

Theorie der Informatik

11. Kontextsensitive und Typ-0-Sprachen

Malte Helmert Gabriele Röger

Universität Basel

7. April 2014

Kontextsensitive und allgemeine Grammatiken

Wiederholung: (kontextsensitive) Grammatiken

Definition (Grammatik)

Eine **Grammatik** ist ein 4-Tupel (Σ, V, P, S) mit

- Σ endliches Terminalalphabet,
- V endliche Menge von Variablen (mit $V \cap \Sigma = \emptyset$),
- $P \subseteq (V \cup \Sigma)^+ \times (V \cup \Sigma)^*$ endliche Menge von Regeln, und
- $S \in V$ Startvariable.

Typ 0 und Typ 1

- Jede **Grammatik** ist eine **Typ-0-Grammatik**.
- Bei **kontextsensitiven (Typ-1)** Grammatiken gilt für alle Regeln $w_1 \rightarrow w_2$, dass $|w_1| \leq |w_2|$.
Einzige Ausnahme: Regel $S \rightarrow \varepsilon$ ist für Startsymbol S erlaubt, falls S auf keiner rechten Regelseite vorkommt.

Fragen

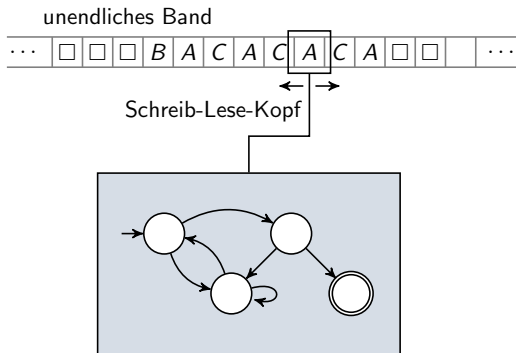


Fragen?

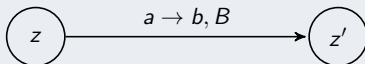
Turingmaschinen

- Britischer Logiker, Mathematiker, Kryptoanalytiker und Informatiker
- Wichtigste Arbeit (für uns):
On Computable Numbers,
with an Application to the
„Entscheidungsproblem“
→ **Turingmaschine**
- Mitarbeit an **Enigma-Entschlüsselung**
- Wegen Homosexualität zu
Gefängnisstrafe verurteilt,
wurde aber im Dezember 2013 von
Elizabeth II begnadigt
- **Turing-Award** wichtigster
Wissenschaftspreis in der Informatik

Turingmaschine: konzeptuell

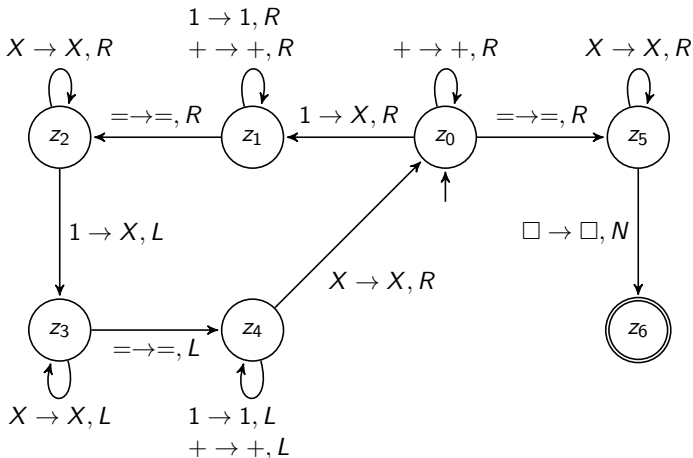


- Z endliche, nicht-leere Menge von Zuständen
- $\Sigma \neq \emptyset$ endliches Eingabealphabet
- $\Gamma \supset \Sigma$ endliches Bandalphabet
- $\delta : (Z \setminus E) \times \Gamma \rightarrow \mathcal{P}(Z \times \Gamma \times \{L, R, N\})$ Übergangsfunktion
- $z_0 \in Z$ Startzustand
- $\square \in \Gamma \setminus \Sigma$ Blank-Zeichen
- $E \subset Z$ Endzustände



Nichtdeterministische Turingmaschine: Beispiel

$$M = (\{z_0, z_1, \dots, z_6\}, \{1, +, =\}, \{1, +, =, X, \square\}, \delta, z_0, \square, \{z_6\})$$



