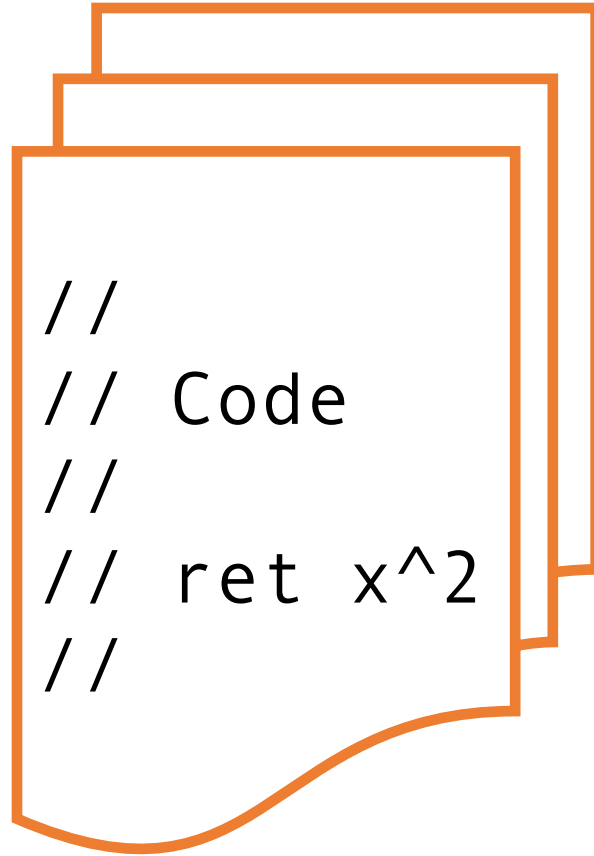


Satz von Rice

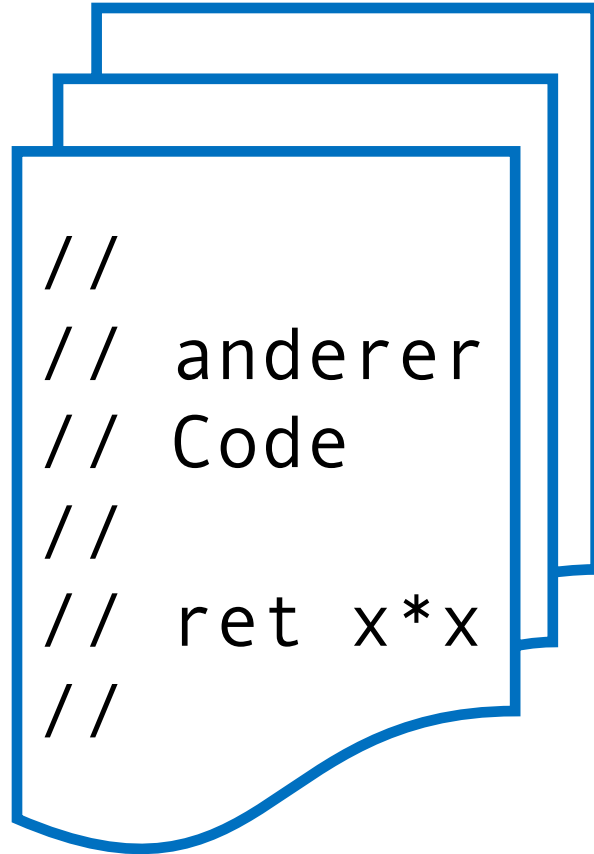
René Filip, TINF13B1

Motivation



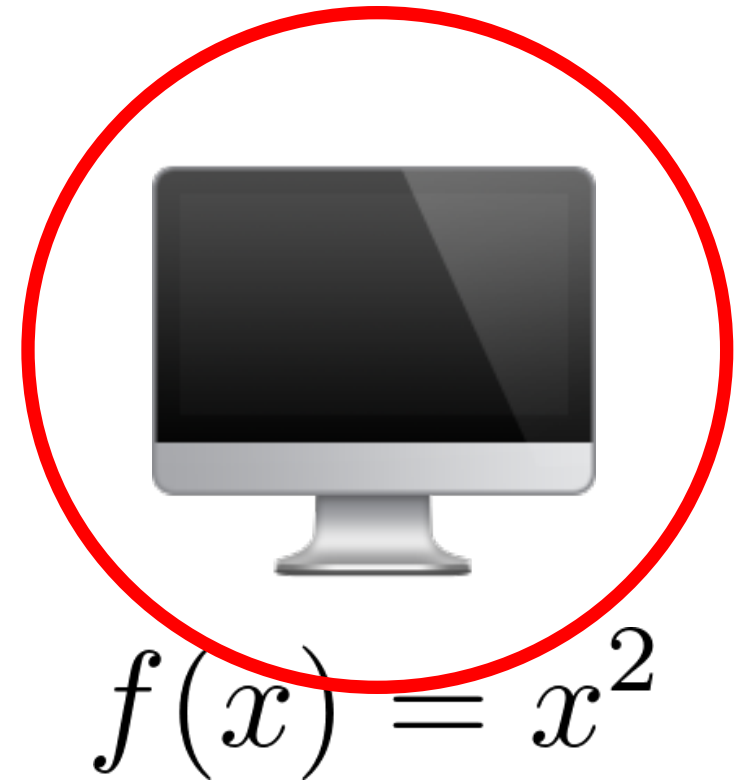
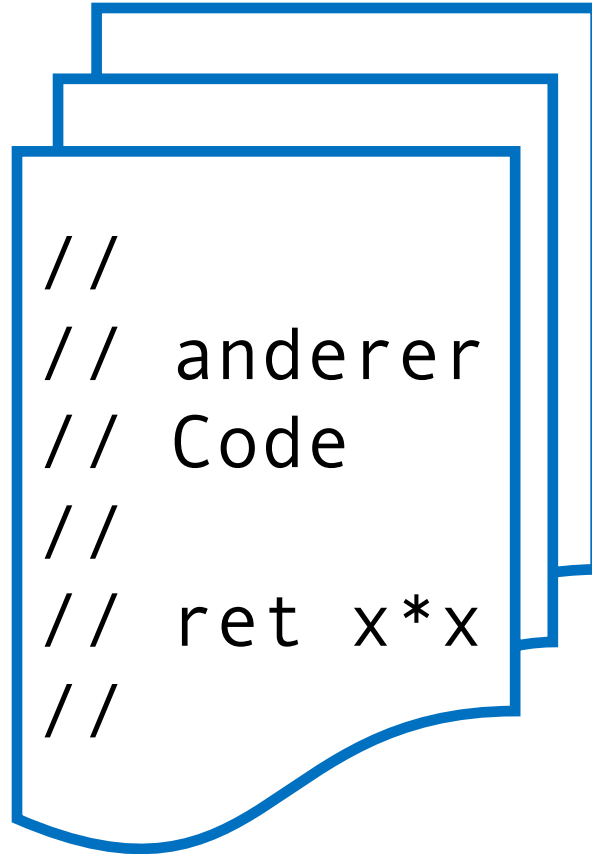
$$f(x) = x^2$$

