# Theoretische Grundlagen der Informatik
# Tutorium XII & XIII (SR 301)
## Tut Nr. 6 – Üb5, Rekursionssatz

David Münch

Karlsruher Institut für Technologie
Institut für Informatik
IKS Müller-Quade

3. Dezember 2009

# Inhaltsverzeichnis

# Inhaltsverzeichnis

# Inhaltsverzeichnis

# Inhaltsverzeichnis

# Organisatorisches

Email: muenchdavid@gmail.com

`https://www.stud.uni-karlsruhe.de/~uhbro/`

Tutorium 12: Donnerstags 8:00 Uhr - Raum 301
Tutorium 13: Donnerstags 9:45 Uhr - Raum 301

Übungsblattabgabe Mittwochs 12:00 Uhr.

# Organisatorisches

Deckblatt benutzen: `http://www.stud.uni-karlsruhe.de/~unbdh/deckblatt/index.php?course=6`

Gruppenarbeit erwünscht, aber jeder muss handschriftliche Lösung mit Namen aller Gruppenteilnehmer abgeben.

50% der Punkte sind notwendig für den Schein.

# Organisatorisches

Nicht abgeholte Übungsblätter können ab sofort bei Nico Döttling R274 abgeholt werden.
`https://puck.iaks.uka.de/eiss/?id=167`

# Was wollen wir heute erreichen?

# Was wollen wir heute erreichen?

- Übungsblatt 5 besprechen.

# Was wollen wir heute erreichen?

- Übungsblatt 5 besprechen.
- Wiederholung von many-one-Reduktionen.

# Was wollen wir heute erreichen?

- Übungsblatt 5 besprechen.
- Wiederholung von many-one-Reduktionen.
- Rekursionssatz verstehen und anwenden können.

## Aufgabe 1

Analog zu Turingmaschinen lassen sich auch andere Maschinenmodelle Gödelisieren, beispielsweise endliche Automaten. Sei $\Sigma$ eine endliches Alphabet. Zeigen Sie:

a) Die Sprache

$$\text{INFINITE}_{DFA} = \{\langle \mathcal{M} \rangle | \ \mathcal{M} \text{ ist ein DFA und } \mathcal{L}(\mathcal{M}) \text{ ist unendlich } \}$$

ist entscheidbar.

b) Die Sprache

$$\text{ALL}_{DFA} = \{\langle \mathcal{M} \rangle | \ \mathcal{M} \text{ ist ein DFA und } \mathcal{L}(\mathcal{M}) = \Sigma^* \ \}$$

ist entscheidbar.

# Lösung 1a)

Ein Entscheider $\mathcal{T}$ für die Sprache INFINITE$_{DFA}$ könnte folgendermaßen aussehen. Die Eingabe für $\mathcal{T}$ sei $\langle \mathcal{M} \rangle$ mit $\mathcal{M} = (\mathcal{Q}, \Sigma, \delta, q_0, \mathcal{F})$.

- Ausgehend von allen Endzuständen $q \in \mathcal{F}$, druchlaufe den Zustandsübergangsgraphen in einer rückwärtigen Breitensuche und gib allen erreichten Zuständen eine Markierung $f$. Eine rückwärtige Breitensuche durchläuft alle Kanten rückwärts.

- Ausgehend vom Startzustand $q_0$ durchlaufe den Zustandsübergangsgraphen in Tiefensuche und markiere alle erreichten Zustände mit einer Markierung $s$. Sobald die Tiefensuche auf einen bereits mit $s$ markierten Zustand $q$ trifft, überprüfe ob $q$ auch eine Markierung $f$ trägt. Falls ja, dann akzeptiere, falls nein gehe die Kante über die $q$ erreicht wurde zurück ohne Markierung von $q$ zu entfernen und fahre mit der Tiefensuche fort. Beim Rückschreiten in der Tiefensuche entferne Markierungen $s$.
- Falls die Tiefensuche vollständig durchlaufen wurde, dann lehne die Eingabe ab.

