Beweisskizze

- Annahme: $HALTEPROBLEM$ ist berechenbar durch das $while$-Programm HALT1.
- Betrachte das Programm KONFUS:

$$begin\ HALT1;\ while\ X1 \neq 0\ do\ X1 := X1\ end$$

- Sei j der Index von KONFUS (d.h. KONFUS berechnet die Funktion SEM_j). Dann gilt
  $SEM_j(j) = \perp \iff HALTEPROBLEM(j) = 0 \iff$
  $SEM_j(j) \neq \perp$.(Widerspruch)

Unlösbarkeit des allgemeinen Halteproblems

- Das Problem, ob ein Programm auf eine beliebige Eingabe hält, ist nicht lösbar, d.h.:

Die Funktion $HALTEPROBLEM2: \mathbb{N} \times \mathbb{N} \longrightarrow \mathbb{N}$ mit

$$HALTEPROBLEM2(i,j) = \begin{cases} 1 & \text{wenn } SEM_i(j) \\ & \text{definiert ist} \\ 0 & \text{sonst} \end{cases}$$

ist nicht berechenbar.

Beweisskizze

- Annahme: $HALTEPROBLEM2$ wird durch HALT2 berechnet.
- Dann berechnet das folgende Programm das spezielle Halteproblem:

$$begin\ X2 := X1;\ HALT2\ end$$

(Widerspruch, da das spezielle Halteproblem nicht berechenbar ist.)

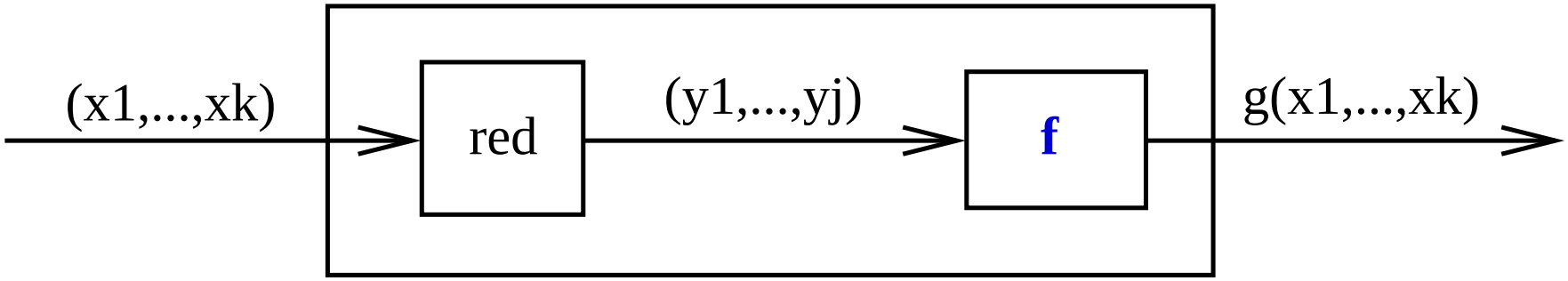
Beweise mittels Reduktion (Skizze)

Behauptung

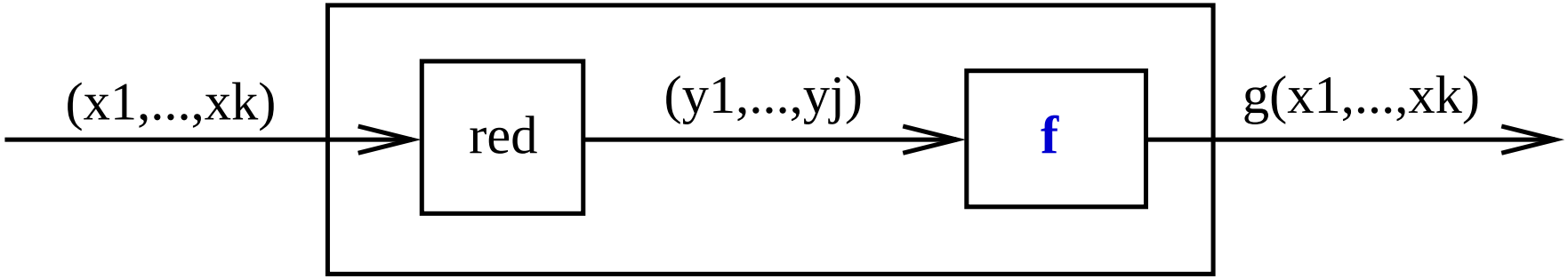
Die Funktion $f : \mathbb{N}^j \longrightarrow \mathbb{N}$ ist nicht berechenbar.