Motivation

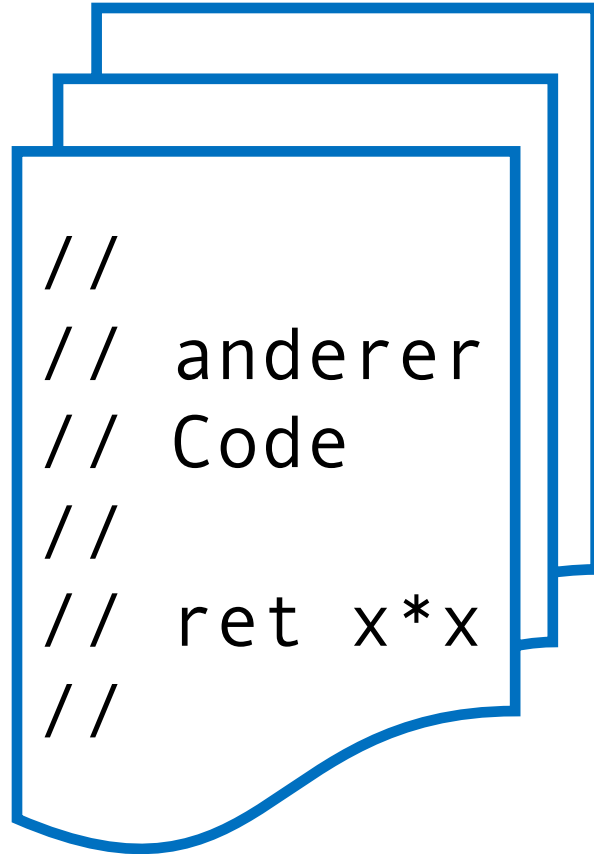


$$f(x) = x^2$$

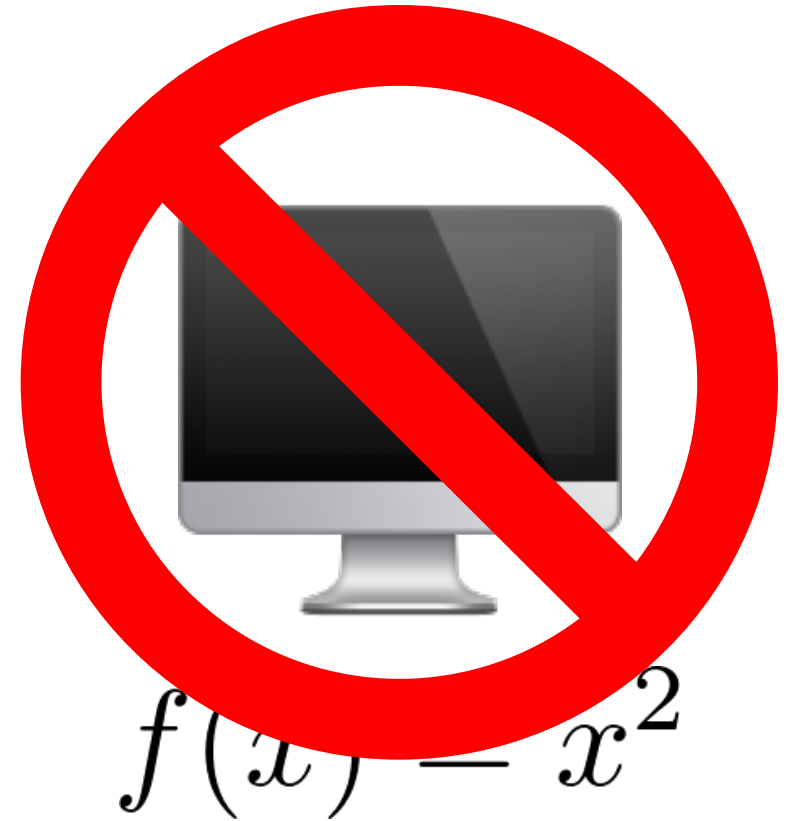
Motivation



Spoiler



Ja / Nein

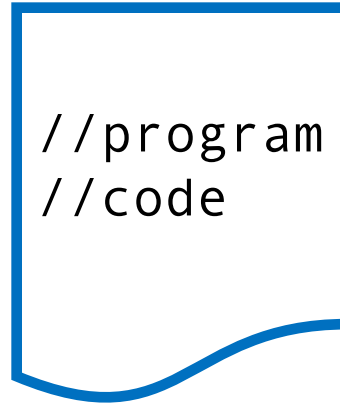


Unterschied zum Halteproblem



Halteproblem

“Hält mein Programm
irgendwan an?”

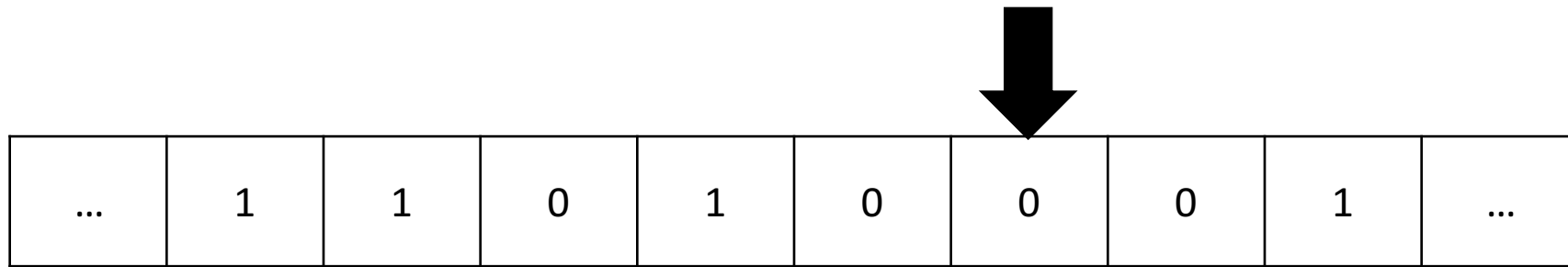


$$f(x) = x^2$$

Satz von Rice

“Berechnet mein
Programm die Funktion f ?”