Die Maschine $\mathcal{T}$ überprüft also ob es im Zustandsübergangsgraphen von $\mathcal{M}$ Zyklen gibt, welche Zustände enthalten, von denen aus ein Endzustand erreichbar ist. Falls ein solcher Zyklus gefunden wurde, dann läßt er sich (ähnlich wie im Beweis des Pumping-Lemmas) beliebig of aufpumpen. Die Maschine $\mathcal{T}$ ist ein Entscheider, den sowohl Breiten als auch Tiefensuche haben nach endlich vielen Schritten alle zyklenfreien Pfade durchlaufen.

# Lösung 1b)

Wir konstruieren einen Entscheider $\mathcal{T}$ der überprüft ob $\overline{\mathcal{L}(\mathcal{M})} = \emptyset$ ist, was äquivalent ist zu $\mathcal{L}(\mathcal{M}) = \Sigma^*$. Die Eingabe für $\mathcal{T}$ sei $\langle \mathcal{M} \rangle$ mit $\mathcal{M} = (\mathcal{Q}, \Sigma, \delta, q_0, \mathcal{F})$.

- Vertausche alle Finalzustände von $\mathcal{M}$ mit Nichtfinalzuständen. Wir setzen also $\mathcal{F}' = \mathcal{Q} \backslash \mathcal{F}$.
- Falls $\mathcal{F}' = \emptyset$ dann akzeptiere.
- Druchlaufe den Zustandsübergangsgraphen von $\mathcal{M}' = (\mathcal{Q}, \Sigma, \delta, q_0, \mathcal{F}')$ mit einer Breitensuche beginnend vom Startzustand $q_0$ aus und markiere alle erreichten Zustände. Falls im Laufe der Breitensuche ein Finalzustand $q \in \mathcal{F}'$ erreicht wird, dann lehne die Eingabe ab.
- Falls die Breitensuche abgeschlossen ist und kein Finalzustand $q \in \mathcal{F}'$ erreicht wurde dann akzeptiere.

$\mathcal{T}$ überprüft also ob $\mathcal{M}'$ Endzustände vom Startzustand aus erreichbar sind. Da $\mathcal{M}'$ genau die Komplementsprache $\overline{\mathcal{L}(\mathcal{M})}$ von $\mathcal{L}(\mathcal{M})$ erkennt, genügt es also zu entscheiden ob $\mathcal{M}'$ die leere Sprache erkennt, was genau der Fall ist wenn kein Endzustand vom Startzustand aus erreichbar ist. Da die Breitensuche immer nach endlich vielen Schritten abgeschlossen ist ist $\mathcal{T}$ ein Entscheider.

## Aufgabe 2

Zeigen Sie:

a) Die Sprache $\mathcal{L}_{rev} = \{\langle \mathcal{M} \rangle |$
Für alle Worte $w \in \Sigma^*$: TM $\mathcal{M}$ akzeptiert $w$
gdw. $\mathcal{M}$ akzeptiert $w^R$ $\}$
ist nicht entscheidbar.

b) Die Sprache

$$\mathcal{L}_C = \{\langle \mathcal{M}_1 \rangle \langle \mathcal{M}_2 \rangle |\ \mathcal{M}_1, \mathcal{M}_2 \text{ TMs, } \mathcal{L}(\mathcal{M}_1) = \overline{\mathcal{L}(\mathcal{M}_2)}\}$$

ist nicht entscheidbar.

# Lösung 2a)

Wir geben eine many-one Reduktion $f$ von $HALT$ auf $\mathcal{L}_{rev}$ an. Sei $\langle \mathcal{M} \rangle w$ eine Instanz von $HALT$. Wir definieren $f(\langle \mathcal{M} \rangle w) = \langle \mathcal{M}' \rangle$. Es gelte o.B.d.A. $\{0, 1\} \subseteq \Sigma$. Wir spezifizieren $\mathcal{M}'$ bei Eingabe $v$:

- Falls $v = 01$, akzeptiere
- Lösche die Eingabe vom Band
- Simuliere $\mathcal{M}$ bei Eingabe $w$
- Akzeptiere