Turingmaschine: Berechnungsschritt

Definition (Berechnungsschritt einer Turingmaschine)

Eine NTM $M = \langle Z, \Sigma, \Gamma, \delta, z_0, \square, E \rangle$ kann nach den folgenden Regeln in einem Berechnungsschritt von Konfiguration k in Konfiguration k' übergehen ($k \vdash_M k'$):

$$a_1 \dots a_m z b_1 \dots b_n \vdash_M$$

$$\left\{ \begin{array}{l} a_1 \dots a_m z' c b_2 \dots b_n, \text{ falls } (z', c, N) \in \delta(z, b_1), m \geq 0, n \geq 1 \\ a_1 \dots a_m c z' b_2 \dots b_n, \text{ falls } (z', c, R) \in \delta(z, b_1), m \geq 0, n \geq 2 \\ a_1 \dots a_{m-1} z' a_m c b_2 \dots b_n, \text{ falls } (z', c, L) \in \delta(z, b_1), m \geq 1, n \geq 1 \\ a_1 \dots a_m c z' \square, \text{ falls } (z', c, R) \in \delta(z, b_1), m \geq 0, n = 1 \\ z' \square c b_2 \dots b_n, \text{ falls } (z', c, L) \in \delta(z, b_1), m = 0, n \geq 1 \end{array} \right.$$

Turingmaschine: Erreichbarkeit von Konfigurationen

Definition (Erreichbare Konfiguration)

Konfiguration k' ist mit NTM M von Konfiguration k aus **erreichbar** ($k \vdash_M^* k'$), falls $k = k'$ oder es gibt Konfigurationen $k_0, \dots, k_n$ ($n \geq 1$), so dass

- $k_0 = k$,
- $k_i \vdash_M k_{i+1}$ für $i \in \{0, \dots, n-1\}$, und
- $k_n = k'$.

Turingmaschine: Erkanntes Wort

Definition (erkanntes Wort bei Turingmaschinen)

NTM $M = \langle Z, \Sigma, \Gamma, \delta, z_0, \square, E \rangle$ **erkennt das Wort** w genau dann, wenn M von der **Startkonfiguration** $z_0 w$ mit endlich vielen Berechnungsschritten in eine Konfiguration mit einem **Endzustand** übergehen kann:

M erkennt w gdw. $z_0 w \vdash_M^* w_1 z w_2$ mit $z \in E$, $w_1 \in \Gamma^*$, $w_2 \in \Gamma^+$.

Spezialfall: Für $w = \varepsilon$ ist die **Startkonfiguration** $z_0 \square$.

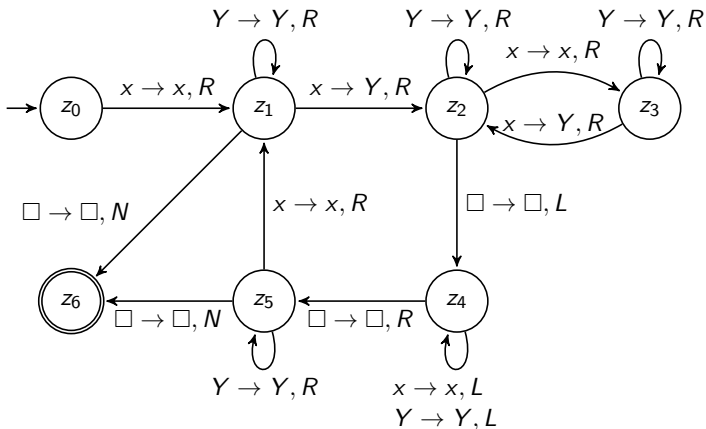
Beispiel: Tafel

Quiz



Quiz

$$M = (\{z_0, z_1, \dots, z_6\}, \{x\}, \{x, Y, \square\}, \delta, z_0, \square, \{z_6, z_7\})$$



Turingmaschine: Akzeptierte Sprache

Definition (akzeptierte Sprache einer NTM)

Sei M eine nichtdeterministische Turingmaschine mit Eingabealphabet Σ . Die von M **akzeptierte Sprache** ist definiert durch

$$\mathcal{L}(M) = \{w \in \Sigma^* \mid w \text{ wird von } M \text{ erkannt}\}.$$

Beispiel: Tafel

Fragen



Fragen?