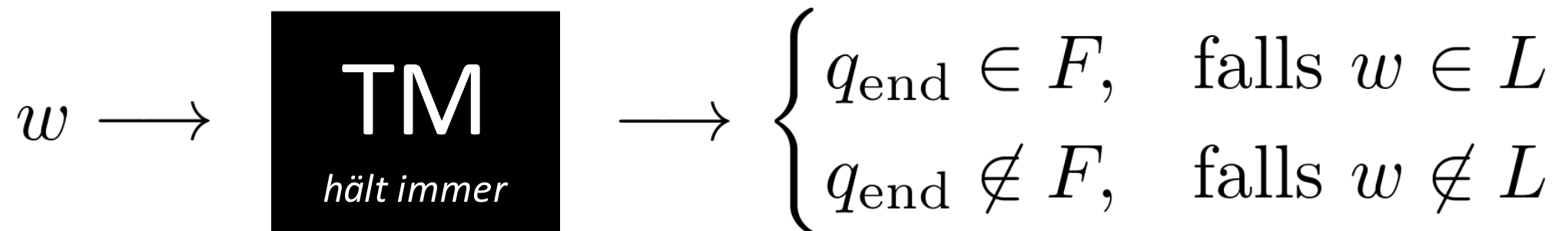
Definition Turingmaschine



$(Q, \Gamma, B, \Sigma, \delta, q_0, F)$

Entscheidbarkeit einer Sprache

2.1.2 Definition: Eine Sprache heißt rekursiv (entscheidbar), wenn es eine Turingmaschine gibt, die auf allen Eingaben stoppt und die Eingabe genau dann akzeptiert, wenn sie Element der Sprache ist.



existiert TM für Sprache $L \subseteq \Sigma^*$?