Es ist klar: $\mathcal{M}'$ akzeptiert $01$ immer. Es ist weiter offensichtlich dass, falls $\mathcal{M}$ bei Eingabe $w$ hält die Maschine $\mathcal{M}'$ ganz $\Sigma^*$ erkennt. In diesem Fall gilt also für alle $w \in \Sigma^*$:
$w \in \mathcal{L}(\mathcal{M}') \Leftrightarrow w^R \in \mathcal{L}(\mathcal{M}')$. Falls $\mathcal{M}$ bei Eingabe $w$ nicht hält, erkennt $\mathcal{M}'$ die Sprache $\{01\}$, also ist in diesem Falle $(01)^R = 10 \notin \mathcal{L}(\mathcal{M}')$. Es gibt also ein $w \in \Sigma^*$ sodass $w \in \mathcal{L}(\mathcal{M}')$ und $w^R \notin \mathcal{L}(\mathcal{M}')$. Damit ist $f$ eine many-one Reduktion von $HALT$ auf $\mathcal{L}_{rev}$, womit $\mathcal{L}_{rev}$ nicht entscheidbar ist.

# Lösung 2b)

Wir geben eine many-one Reduktion $f$ von $HALT$ auf $\mathcal{L}_C$ an. Sei $\langle \mathcal{M} \rangle w$ eine Instanz von $HALT$. Wir definieren $f(\langle \mathcal{M} \rangle w) = \langle \mathcal{M}_1 \rangle \langle \mathcal{M}_2 \rangle$. Wir spezifizieren nun $\mathcal{M}_1$ bei Eingabe $v$

- Lösche die Eingabe vom Band.
- Simuliere $\mathcal{M}$ bei Eingabe $w$.
- Akzeptiere

Wir spezifizieren nun $\mathcal{M}_2$ bei Eingabe $v$

- Lehne ab

Es ist offensichtlich dass $\mathcal{M}_2$ die Sprache $\emptyset$ akzeptiert. Wenn $\mathcal{M}$ bei Eingabe $w$ hält akzeptiert $\mathcal{M}_1$ ganz $\Sigma^*$, ansonsten $\emptyset$. Damit sind die Sprachen $\mathcal{L}(\mathcal{M}_1)$ und $\mathcal{L}(\mathcal{M}_2)$ genau dann komplementär wenn $\mathcal{M}$ bei Eingabe $w$ hält. Somit ist $f$ eine many-one Reduktion von $HALT$ auf $\mathcal{L}_{EQ}$, womit $\mathcal{L}_{EQ}$ nicht entscheidbar ist.

## Definition: Many-One Reduzierbar

Eine Sprache $\mathcal{A}$ ist many-one reduzierbar auf eine Sprache $\mathcal{B}$, kurz $\mathcal{A} \leq_m \mathcal{B}$, falls eine berechenbare Funktion $f : \Sigma^* \to \Sigma^*$ existiert, sodass für alle $w \in \Sigma^*$

$$w \in \mathcal{A} \Leftrightarrow f(w) \in \mathcal{B}.$$

## Korollar

Ist $\mathcal{A} \leq_m \mathcal{B}$ und $\mathcal{A}$ nicht entscheidbar, dann ist auch $\mathcal{B}$ nicht entscheidbar.

## Definition: Halteproblem

$$HALT = \{\langle \mathcal{M} \rangle w \in \Sigma^* | \mathcal{M} \text{ hält bei Eingabe } w\}$$

## Lemma

Das Halteproblem $HALT$ ist nicht entscheidbar.

## Aufgabe

Zeigen Sie, dass die Sprache
$\mathcal{L} = \{\langle \mathcal{M} \rangle \mid \text{TM } \mathcal{M} \text{ hat min. einen unerreichbaren Zustand}\}$
nicht entscheidbar ist!

Es gilt: $\mathcal{L}$ entscheidbar $\Leftrightarrow$ $\bar{\mathcal{L}}$ entscheidbar

Es gilt: $\mathcal{L}$ entscheidbar $\Leftrightarrow$ $\bar{\mathcal{L}}$ entscheidbar

$\bar{\mathcal{L}} = \{\langle \mathcal{M} \rangle \mid \text{TM } \mathcal{M} \text{ hat keinen unerreichbaren Zustand}\}$

Es gilt: $\mathcal{L}$ entscheidbar $\Leftrightarrow$ $\bar{\mathcal{L}}$ entscheidbar
$\bar{\mathcal{L}} = \{\langle \mathcal{M} \rangle \mid \text{TM } \mathcal{M} \text{ hat keinen unerreichbaren Zustand}\}$
Zeige also die Reduktion HALT $\leq_m \bar{\mathcal{L}}$.