Beweisskizze

Annahme: f ist berechenbar.
Wähle eine nicht berechenbare Funktion g: $\mathbb{N}^k \longrightarrow \mathbb{N}$ und reduziere g auf f:



Gesucht ist somit eine totale berechenbare Funktion red: $\mathbb{N}^k \longrightarrow \mathbb{N}^j$, so dass $f(red(i)) = g(i)$ für alle $i \in \mathbb{N}^k$. (Ein bisschen gemogelt, da bei uns berechenbare Funktionen keine Tupel ausgeben können; ist aber leicht zu beheben.)



Wenn man g auf f reduziert, wird g berechenbar.
(Widerspruch, da g nicht berechenbar ist.)

Beispiel

Behauptung Die Funktion $HALTEPROBLEM2$ ist nicht berechenbar.

Beweis (Reduktion von $HALTEPROBLEM$ auf $HALTEPROBLEM2$)

Sei $red\colon \mathbb{N} \to \mathbb{N}^2$ definiert durch $red(i) = (i, i)$.

Dann ist red berechenbar durch

$$begin\ X2 := X1\ end$$

und es gilt

$$HALTEPROBLEM2(red(i)) =$$
$$HALTEPROBLEM2(i, i) =$$
$$HALTEPROBLEM(i)$$

Weitere unlösbare Probleme

Das Korrektheitsproblem

Sei f eine berechenbare Funktion.

- Eingabe: Index i eines $while$-Programms.
- Ausgabe: 1, falls $SEM_i = f$.
  0 sonst.

Das Korrektheitsproblem stellt die Frage, ob ein Programm eine bestimmte Funktion berechnet; es ist nicht lösbar.

Das Äquivalenzproblem

- Eingabe: $i, j \in \mathbb{N}$
- Ausgabe: 1, falls $SEM_i = SEM_j$.
  0, sonst.

Das Äquivalenzproblem stellt die Frage, ob 2 Programme dieselbe Semantik haben; es ist nicht lösbar.

Der Satz von Rice

Sei M die Menge der berechenbaren Funktionen.
Sei $E \subset M$ mit $\emptyset \neq E$.
Das Problem, ob ein Programm eine Funktion aus E berechnet, ist nicht lösbar, d.h.:

Die Funktion $check_E : \mathbb{N} \longrightarrow \mathbb{N}$ mit

$$check_E(i) = \begin{cases} 1 & \text{falls } SEM_i \in E \\ 0 & \text{sonst} \end{cases}$$

ist nicht berechenbar.

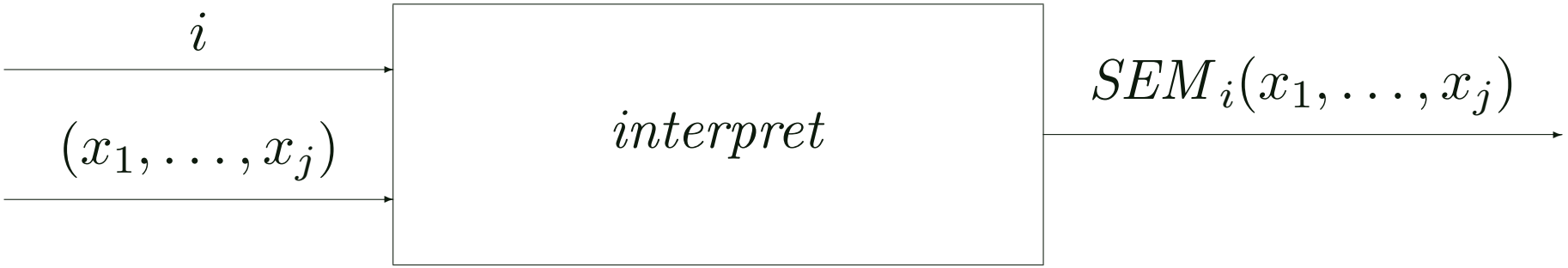
Beispiele

- $E = \{f \in M \mid f(n) = 3 \text{ für alle } n \in \mathbb{N}\}$
  ($check_E(i) = 1$ gdw. P_i immer den Wert 3 ausgibt.)
- $E = \{f \in M \mid f(n) = n \text{ für alle } n \in \mathbb{N}\}$
  ($check_E(i) = 1$ gdw. P_i immer die Eingabe ausgibt.)
- $E = \{f \in M \mid f \text{ ist total}\}$
  ($check_E(i) = 1$ gdw. P_i für alle Eingaben anhält.)
- $E = \{f \in M \mid f(n) \in \mathbb{N}\}$ mit $n \in \mathbb{N}$
  ($check_E(i) = 1$ gdw. P_i für die Eingabe n anhält.)
- $E = \{f \in M \mid f(n) = g(n) \text{ für alle } n \in \mathbb{N}\}$ mit $g \in M$
  ($check_E$ ist das Korrektheitsproblem.)