Gödelnummer

$$M = (\{q_1, \dots, q_t\}, \{0, 1, B\}, B, \{0, 1\}, \delta, q_1, \{q_2\})$$

0		X ₁	L		D ₁
1		X ₂	N		D ₂
B		X ₃	R		D ₃

$$\delta(q_i, X_j) = (q_k, X_l, D_m) \implies \text{code}(z) = 0^i 10^j 10^k 10^l 10^m$$

$$\langle M \rangle := 111 \text{code}(1)11 \text{code}(2)11 \dots 11 \text{code}(s)111$$

Universelle Turingmaschine

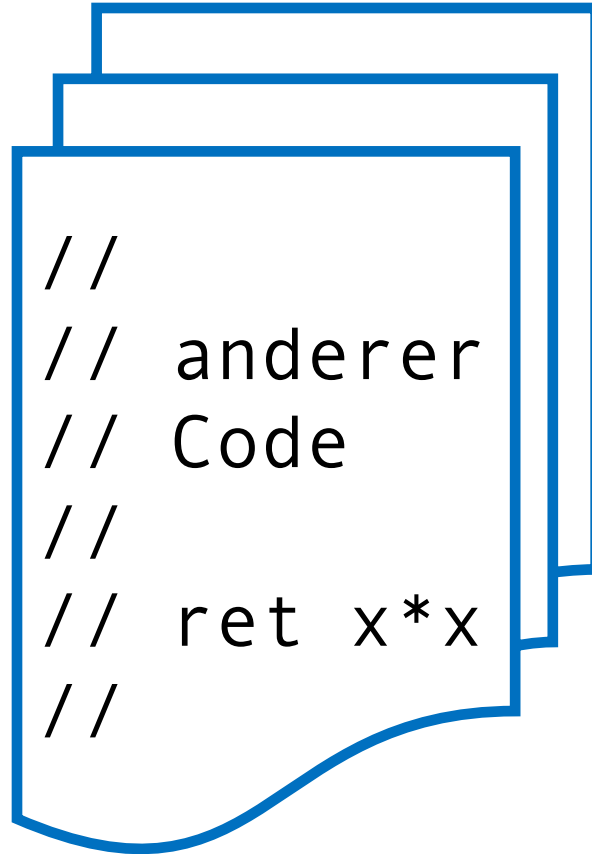
$$M = (\{q_1, \dots, q_t\}, \{0, 1, B\}, B, \{0, 1\}, \delta, q_1, \{q_2\})$$

$$\delta(q_i, X_j) = (q_k, X_l, D_m) \implies \text{code}(z) = 0^i 10^j 10^k 10^l 10^m$$

$$\langle M \rangle := 111 \text{code}(1)11 \text{code}(2)11 \dots 11 \text{code}(s)111$$