Linear beschränkte Turingmaschinen

Brauchen wir
nicht unterschiedliche
Automatenmodelle für
kontextsensitive und
Typ-0-Sprachen?



Linear beschränkte Turingmaschinen: Idee

- Linear beschränkte Turingmaschinen dürfen nur den Teil des Bandes verwenden, der durch das Eingabewort belegt wird.

Linear beschränkte Turingmaschinen: Idee

- **Linear beschränkte** Turingmaschinen dürfen nur den **Teil des Bandes verwenden, der durch das Eingabewort belegt wird.**
- Kann vorderes Ende der Eingabe im ersten Schritt markieren.

Linear beschränkte Turingmaschinen: Idee

- **Linear beschränkte** Turingmaschinen dürfen nur den **Teil des Bandes verwenden, der durch das Eingabewort belegt wird.**
- Kann vorderes Ende der Eingabe im ersten Schritt markieren.
- Problem: Muss verhindern, über das hintere Bandende hinauszulaufen

Linear beschränkte Turingmaschinen: Idee

- **Linear beschränkte** Turingmaschinen dürfen nur den **Teil des Bandes verwenden, der durch das Eingabewort belegt wird.**
- Kann vorderes Ende der Eingabe im ersten Schritt markieren.
- Problem: Muss verhindern, über das hintere Bandende hinauszulaufen
- Für Sprache über Σ hat linear beschränkte Turingmaschine **modifiziertes Eingabealphabet $\Sigma \cup \{\hat{a} \mid a \in \Sigma\}$.**

Linear beschränkte Turingmaschinen: Idee

- **Linear beschränkte** Turingmaschinen dürfen nur den **Teil des Bandes verwenden, der durch das Eingabewort belegt wird**.
- Kann vorderes Ende der Eingabe im ersten Schritt markieren.
- Problem: Muss verhindern, über das hintere Bandende hinauszulaufen
- Für Sprache über Σ hat linear beschränkte Turingmaschine **modifiziertes Eingabealphabet $\Sigma \cup \{\hat{a} \mid a \in \Sigma\}$** .
- Eigentliche Eingabe $a_1 \dots a_n$ wird auf dem Band repräsentiert als $a_1 \dots a_{n-1} \hat{a}_n$.

Linear beschränkte Turingmaschinen: Definition

Definition (Linear beschränkte NTM)

Eine (nichtdeterministische) Turingmaschine heisst **linear beschränkt**, wenn für alle $a_1 \dots a_n \in \Sigma^+$ und alle Konfigurationen $w_1 z w_2$ mit $z_0 a_1 \dots a_{n-1} \hat{a}_n \vdash^* w_1 z w_2$ gilt $|w_1 w_2| = n$.

Linear beschränkte NTMs akzeptieren Typ-1-Sprachen

Satz

Die von linear beschränkten, nichtdeterministischen Turingmaschinen (LBAs) akzeptierbaren Sprachen sind genau die kontextsensitiven (Typ 1) Sprachen.

Linear beschränkte NTMs akzeptieren Typ-1-Sprachen

Satz

Die von linear beschränkten, nichtdeterministischen Turingmaschinen (LBAs) akzeptierbaren Sprachen sind genau die kontextsensitiven (Typ 1) Sprachen.

Beweis.

„ $\Leftarrow$ “ Sei $G = (V, \Sigma, P, S)$ eine kontextsensitive Grammatik. Wir beschreiben eine TM M mit $\mathcal{L}(M) = \mathcal{L}(G)$

Linear beschränkte NTMs akzeptieren Typ-1-Sprachen

Beweis (Fortsetzung).

Gestartet auf Eingabe $x \in \Sigma^*$ wählt M nichtdeterministisch eine Regel $u \rightarrow v \in P$ und sucht ein Vorkommen von v auf dem Band. Falls sie eines findet, ersetzt sie v durch u , wobei im Fall $|u| < |v|$ die Bandsymbole hinter u entsprechend nach vorne verschoben werden. Falls der nicht-leere Teil des Bandes nur noch aus der Startvariable S besteht, wechselt die TM in einen Endzustand. Sonst wiederholt sie die nichtdeterministische Ersetzung. Es gilt:

Linear beschränkte NTMs akzeptieren Typ-1-Sprachen

Beweis (Fortsetzung).

