, dass f durch *keinen* Algorithmus dieser formal definierten Klasse berechnet werden kann.
- Beispiele von formalisierten Algorithmenklassen:

Was ist Berechenbarkeit?

- Nur intuitiv zu argumentieren, dass eine bestimmte Funktion f *nicht* berechenbar ist, ist unmöglich.
- Für einen solchen Nachweis ist es unabdingbar, dass man den Algorithmenbegriff im mathematischen Sinne formalisiert und zeigt, dass f durch *keinen* Algorithmus dieser formal definierten Klasse berechnet werden kann.
- Beispiele von formalisierten Algorithmenklassen:
  - Programme in einer fixierten Programmiersprache, z.B. Java, C, ...;
  - endliche Automaten (DFAs bzw. NFAs);
  - Kellerautomaten (PDAs);
  - deterministische Kellerautomaten (DPDAs);
  - linear beschränkte Automaten (LBAs);
  - Turingmaschinen (TMs).

Was ist Berechenbarkeit?

- Alle diese Modelle können so modifiziert werden, dass sie Funktionen berechnen, nicht Sprachen entscheiden.
- Beispielsweise gibt es Funktionen, die sich zwar durch Turingmaschinen, nicht aber durch endliche Automaten berechnen lassen.
- Hat man eine Algorithmenklasse fixiert, so tritt das neue Problem auf, festzustellen, ob sie wirklich genau den Begriff des intuitiv Berechenbaren erfasst.
- Dies kann nur intuitiv begründet, nicht aber formal bewiesen werden.

Was ist Berechenbarkeit?

Beispiel:

- Sei

$$\tilde{\pi} = 1415\cdots$$

die Folge der Nachkommaziffern der Zahl $\pi = 3,1415\cdots$.

- Die Anfangswortrelation und Teilwortrelation wird mit $\sqsubseteq_a$ bzw. $\sqsubseteq$ bezeichnet.
- Betrachte die folgenden Funktionen und überlege, ob sie *intuitiv* berechenbar sind oder nicht:

1. Definiere die Funktion $f : \{0,1,\ldots,9\}^* \to \{0,1\}$ durch

$$f(n) = \begin{cases} 1 & \text{falls } n \sqsubseteq_a \tilde{\pi} = 1415\cdots \\ 0 & \text{sonst.} \end{cases}$$

Was ist Berechenbarkeit?

Offenbar ist f berechenbar.

Was ist Berechenbarkeit?

Offenbar ist f berechenbar.

Denn es gibt Näherungsverfahren für π, die nur bis zur durch die Länge von n vorgegebenen Genauigkeit laufen müssen, also in endlicher Zeit zum Ergebnis kommen.

Was ist Berechenbarkeit?

Offenbar ist f berechenbar.
Denn es gibt Näherungsverfahren für π, die nur bis zur durch die Länge von n vorgegebenen Genauigkeit laufen müssen, also in endlicher Zeit zum Ergebnis kommen.

2. Definiere die Funktion $g : \{0, 1, \ldots, 9\}^* \to \{0, 1\}$ durch

$$g(n) = \begin{cases} 1 & \text{falls } n \sqsubseteq \tilde{\pi} = 1415\cdots \\ 0 & \text{sonst.} \end{cases}$$

Was ist Berechenbarkeit?

Es ist offen, ob g berechenbar ist.

Was ist Berechenbarkeit?

Es ist offen, ob g berechenbar ist.

Wäre die Zahl π so „zufällig“, dass *jede* endliche Ziffernfolge als ein Teilwort in $\tilde{\pi}$ erscheint, dann wäre

$$g \equiv 1$$

(d.h., $g(n) = 1$ für alle n) und somit berechenbar.

Was ist Berechenbarkeit?

Es ist offen, ob g berechenbar ist.

Wäre die Zahl π so „zufällig“, dass *jede* endliche Ziffernfolge als ein Teilwort in $\tilde{\pi}$ erscheint, dann wäre

$$g \equiv 1$$

(d.h., $g(n) = 1$ für alle n) und somit berechenbar.