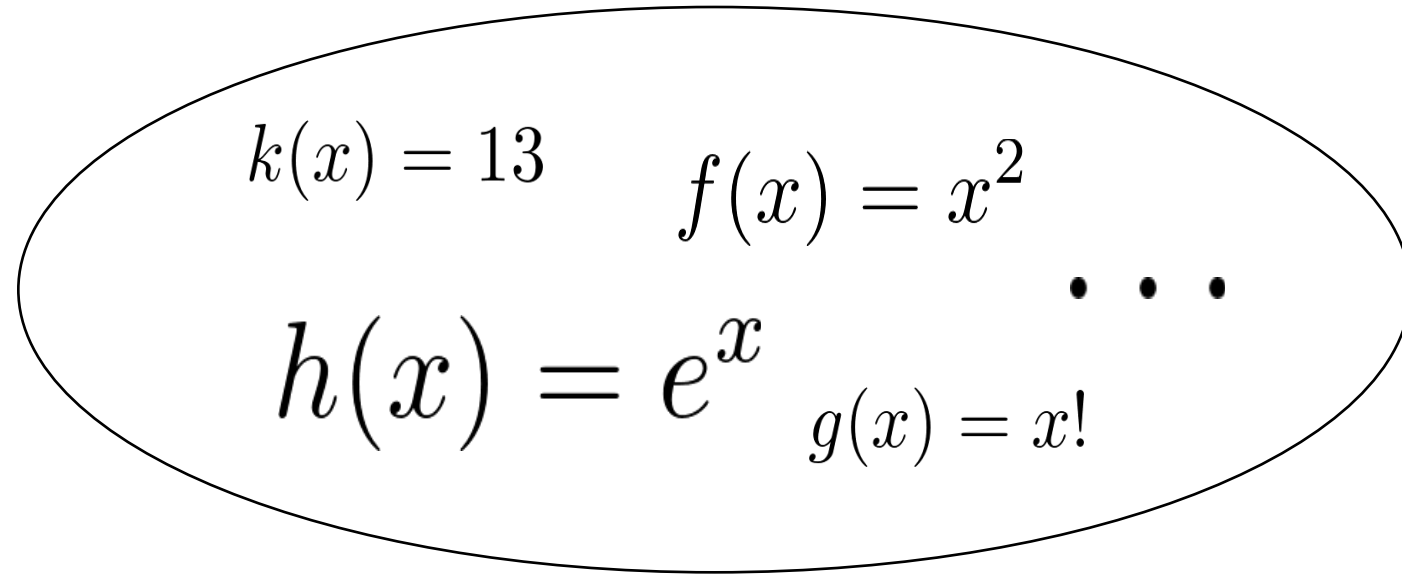
$$(\langle M \rangle, w)$$

Formalisierung



$$f(x) = x^2$$

Menge der von TM berechenbaren
Funktionen \mathcal{R}

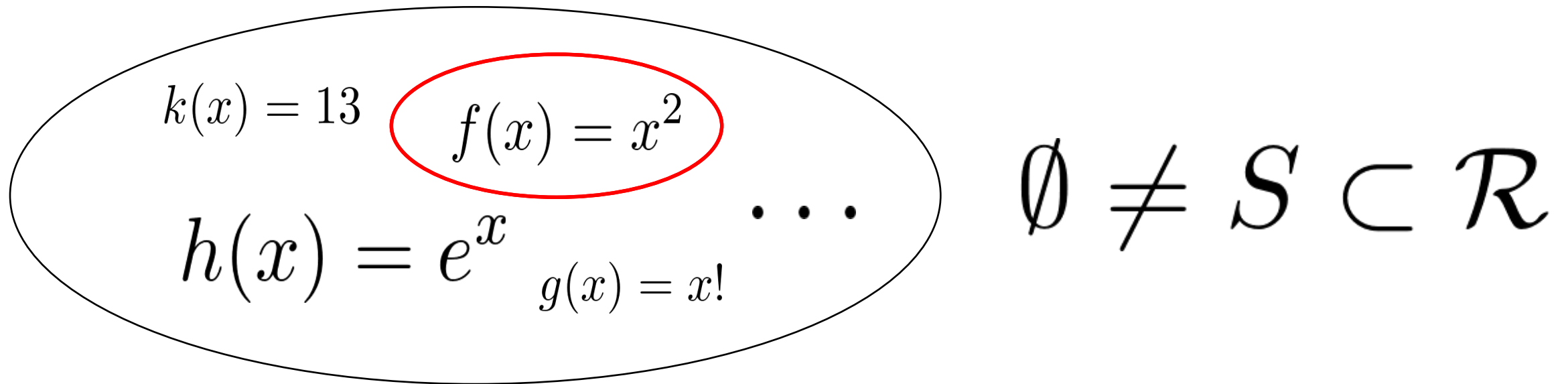


An oval containing the following mathematical expressions:

$$k(x) = 13 \quad f(x) = x^2 \quad \dots$$
$$h(x) = e^x \quad g(x) = x!$$

$$\mathcal{R} = \{x^2, 13, e^x, \dots\}$$

Nicht-triviale Teilmenge S


$$\begin{array}{l} k(x) = 13 \\ f(x) = x^2 \\ h(x) = e^x \\ g(x) = x! \end{array} \dots \quad \emptyset \neq S \subset \mathcal{R}$$