Es gilt: $\mathcal{L}$ entscheidbar $\Leftrightarrow$ $\bar{\mathcal{L}}$ entscheidbar
$\bar{\mathcal{L}} = \{\langle \mathcal{M} \rangle \mid \text{TM } \mathcal{M} \text{ hat keinen unerreichbaren Zustand}\}$
Zeige also die Reduktion HALT $\leq_m \bar{\mathcal{L}}$.
Konstruiere dazu aus einer Instanz $(\langle \mathcal{M} \rangle, w) \in$ HALT, also aus der Turingmaschine $\mathcal{M}$ und deren Eingabe $w$, eine neue Turingmaschine $\mathcal{M}'$:

Es gilt: $\mathcal{L}$ entscheidbar $\Leftrightarrow$ $\bar{\mathcal{L}}$ entscheidbar
$\bar{\mathcal{L}} = \{\langle \mathcal{M} \rangle \mid \text{TM } \mathcal{M} \text{ hat keinen unerreichbaren Zustand}\}$
Zeige also die Reduktion HALT $\leq_m \bar{\mathcal{L}}$.
Konstruiere dazu aus einer Instanz $(\langle \mathcal{M} \rangle, w) \in$ HALT, also aus der Turingmaschine $\mathcal{M}$ und deren Eingabe $w$, eine neue Turingmaschine $\mathcal{M}'$:
1. Leere das Band

Es gilt: $\mathcal{L}$ entscheidbar $\Leftrightarrow$ $\bar{\mathcal{L}}$ entscheidbar
$\bar{\mathcal{L}} = \{\langle \mathcal{M} \rangle \mid \text{TM } \mathcal{M} \text{ hat keinen unerreichbaren Zustand}\}$
Zeige also die Reduktion HALT $\leq_m \bar{\mathcal{L}}$.
Konstruiere dazu aus einer Instanz $(\langle \mathcal{M} \rangle, w) \in$ HALT, also aus der Turingmaschine $\mathcal{M}$ und deren Eingabe $w$, eine neue Turingmaschine $\mathcal{M}'$:
1. Leere das Band
2. Schreibe $w$ auf das Band

Es gilt: $\mathcal{L}$ entscheidbar $\Leftrightarrow$ $\bar{\mathcal{L}}$ entscheidbar
$\bar{\mathcal{L}} = \{\langle \mathcal{M} \rangle \mid \text{TM } \mathcal{M} \text{ hat keinen unerreichbaren Zustand}\}$
Zeige also die Reduktion HALT $\leq_m \bar{\mathcal{L}}$.
Konstruiere dazu aus einer Instanz $(\langle \mathcal{M} \rangle, w) \in$ HALT, also aus der Turingmaschine $\mathcal{M}$ und deren Eingabe $w$, eine neue Turingmaschine $\mathcal{M}'$:
1. Leere das Band
2. Schreibe $w$ auf das Band
3. Simuliere $\mathcal{M}$

Es gilt: $\mathcal{L}$ entscheidbar $\Leftrightarrow$ $\bar{\mathcal{L}}$ entscheidbar
$\bar{\mathcal{L}} = \{\langle \mathcal{M} \rangle \mid \text{TM } \mathcal{M} \text{ hat keinen unerreichbaren Zustand}\}$
Zeige also die Reduktion HALT $\leq_m \bar{\mathcal{L}}$.
Konstruiere dazu aus einer Instanz $(\langle \mathcal{M} \rangle, w) \in$ HALT, also aus der Turingmaschine $\mathcal{M}$ und deren Eingabe $w$, eine neue Turingmaschine $\mathcal{M}'$:

1. Leere das Band
2. Schreibe $w$ auf das Band
3. Simuliere $\mathcal{M}$
4. Gehe in einen speziellen Zustand $q_S$

Es gilt: $\mathcal{L}$ entscheidbar $\Leftrightarrow$ $\bar{\mathcal{L}}$ entscheidbar
$\bar{\mathcal{L}} = \{\langle \mathcal{M} \rangle \mid \text{TM } \mathcal{M} \text{ hat keinen unerreichbaren Zustand}\}$
Zeige also die Reduktion HALT $\leq_m \bar{\mathcal{L}}$.
Konstruiere dazu aus einer Instanz $(\langle \mathcal{M} \rangle, w) \in$ HALT, also aus der Turingmaschine $\mathcal{M}$ und deren Eingabe $w$, eine neue Turingmaschine $\mathcal{M}'$:
1. Leere das Band
2. Schreibe $w$ auf das Band
3. Simuliere $\mathcal{M}$
4. Gehe in einen speziellen Zustand $q_S$

Dabei hat $\mathcal{M}'$ bezüglich der Schritte 1. bis 3. keine unerreichbaren Zustände, der einzige potentiell unerreichbare Zustand ist also $q_S$.

Sei nun $f : \text{HALT} \to \bar{\mathcal{L}}, (\langle \mathcal{M} \rangle, w) \mapsto \langle \mathcal{M}' \rangle$ die totale und berechenbare Funktion, die die Reduktion nach der obigen Beschreibung liefert.

Sei nun $f : \text{HALT} \to \bar{\mathcal{L}}, (\langle \mathcal{M} \rangle, w) \mapsto \langle \mathcal{M}' \rangle$ die totale und berechenbare Funktion, die die Reduktion nach der obigen Beschreibung liefert. Dann gilt:

$(\langle \mathcal{M} \rangle, w) \in \text{HALT} \Leftrightarrow \mathcal{M}$ hält bei der Eingabe von $w \Leftrightarrow$ $\mathcal{M}'$ erreicht den Zustand $q_S \Leftrightarrow$

Sei nun $f : \text{HALT} \to \bar{\mathcal{L}}, (\langle \mathcal{M} \rangle, w) \mapsto \langle \mathcal{M}' \rangle$ die totale und berechenbare Funktion, die die Reduktion nach der obigen Beschreibung liefert. Dann gilt:

$(\langle \mathcal{M} \rangle, w) \in \text{HALT} \Leftrightarrow \mathcal{M}$ hält bei der Eingabe von $w \Leftrightarrow$
$\mathcal{M}'$ erreicht den Zustand $q_S \Leftrightarrow$
$\mathcal{M}'$ hat keinen unerreichbaren Zustand $\Leftrightarrow \langle \mathcal{M}' \rangle = f((\langle \mathcal{M} \rangle, w)) \in \bar{\mathcal{L}}$

Sei nun $f : \text{HALT} \to \bar{\mathcal{L}}, (\langle \mathcal{M} \rangle, w) \mapsto \langle \mathcal{M}' \rangle$ die totale und berechenbare Funktion, die die Reduktion nach der obigen Beschreibung liefert. Dann gilt:

$(\langle \mathcal{M} \rangle, w) \in \text{HALT} \Leftrightarrow \mathcal{M}$ hält bei der Eingabe von $w \Leftrightarrow$
$\mathcal{M}'$ erreicht den Zustand $q_S \Leftrightarrow$
$\mathcal{M}'$ hat keinen unerreichbaren Zustand $\Leftrightarrow \langle \mathcal{M}' \rangle = f((\langle \mathcal{M} \rangle, w)) \in \bar{\mathcal{L}}$

Damit ergibt sich aus der Annahme, dass $\bar{\mathcal{L}}$ entscheidbar ist, direkt, dass auch HALT entscheidbar ist. Dies ist aber ein Widerspruch, da HALT als nicht entscheidbar bekannt ist. Damit kann also $\bar{\mathcal{L}}$ und damit auch $\mathcal{L}$ nicht entscheidbar sein.

## Rekursionssatz

Es sei $\mathcal{T}$ eine TM, die die Funktion $t : \Sigma^* \times \Sigma^* \to \Sigma^*$ berechnet, dann existiert eine TM $\mathcal{R}$, die die Funktion $r : \Sigma^* \to \Sigma^*$ für alle $w$ berechnet:
$r(w) = t(\langle \mathcal{R} \rangle, w)$

## Rekursionssatz

Es sei $\mathcal{T}$ eine TM, die die Funktion $t : \Sigma^* \times \Sigma^* \to \Sigma^*$ berechnet, dann existiert eine TM $\mathcal{R}$, die die Funktion $r : \Sigma^* \to \Sigma^*$ für alle $w$ berechnet:
$r(w) = t(\langle \mathcal{R} \rangle, w)$

Interpretation:
Für die Berechnung einer Funktion kann man eine TM finden, welche ihre eigene Codierung mitverwenden kann.