Existenz eines universellen *while*-Programms — ein lösbares Problem

Universelle Funktion

- Eingabe:
  - Index i eines $while$-Programms
  - $(x_1, \ldots, x_j) \in \mathbb{N}^j$
- Ausgabe: $SEM_i(x_1, \ldots, x_j)$.



Berechenbarkeit der universellen Funktion

Theorem

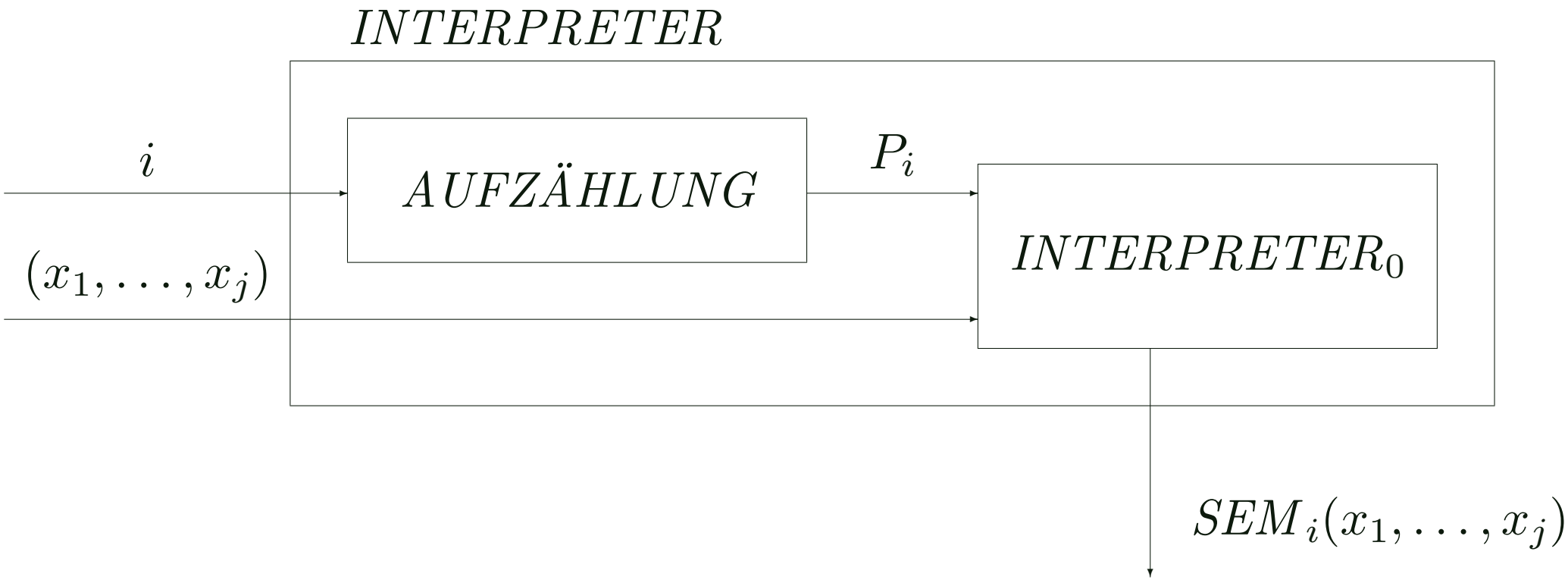
Die partielle Funktion $interpret\colon \mathbb{N}^{j+1} \to \mathbb{N}$, die für alle $j \in \mathbb{N}$ definiert ist durch