Beispiel: $S = \{f(x) = x^2\}$

Sprache $L(S)$

$$S = \{f(x) = x^2\}$$

$$L(S) := \{\langle M \rangle \mid M \text{ berechnet eine Funktion aus } S\}$$



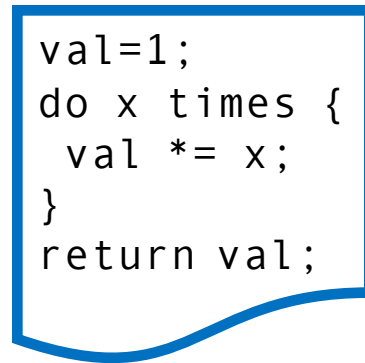
M_1

$\langle M_1 \rangle = 11100\dots$



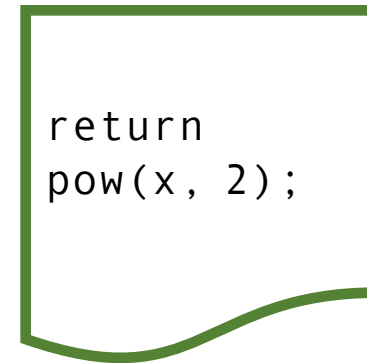
M_2

$\langle M_2 \rangle = 11101\dots$



M_3

$\langle M_3 \rangle = 11110\dots$



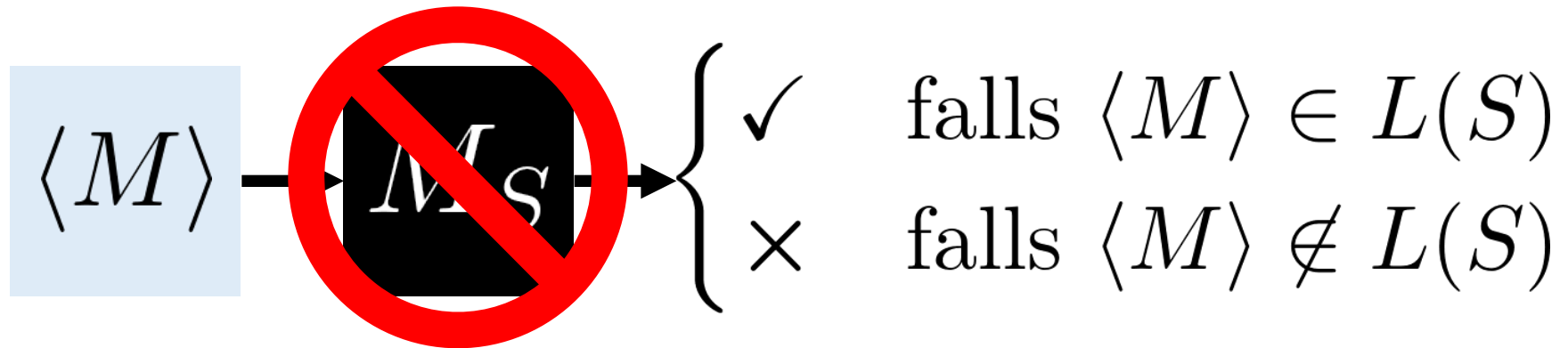
M_4

$\langle M_4 \rangle = 11111\dots$

...

Satz von Rice

$$L(S) := \{ \langle M \rangle \mid M \text{ berechnet eine Funktion aus } S \}$$



2.6.11 Satz von Rice: Die Sprache $L(S)$ ist nicht rekursiv.

Beweis

Zutaten:

- M_S : Variable $\langle M'' \rangle$; Parameter Annahme; Turingmaschine die $L(S)$ entscheidet
- $u \in \mathcal{R}$: Überall undefinierte Funktion
- M_f : Variable x ; berechnet $f \in \mathcal{R} - S$
- M' : Variable ε ; Parameter $\langle M \rangle$
 1. Berechne $\langle M'' \rangle$
- M'' : Variable x ; Parameter (M_f, M, ε)

Beweis-Strategie

2.6.2 Satz: D ist nicht rekursiv.

\Rightarrow **2.6.3 Korollar:** \overline{D} ist nicht rekursiv.

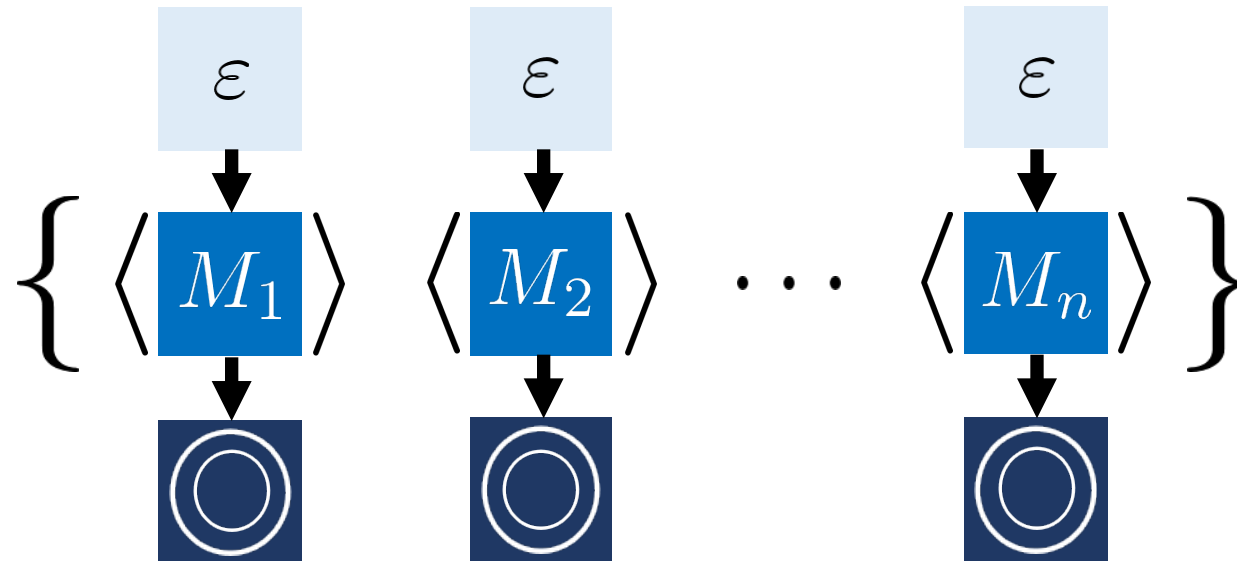
\Rightarrow **2.6.5 Satz:** H ist nicht rekursiv.

\Rightarrow **2.6.10 Satz:** H_ε ist nicht rekursiv.