Gestartet auf Eingabe $x \in \Sigma^*$ wählt M nichtdeterministisch eine Regel $u \rightarrow v \in P$ und sucht ein Vorkommen von v auf dem Band. Falls sie eines findet, ersetzt sie v durch u , wobei im Fall $|u| < |v|$ die Bandsymbole hinter u entsprechend nach vorne verschoben werden. Falls der nicht-leere Teil des Bandes nur noch aus der Startvariable S besteht, wechselt die TM in einen Endzustand. Sonst wiederholt sie die nichtdeterministische Ersetzung. Es gilt:

$x \in \mathcal{L}(G)$ gdw. es gibt Ableitung $S \Rightarrow \dots \Rightarrow x$ mit G

gdw. es gibt Rechnung von M , die diese Ableitung
in umgekehrter Richtung simuliert

gdw. $x \in \mathcal{L}(M)$

Linear beschränkte NTMs akzeptieren Typ-1-Sprachen

Beweis (Fortsetzung).

Gestartet auf Eingabe $x \in \Sigma^*$ wählt M nichtdeterministisch eine Regel $u \rightarrow v \in P$ und sucht ein Vorkommen von v auf dem Band. Falls sie eines findet, ersetzt sie v durch u , wobei im Fall $|u| < |v|$ die Bandsymbole hinter u entsprechend nach vorne verschoben werden. Falls der nicht-leere Teil des Bandes nur noch aus der Startvariable S besteht, wechselt die TM in einen Endzustand. Sonst wiederholt sie die nichtdeterministische Ersetzung. Es gilt:

$x \in \mathcal{L}(G)$ gdw. es gibt Ableitung $S \Rightarrow \dots \Rightarrow x$ mit G

gdw. es gibt Rechnung von M , die diese Ableitung
in umgekehrter Richtung simuliert

gdw. $x \in \mathcal{L}(M)$

Da für $u \rightarrow v \in P$ gilt, dass $|u| \leq |v|$, ist M linear beschränkt.

Linear beschränkte NTMs akzeptieren Typ-1-Sprachen

Beweis (Fortsetzung).

„ $\Rightarrow$ “ Sei M eine LBA. Wir konstruieren Grammatik G , die auf Wörtern operiert, die Konfigurationen von M entsprechen.

Die Wörter sollen nicht länger werden als die Eingabe. Verwende daher Alphabet $\Delta = \Gamma \cup (Z \times \Gamma)$. Eine Konfiguration $azbcd$ wird dann z.B. durch das Wort $a(z, b)cd$ repräsentiert.

Linear beschränkte NTMs akzeptieren Typ-1-Sprachen

Beweis (Fortsetzung).

„ $\Rightarrow$ “ Sei M eine LBA. Wir konstruieren Grammatik G , die auf Wörtern operiert, die Konfigurationen von M entsprechen.

Die Wörter sollen nicht länger werden als die Eingabe. Verwende daher Alphabet $\Delta = \Gamma \cup (Z \times \Gamma)$. Eine Konfiguration $azbcd$ wird dann z.B. durch das Wort $a(z, b)cd$ repräsentiert.

Ein δ -Übergang $(z', b, L) \in \delta(z, a)$ von M kann dann zum Beispiel durch Grammatikregeln $c(z, a) \rightarrow (z', c)b$ für alle $c \in \Gamma$ beschrieben werden.

Die Menge aller solcher Regeln sei P' . Dann gilt $k \vdash_M^* k'$ gdw. $\tilde{k} \Rightarrow \tilde{k}'$ mit P' , wobei $\tilde{k}$ die oben beschriebene Darstellung von k ist.

...

Linear beschränkte NTMs akzeptieren Typ-1-Sprachen

Beweis (Fortsetzung).

Insgesamt wollen wir eine Grammatik konstruieren, die folgendermassen Wörter $a_1 \dots a_n$ ableiten kann:

- ① $S \Rightarrow^* ((z_0, a_1), a_1)(a_2, a_2) \dots (a_{n-1}, a_{n-1})(\hat{a}_n, a_n)$,
wobei die erste Komponente dabei eine TM-Konfiguration,
und die zweite Komponente das Eingabewort ist.
Idee: Starte von Startkonfiguration.
- ② $\dots \Rightarrow^* (\gamma_1, a_1) \dots (\gamma_n, a_n)$, wobei für ein $i \in \{1, \dots, n\}$ gilt
 $\gamma_i \in E \times \Gamma$ und für alle $j \neq i$ gilt $\gamma_j \in \Gamma$.
Idee: Simuliere Berechnung der TM auf erster Komponente
(mit Regeln aus P') bis Endzustand erreicht ist.
- ③ $\dots \Rightarrow^* a_1 \dots a_n$
Idee: Lösche dann erste Komponente.