## Rekursionssatz

Es sei $t : \Sigma^* \to \Sigma^*$ eine beliebig berechenbare Funktion, dann gibt es eine TM $\mathcal{F}$, für welche: $\langle \mathcal{G} \rangle = t(\langle \mathcal{F} \rangle)$ äquivalent $\langle \mathcal{F} \rangle$.

## Rekursionssatz

Es sei $t : \Sigma^* \to \Sigma^*$ eine beliebig berechenbare Funktion, dann gibt es eine TM $\mathcal{F}$, für welche: $\langle \mathcal{G} \rangle = t(\langle \mathcal{F} \rangle)$ äquivalent $\langle \mathcal{F} \rangle$.

Interpretation:
Für jede Programmtransformation gibt es einen Fixpunkt.

## Programmumschreibefunktion

$t : \Sigma^* \to \Sigma^*$ heißt Programmumschreibefunktion oder Programmtransformation.

## Programmumschreibefunktion

$t : \Sigma^* \to \Sigma^*$ heißt Programmumschreibefunktion oder Programmtransformation.

**Idee**:

- $t$ erhält als Eingabe ein Programm/TM $\langle \mathcal{F} \rangle$ (codiert als Gödelnummer)
- $t$ wandelt dieses Programm/TM $\langle \mathcal{F} \rangle$ in ein Programm/TM $\langle \mathcal{G} \rangle$ ab
- $t$ gibt das Programm/TM $\langle \mathcal{G} \rangle$ zurück

## Programmumschreibefunktion

$t : \Sigma^* \to \Sigma^*$ heißt Programmumschreibefunktion oder Programmtransformation.

**Idee**:

- $t$ erhält als Eingabe ein Programm/TM $\langle \mathcal{F} \rangle$ (codiert als Gödelnummer)
- $t$ wandelt dieses Programm/TM $\langle \mathcal{F} \rangle$ in ein Programm/TM $\langle \mathcal{G} \rangle$ ab
- $t$ gibt das Programm/TM $\langle \mathcal{G} \rangle$ zurück

**Beispiel**: $t(n) = \text{“}x_1 = 1; \langle \mathcal{F} \rangle\text{”}$.

## Aufgabe

Beweisen Sie, dass es eine Gödelnummer $n = \langle \mathcal{M} \rangle \in \mathbb{N}_0$ zu einer Turingmaschine $\mathcal{M}$ gibt, die die Funktion $f_n(x) = (n + x)^2$ für alle $x \in \mathbb{N}_0$ berechnet!

**Lösung:**

Die Funktion $t : \mathbb{N}_0^* \times \mathbb{N}_0^*, (a, x) \mapsto (a + x)^2$ ist sicher berechenbar. Nach dem Rekursionssatz $\exists$ TM $\mathcal{F}$ mit $\langle \mathcal{F} \rangle = n$ mit $f(x) = t(\langle \mathcal{F} \rangle, x) = t(n, x) = (n + x)^2$. Wir wenden den Rekursionssatz auf folgende Turingmaschine $\mathcal{M}$ an, welche eine Eingabe $x$ erhält:

1. Hole eigene Beschreibung $n = \langle \mathcal{M} \rangle$
2. Berechne $y = n + x$
3. Berechne $z = y^2$
4. Gib $z$ aus

Sei $n = \langle \mathcal{M}' \rangle$ die Gödelnummer von $\mathcal{M}$. $\mathcal{M}$ berechnet also die Funktion $f_n(x) = (n + x)^2$.

# Reflexion

Was haben wir heute gelernt?

# Reflexion

Was haben wir heute gelernt?

# Reflexion

Was haben wir heute gelernt?

- many-one-Reduktion wiederholt

# Reflexion

Was haben wir heute gelernt?

- many-one-Reduktion wiederholt
- Rekursionssatz

# Reflexion

Was haben wir heute gelernt?