$$interpret(i, x_1, \ldots, x_j) = SEM_i(x_1, \ldots, x_j),$$

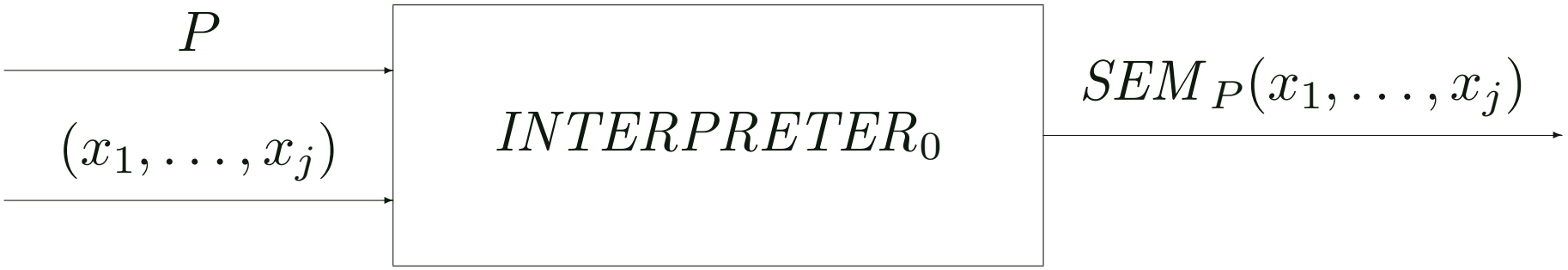
ist berechenbar.

Beachte: Gemäß der Churchschen These ist $interpret$ berechenbar, falls es einen Algorithmus für $interpret$ gibt.

PASCALchen-Interpreter als universelle Funktion



Algorithmus für $INTERPRETER_0$



1. Wandle das k-variable Programm P in sein Flussdiagramm um.
2. Wandle $(x_1, \ldots, x_j)$ in eine Eingabe a für P um.
3. Berechne P für Eingabe a.
4. Gib bei Termination den Wert von $X1$ aus.

Zusammenfassung und Ausblick

Zusammenfassung

Dieser Kurs beinhaltete folgende Themen:

1. Automatentheorie
2. Formale Sprachen
3. Berechenbarkeit

Endliche Automaten

- erkennen genau die regulären Sprachen
- Äquivalenz von nichtdeterministischen und deterministischen endlichen Automaten (Potenzautomat)
- korrekte Übersetzung in rechtslineare Grammatiken
- schnelle Worterkennung,
- werden in vielen Bereichen praktisch eingesetzt (Model-Checking, UML, XML-Parser, Kontrollbedingungen usw.),
- können keine gängigen Programmiersprachen erkennen

Kellerautomaten

- erkennen genau die kontextfreien Sprachen
- deterministische Kellerautomaten können weniger Sprachen erkennen als nichtdeterministische
- können $while$-Programme erkennnen
- deterministische Kellerautomaten werden im Compilerbau eingesetzt
- können nicht alle algorithmisch beschreibbaren Sprachen erkennen