3. Definiere die Funktion $h : \mathbb{N} \to \{0, 1\}$ durch

$$h(n) = \begin{cases} 1 & \text{falls in } \tilde{\pi} = 1415\cdots \text{ mindestens } n\text{-mal} \\ & \quad \text{hintereinander eine 7 vorkommt} \\ 0 & \text{sonst.} \end{cases}$$

Was ist Berechenbarkeit?

Obwohl wir nicht wissen, ob $h(n) = 1$ für jedes n ist, ist die Funktion h berechenbar!

Was ist Berechenbarkeit?

Obwohl wir nicht wissen, ob $h(n) = 1$ für jedes n ist, ist die Funktion h berechenbar!

Begründung:

Fall 1: Es gibt in $\tilde{\pi}$ beliebig lange Blöcke $77 \cdots 7$.

Dann ist $h(n) = 1$ für alle n und h somit berechenbar.

Was ist Berechenbarkeit?

Obwohl wir nicht wissen, ob $h(n) = 1$ für jedes n ist, ist die Funktion h berechenbar!

Begründung:

Fall 1: Es gibt in $\tilde{\pi}$ beliebig lange Blöcke $77 \cdots 7$.
Dann ist $h(n) = 1$ für alle n und h somit berechenbar.

Fall 2: Es gibt ein n_0, so dass in $\tilde{\pi}$ Blöcke der Form $77 \cdots 7$ bis zur Länge n_0 vorkommen, aber keine der Länge $n_0 + 1$.
Dann gilt

$$h(n) = \begin{cases} 1 & \text{falls } n \leq n_0 \\ 0 & \text{sonst.} \end{cases}$$

Was ist Berechenbarkeit?

Das heißt, „Berechenbarkeit“ ist nicht-konstruktiv definiert: Es genügt, die *Existenz* eines Algorithmus für h zu beweisen; wir müssen ihn nicht explizit angeben können.

Was ist Berechenbarkeit?

Das heißt, „Berechenbarkeit“ ist nicht-konstruktiv definiert: Es genügt, die *Existenz* eines Algorithmus für h zu beweisen; wir müssen ihn nicht explizit angeben können.

4. Definiere die Funktion $i : \mathbb{N} \to \{0, 1\}$ durch

$$i(n) = \begin{cases} 1 & \text{falls das 1. LBA Problem eine positive Lösung hat} \\ 0 & \text{sonst.} \end{cases}$$

Was ist Berechenbarkeit?

Die Funktion *i* ist berechenbar, auch wenn wir das 1. LBA Problem derzeit nicht lösen können.

Was ist Berechenbarkeit?

Die Funktion i ist berechenbar, auch wenn wir das 1. LBA Problem derzeit nicht lösen können.

Denn

- entweder ist $i \equiv 1$
- oder $i \equiv 0$,

und beide konstante Funktionen sind selbstverständlich berechenbar.

Was ist Berechenbarkeit?

Im ersten Beispiel oben ordneten wir der reellen Zahl π die Funktion

$$f(n) = f_\pi(n) = \begin{cases} 1 & \text{falls } n \sqsubseteq_a \tilde{\pi} = 1415\cdots \\ 0 & \text{sonst} \end{cases}$$

zu.

Was ist Berechenbarkeit?

Im ersten Beispiel oben ordneten wir der reellen Zahl π die Funktion

$$f(n) = f_\pi(n) = \begin{cases} 1 & \text{falls } n \sqsubseteq_a \tilde{\pi} = 1415\cdots \\ 0 & \text{sonst} \end{cases}$$

zu. Dies kann man entsprechend für jede reelle Zahl r tun und erhält so die Funktion

$$f_r(n) = \begin{cases} 1 & \text{falls } n \sqsubseteq_a \tilde{r} = \text{Nachkommastellen von } r \\ 0 & \text{sonst.} \end{cases}$$

Was ist Berechenbarkeit?