Linear beschränkte NTMs akzeptieren Typ-1-Sprachen

Beweis (Fortsetzung).

Dies wird durch die Grammatik $G = (\Sigma, V, P, S)$ erreicht, wobei:

$$V = \{S, A\} \cup (\Delta \times \Sigma)$$



Linear beschränkte NTMs akzeptieren Typ-1-Sprachen

Beweis (Fortsetzung).

Dies wird durch die Grammatik $G = (\Sigma, V, P, S)$ erreicht, wobei:

$$V = \{S, A\} \cup (\Delta \times \Sigma)$$

$$P = \{S \rightarrow A(\hat{a}, a) \mid a \in \Sigma\} \cup$$

$$\{A \rightarrow A(a, a) \mid a \in \Sigma\} \cup$$

$$\{A \rightarrow ((z_0, a), a) \mid a \in \Sigma\} \cup$$



Linear beschränkte NTMs akzeptieren Typ-1-Sprachen

Beweis (Fortsetzung).

Dies wird durch die Grammatik $G = (\Sigma, V, P, S)$ erreicht, wobei:

$$V = \{S, A\} \cup (\Delta \times \Sigma)$$

$$P = \{S \rightarrow A(\hat{a}, a) \mid a \in \Sigma\} \cup$$

$$\{A \rightarrow A(a, a) \mid a \in \Sigma\} \cup$$

$$\{A \rightarrow ((z_0, a), a) \mid a \in \Sigma\} \cup$$

$$\{(\alpha_1, a)(\alpha_2, b) \rightarrow (\beta_1, a)(\beta_2, b) \mid \alpha_1\alpha_2 \rightarrow \beta_1\beta_2 \in P', \\ a, b \in \Sigma\} \cup$$



Linear beschränkte NTMs akzeptieren Typ-1-Sprachen

Beweis (Fortsetzung).

Dies wird durch die Grammatik $G = (\Sigma, V, P, S)$ erreicht, wobei:

$$V = \{S, A\} \cup (\Delta \times \Sigma)$$

$$P = \{S \rightarrow A(\hat{a}, a) \mid a \in \Sigma\} \cup$$

$$\{A \rightarrow A(a, a) \mid a \in \Sigma\} \cup$$

$$\{A \rightarrow ((z_0, a), a) \mid a \in \Sigma\} \cup$$

$$\{(\alpha_1, a)(\alpha_2, b) \rightarrow (\beta_1, a)(\beta_2, b) \mid \alpha_1\alpha_2 \rightarrow \beta_1\beta_2 \in P', \\ a, b \in \Sigma\} \cup$$

$$\{((z, a), b) \rightarrow b \mid z \in E, a \in \Gamma, b \in \Sigma\} \cup$$

$$\{(a, b) \rightarrow b \mid a \in \Gamma, b \in \Sigma\}$$

Offensichtlich sind alle Regeln vom Typ 1.



NTMs akzeptieren Typ-0-Sprachen

Satz

Die durch nichtdeterministische Turingmaschinen akzeptierbaren Sprachen sind genau die Typ-0-Sprachen.

Beweisskizze.

Analog zum Beweis für LBAs und kontextsensitive Sprachen.

Fragen



Fragen?

Deterministische Turingmaschinen

Definition (Deterministische Turingmaschine)

Eine **deterministische Turingmaschine (DTM)** ist eine Turingmaschine $M = \langle Z, \Sigma, \Gamma, \delta, z_0, \square, E \rangle$ mit $\delta : (Z \setminus E) \times \Gamma \rightarrow Z \times \Gamma \times \{L, R, N\}$.

Deterministische Turingmaschinen

Satz

Für jede Typ-0-Sprache L gibt es eine deterministische Turingmaschine M mit $\mathcal{L}(M) = L$.

Deterministische Turingmaschinen

Satz

Für jede Typ-0-Sprache L gibt es eine deterministische Turingmaschine M mit $\mathcal{L}(M) = L$.