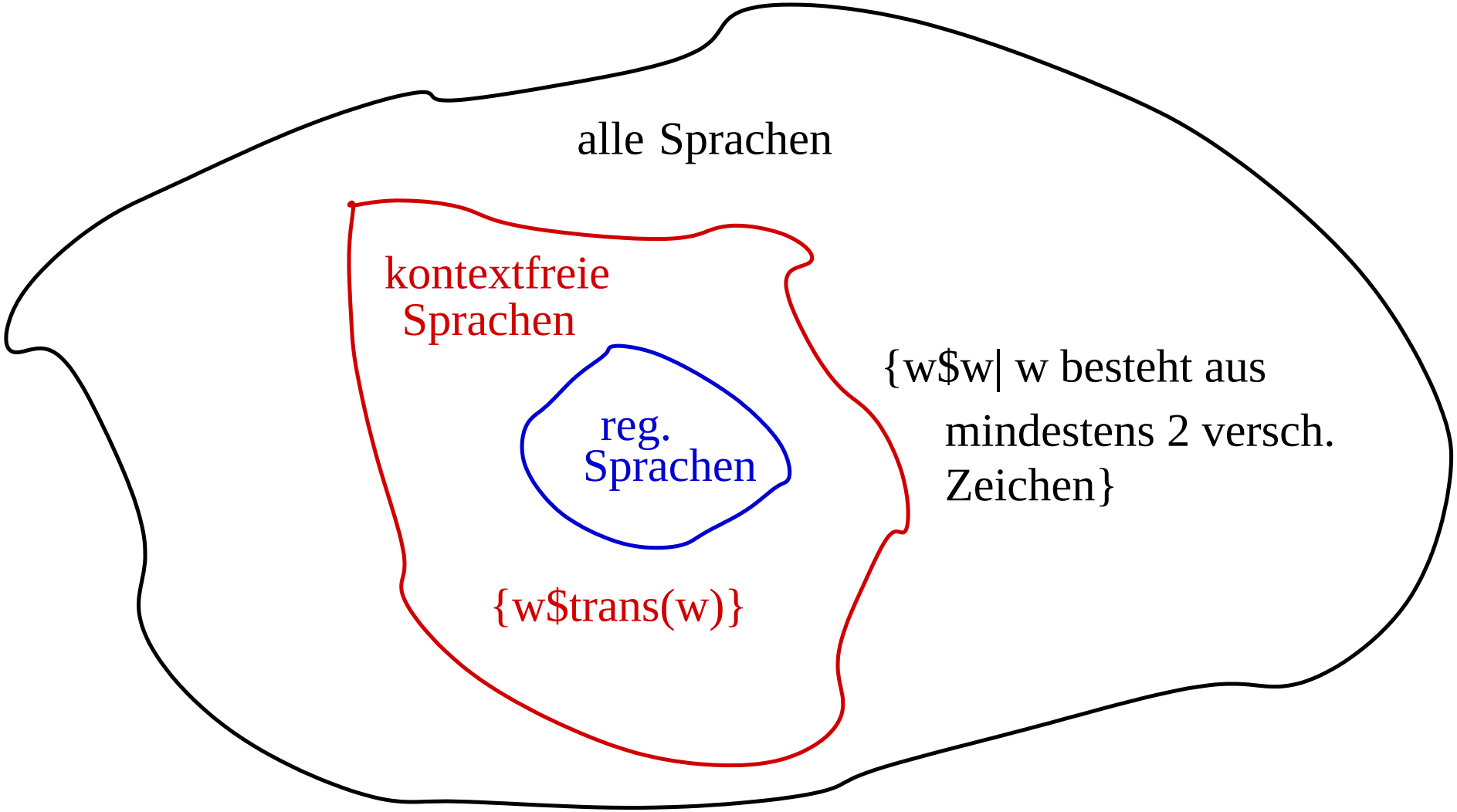
Reguläre Ausdrücke

- Beschreibungsmittel für die regulären Sprachen
- finden Anwendung im Compilerbau, Softwaretechnik, Webbrowsern usw.

Grammatiken

- kontextfreie Grammatiken
  - erzeugen genau die kontextfreien Sprachen
  - lange Ableitungen lassen sich in viele kleine zerlegen (Kontextfreiheitslemma)
  - korrekte Übersetzung in Kellerautomaten
- Rechtslineare Grammatiken
  - sind ein Spezialfall der kontextfreien Grammatiken
  - erzeugen genau die regulären Sprachen
  - erzeugen Wörter von links nach rechts

Sprachklassen



Reguläre Sprachen

- werden von endlichen Automaten erkannt
- werden von regulären Ausdrücken beschrieben
- werden von rechtslinearen Grammatiken erzeugt
- genügend lange Wörter sind pumpbar (Pumping-Lemma)
- Wortproblem entscheidbar
- Leerheitsproblem entscheidbar

Kontextfreie Sprachen

- werden von Kellerautomaten erkannt
- werden von kontextfreien Grammatiken erzeugt
- genügend lange Wörter sind pumpbar (Pumping-Lemma)

Abschlusseigenschaften regulärer und kontextfreier Sprachen

	Regulär	Kontextfrei
Vereinigung	+	+
Konkatenation	+	+
Kleene Hülle	+	+
Schnitt	+	-
Komplement	(+)	(-)

+, -: In diesem Kurs behandelt.

(+), (-): In diesem Kurs nicht behandelt.

Berechenbarkeit

- $while$-Progamme als Berechenbarkeitsmodell
- Churchsche These
- Effektive Aufzählbarkeit der $while$-Programme
- Unlösbarkeit des Halteproblems
- Reduktion
- Satz von Rice
- Existenz eines universellen $while$-Programms

Ausblick

Theoretische Informatik 2 umfasst folgende Themen:

- Formale Sprachen (Fortführung)
- Berechenbarkeit (Fortführung)
- Komplexitätstheorie

Formale Sprachen

- Welche allgemeineren Grammatiktypen gibt es, und wie hängen diese mit den bisher betrachteten zusammen?
- Für welche Sprachklassen ist das Wortproblem (noch) entscheidbar?
- Algorithmen für die Lösung des Wortproblems

Berechenbarkeit

- Weitere Berechenbarkeitsmodelle (Turingmaschinen, rekursive Funktionen)
- Zusammenhang zu $while$-Programmen

Komplexitätstheorie

- Wie lange dauert es mindestens und höchstens, bis ein Computer ein Problem berechnet hat, und wieviel Speicherplatz verbraucht das?
- Welche Berechnungszeiten sind (noch) akzeptabel?
- Wie komplex sind konkrete Algorithmen, wie z.B. Sortieren oder Matrizenmultiplikation?