- many-one-Reduktion wiederholt
- Rekursionssatz
- Programmumschreibefunktionen

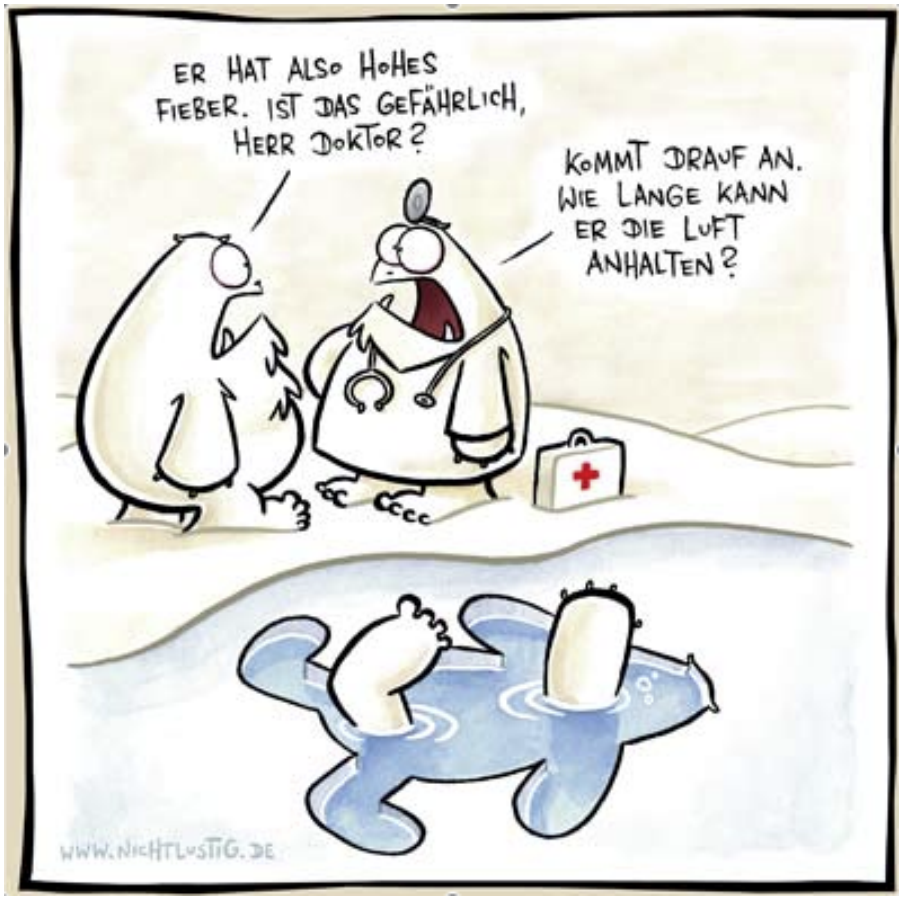
Noch Fragen?

# Vorschau

# Vorschau

- ?

# Bis zum nächsten Mal



## Aufgabe

Gegeben sei das folgende rekursive while-Programm:

procedure $R_1$
**if** $x_1 = 0$; **then** $x_1 := 1$;
**else if** $x_1 < 50$ **then** $x_1 := x_1 + 7$
**else** $x_1 := x_1 - 8; R_1; R_1; x_1 := x_1 - 1$;**end**
**end**

Berechne die zugehörige Funktionstransformationen und ihren kleinsten Fixpunkt. Welche Funktionen werden von den rekursiven Programmen berechnet?

**Lösung:**
Die Transformation $T_1$ ergibt sich zu

$$T_1(\varphi)(x) := \begin{cases} 1, & \text{falls } x = 0 \\ x + 7, & \text{falls } 0 < x < 50 \\ \varphi^2(x - 8) - 1, & \text{sonst} \end{cases}$$

und ihr kleinster Fixpunkt ist:

$$sup T_1^m(\bot)(x) := \begin{cases} 1, & \text{falls } x = 0 \\ x + 7, & \text{falls } 0 < x < 50 \\ 55, & \text{sonst} \end{cases}$$

Das Programm berechnet demnach die Funktion
$\varphi = sup T_1^m(\bot)(x)$.