Beweisskizze.

Sei M' eine nichtdeterministische Turingmaschine mit $\mathcal{L}(M') = L$. Man konstruiert eine deterministische Turingmaschine, die systematisch den Berechnungsbaum von M' nach einer Konfiguration mit Endzustand untersucht.

Deterministische Turingmaschinen

Satz

Für jede Typ-0-Sprache L gibt es eine deterministische Turingmaschine M mit $\mathcal{L}(M) = L$.

Beweisskizze.

Sei M' eine nichtdeterministische Turingmaschine mit $\mathcal{L}(M') = L$. Man konstruiert eine deterministische Turingmaschine, die systematisch den Berechnungsbaum von M' nach einer Konfiguration mit Endzustand untersucht.

Bemerkung: Es ist ein offenes Problem, ob der Satz analog für Typ-1-Sprachen und deterministische LBAs gilt.

Abschlusseigenschaften und Entscheidbarkeit

Abschlusseigenschaften

	Schnitt	Vereinigung	Komplement	Produkt	Stern
Typ 3	Ja	Ja	Ja	Ja	Ja
Typ 2	Nein	Ja	Nein	Ja	Ja
Typ 1	Ja ⁽¹⁾	Ja ⁽³⁾	Ja ⁽¹⁾	Ja ⁽³⁾	Ja ⁽³⁾
Typ 0	Ja ⁽³⁾	Ja ⁽³⁾	Nein ⁽²⁾	Ja ⁽³⁾	Ja ⁽³⁾

Beweise?

(1) ohne Beweis

(2) Beweis in späteren Vorlesungskapiteln

(3) Beweis in den Übungen

Entscheidbarkeit

	Wort- problem	Leerheits- problem	Äquivalenz- problem	Schnitt- problem
Typ 3	Ja	Ja	Ja	Ja
Typ 2	Ja	Ja	Nein	Nein
Typ 1	Ja	Nein ⁽¹⁾	Nein ⁽¹⁾	Nein ⁽¹⁾
Typ 0	Nein ⁽²⁾	Nein ⁽²⁾	Nein ⁽²⁾	Nein ⁽²⁾

(1) ohne Beweis

(2) Beweis in Vorlesungsabschnitt 3

Wortproblem bei kontextsensitiven Sprachen:

Mit Typ-1-Grammatiken werden Wörter durch Ableitungen niemals kürzer, daher kann man alle Ableitungsmöglichkeiten bis zur Grösse des gegebenen Wortes ausprobieren.

Fragen



Fragen?

Zusammenfassung

Zusammenfassung

- Turingmaschinen haben zwar nur endlich viele Zustände, aber ein **unendlich grosses Band** als „Speicher“.
- **Turingmaschinen akzeptieren genau die Typ-0-Sprachen.**
Dies gilt auch für **deterministische Turingmaschinen.**
- **Linear beschränkte Turingmaschinen akzeptieren genau die kontextsensitiven Sprachen.**
- Die kontextsensitiven und Typ-0-Sprachen sind unter **fast allen gängigen Operationen abgeschlossen.**
Ausnahme: **Typ-0 nicht unter Komplement abgeschlossen**
- Für kontextsensitive und Typ-0-Sprachen ist **fast kein Problem entscheidbar.**
Ausnahme: **Wortproblem für Typ-1 ist entscheidbar.**

Wo sind wir und wie geht es weiter?

Inhalte dieser Vorlesung

- Logik ✓
 - ▷ Wie kann man Wissen und Zusammenhänge repräsentieren und automatisiert verarbeiten?
- Automatentheorie und formale Sprachen
 - ▷ Was ist eine Berechnung?
- Berechenbarkeitstheorie
 - ▷ Was kann überhaupt berechnet werden?
- Komplexitätstheorie
 - ▷ Was kann effizient berechnet werden?

Wo sind wir und wie geht es weiter?

Inhalte dieser Vorlesung

- Logik ✓
 - ▷ Wie kann man Wissen und Zusammenhänge repräsentieren und automatisiert verarbeiten?
- Automatentheorie und formale Sprachen ✓
 - ▷ Was ist eine Berechnung?
- Berechenbarkeitstheorie
 - ▷ Was kann überhaupt berechnet werden?
- Komplexitätstheorie
 - ▷ Was kann effizient berechnet werden?