\Rightarrow **2.6.11 Satz von Rice**

Das Spezielle Halteproblem

$$H_\varepsilon := \{ \langle M \rangle \mid M \text{ hält auf der Eingabe } \varepsilon \}$$

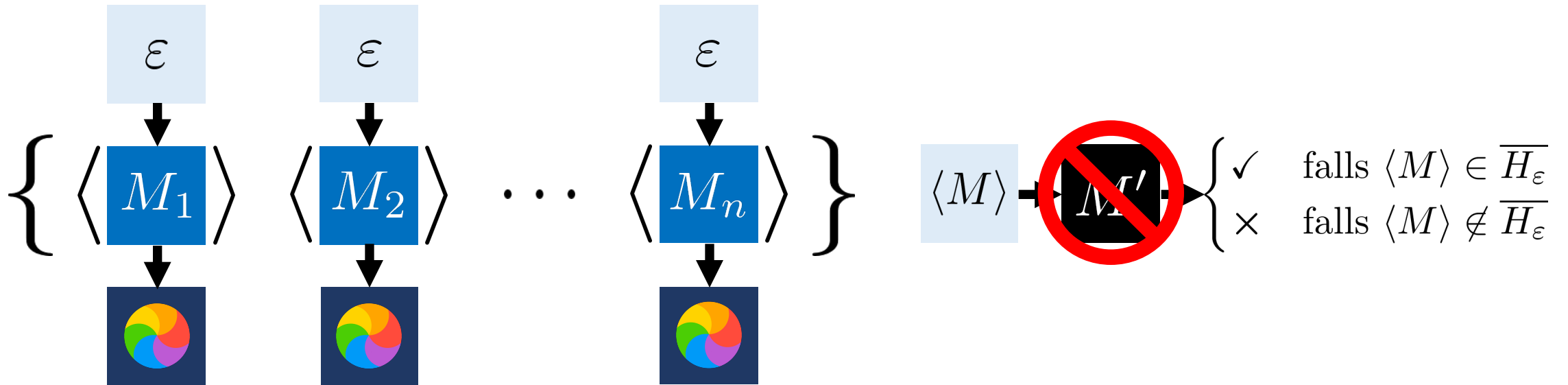


2.6.10 Satz: H_ε ist nicht rekursiv.

Beweis

Komplementär von H_ε

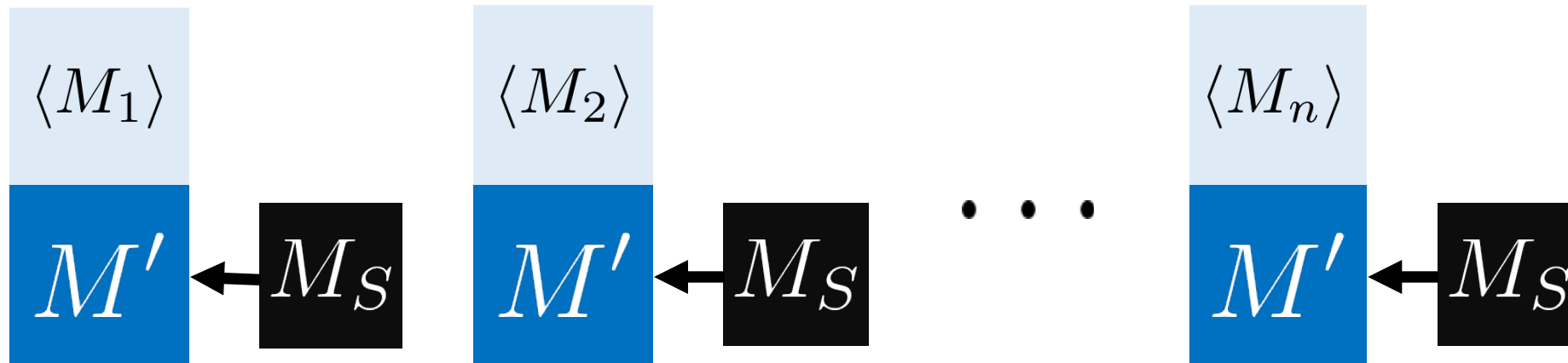
$$\overline{H_\varepsilon} := \{ \langle M \rangle \mid M \text{ hält nicht auf der Eingabe } \varepsilon \}$$



Satz 2.6.10 + Korollar 2.6.3: $\overline{H_\varepsilon}$ ist nicht rekursiv.