Im ersten Beispiel oben ordneten wir der reellen Zahl π die Funktion

$$f(n) = f_\pi(n) = \begin{cases} 1 & \text{falls } n \sqsubseteq_a \tilde{\pi} = 1415\cdots \\ 0 & \text{sonst} \end{cases}$$

zu. Dies kann man entsprechend für jede reelle Zahl r tun und erhält so die Funktion

$$f_r(n) = \begin{cases} 1 & \text{falls } n \sqsubseteq_a \tilde{r} = \text{Nachkommastellen von } r \\ 0 & \text{sonst.} \end{cases}$$

Ist f_r für jede reelle Zahl r berechenbar?

Was ist Berechenbarkeit?

Nein!

Was ist Berechenbarkeit?

Nein! Denn:

- Die Menge $\mathbb{R}$ der reellen Zahlen in $[0,1)$ ist überabzählbar.

- Sind r und r' Zahlen in $[0,1)$ von $\mathbb{R}$ mit $r \neq r'$, so gilt $f_r \not\equiv f_{r'}$.
  Somit haben f_r und $f_{r'}$ *verschiedene* Berechnungsalgorithmen.

Was ist Berechenbarkeit?

Nein! Denn:

- Die Menge $\mathbb{R}$ der reellen Zahlen in $[0, 1)$ ist überabzählbar.
- Sind r und r' Zahlen in $[0, 1)$ von $\mathbb{R}$ mit $r \neq r'$, so gilt $f_r \not\equiv f_{r'}$. Somit haben f_r und $f_{r'}$ *verschiedene* Berechnungsalgorithmen.
- Die Menge aller Algorithmen (egal in welcher fest gewählten Formalisierung) ist aber nur abzählbar unendlich, da sich ein jeder Algorithmus durch einen endlichen Text beschreiben lassen muss.

Formalisierungen des Algorithmenbegriffs

Es sind seit den 30er Jahren des 20. Jahrhunderts eine Reihe von Formalisierungen des Algorithmenbegriffs vorgeschlagen worden, beispielsweise:

- die Turing-Berechenbarkeit von Turing;
- der λ-Kalkül von Church und Rosser;
- die Markov-Berechenbarkeit von Markov;
- der Gleichungskalkül von Gödel und Herbrand;
- die partiell rekursiven Funktionen von Kleene,
- weitere Formalisierungen von Post und anderen.

Churchsche These

These: Die durch die formale Definition

1. der Turing-Berechenbarkeit,
2. der WHILE-Berechenbarkeit,
3. der GOTO-Berechenbarkeit,
4. der μ-Rekursivität,
5. der Markov-Berechenbarkeit,
6. des λ-Kalküls
7. und einer Reihe von anderen Formalisierungen des Algorithmenbegriffs

beschriebenen Klassen von Funktionen stimmen jeweils genau mit der Klasse der im intuitiven Sinne berechenbaren Funktionen überein.

Churchsche These

Bemerkung:

- Die These ist natürlich unmöglich zu beweisen, da der intuitive Begriff der Berechenbarkeit nicht formal definierbar ist.

Churchsche These

Bemerkung:

- Die These ist natürlich unmöglich zu beweisen, da der intuitive Begriff der Berechenbarkeit nicht formal definierbar ist.

- Die These ist allgemein akzeptiert.

Churchsche These

Bemerkung:

- Die These ist natürlich unmöglich zu beweisen, da der intuitive Begriff der Berechenbarkeit nicht formal definierbar ist.
- Die These ist allgemein akzeptiert.
- Einige dieser Formalisierungen des Algorithmenbegriffs werden wir im Weiteren kennenlernen.

Historischer Kontext

- um 1880: **Georg Cantor** begründet seine axiomatische Mengentheorie.

Historischer Kontext

- um 1880: **Georg Cantor** begründet seine axiomatische Mengentheorie.
- 1900: **David Hilbert** unternimmt den Versuch, die gesamte Mathematik auf ein einheitliches axiomatisches Fundament zu stellen. Forschungsprogramm:
  - Vollständigkeit der Mathematik.
  - Widerspruchsfreiheit der Mathematik.

Historischer Kontext

- um 1880: **Georg Cantor** begründet seine axiomatische Mengentheorie.
- 1900: **David Hilbert** unternimmt den Versuch, die gesamte Mathematik auf ein einheitliches axiomatisches Fundament zu stellen. Forschungsprogramm:
  - Vollständigkeit der Mathematik.
  - Widerspruchsfreiheit der Mathematik.
- 1901: **Bertrand Russells** Paradoxon:

  „Enthält die Menge $R = \{X \mid X \notin X\}$ sich selbst als Element?“

  verursacht zunächst viel Verwirrung, trägt aber dann dazu bei, die Axiome der Mengentheorie korrekt zu formulieren.

Historischer Kontext

- 1902: **Gottlieb Frege** veröffentlicht sein einflussreiches Werk „Grundgesetze der Arithmetik“.

Historischer Kontext

- 1902: **Gottlieb Frege** veröffentlicht sein einflussreiches Werk „Grundgesetze der Arithmetik“.

- 1931: **Kurt Gödels** Unvollständigkeitssätze beschäftigen sich ebenfalls mit dem Begriff des mathematischen Beweises und mit den Grundlagen der Logik.

Historischer Kontext

- 1902: **Gottlieb Frege** veröffentlicht sein einflussreiches Werk „Grundgesetze der Arithmetik“.

- 1931: **Kurt Gödels** Unvollständigkeitssätze beschäftigen sich ebenfalls mit dem Begriff des mathematischen Beweises und mit den Grundlagen der Logik.

- 1936: **Alan Turing** knüpft an Gödels Arbeit an und fragt nach der *algorithmischen* Entscheidbarkeit von Problemen.
  Er führt dazu das grundlegende Modell der Turingmaschine ein, das nach der Churchschen These genau das intuitiv Berechenbare erfasst.

Gödels Unvollständigkeitssätze (informal)

Theorem (1. Satz von Gödel)

Wenn die axiomatische Mengentheorie widerspruchsfrei ist, dann gibt es in ihr Sätze, die innerhalb der Theorie weder bewiesen noch widerlegt werden können.
Das heißt, die Widerspruchsfreiheit (Konsistenz) einer Theorie impliziert ihre Unvollständigkeit. In diesem Sinne ist jede hinreichend ausdrucksstarke Theorie unvollständig.

Gödels Unvollständigkeitssätze (informal)

Theorem (1. Satz von Gödel)

Wenn die axiomatische Mengentheorie widerspruchsfrei ist, dann gibt es in ihr Sätze, die innerhalb der Theorie weder bewiesen noch widerlegt werden können.
Das heißt, die Widerspruchsfreiheit (Konsistenz) einer Theorie impliziert ihre Unvollständigkeit. In diesem Sinne ist jede hinreichend ausdrucksstarke Theorie unvollständig.

Theorem (2. Satz von Gödel)

Es gibt kein konstruktives Verfahren, das die Konsistenz einer axiomatischen Theorie beweisen könnte.

Gödels Unvollständigkeitssätze (informal)

Theorem (1. Satz von Gödel)

Wenn die axiomatische Mengentheorie widerspruchsfrei ist, dann gibt es in ihr Sätze, die innerhalb der Theorie weder bewiesen noch widerlegt werden können.
Das heißt, die Widerspruchsfreiheit (Konsistenz) einer Theorie impliziert ihre Unvollständigkeit. In diesem Sinne ist jede hinreichend ausdrucksstarke Theorie unvollständig.

Theorem (2. Satz von Gödel)

Es gibt kein konstruktives Verfahren, das die Konsistenz einer axiomatischen Theorie beweisen könnte.

Dies zeigt, dass Hilberts Programm scheitern muss.