Lösungsidee

Annahme: M_S existiert und entscheidet $L(S)$



$\overline{H_\varepsilon} := \{ \langle M \rangle \mid M \text{ hält nicht auf der Eingabe } \varepsilon \}$

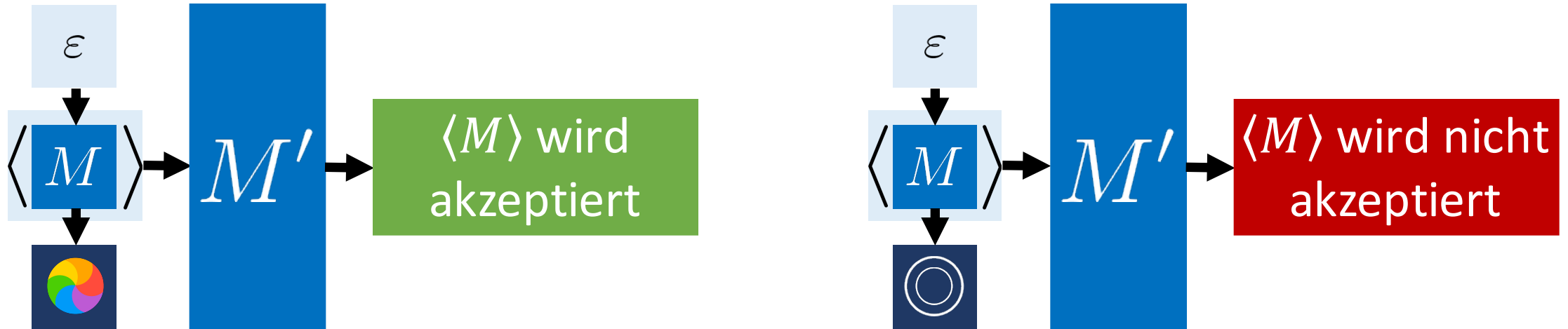
Satz 2.6.10 + Korollar 2.6.3: $\overline{H_\varepsilon}$ ist nicht rekursiv.

Ziel: M' ist eine stets haltende Turingmaschine, die $\overline{H_\varepsilon}$ akzeptiert.



Verhalten von M'

M' entscheidet, ob M_1, M_2, \dots auf die Eingabe ε hält

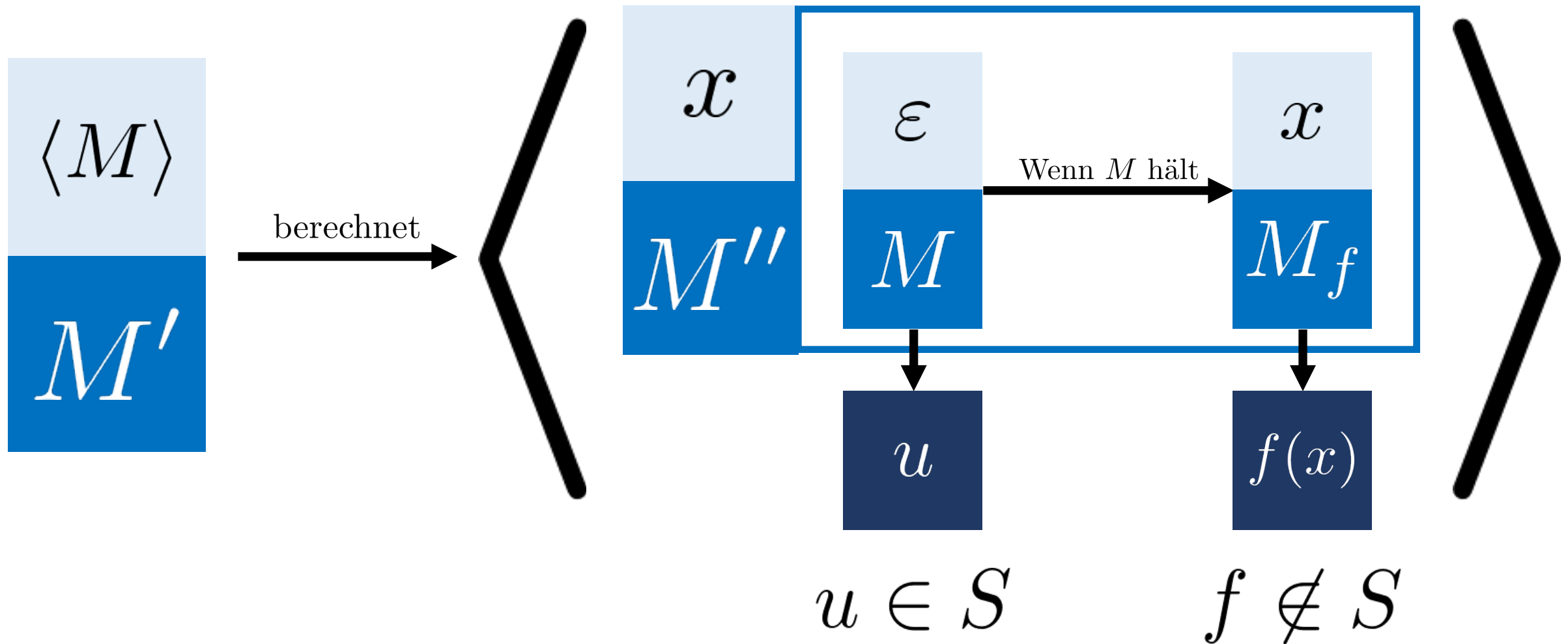


Wenn M auf ε nicht hält, dann muss M' das Programm $\langle M \rangle$ akzeptieren

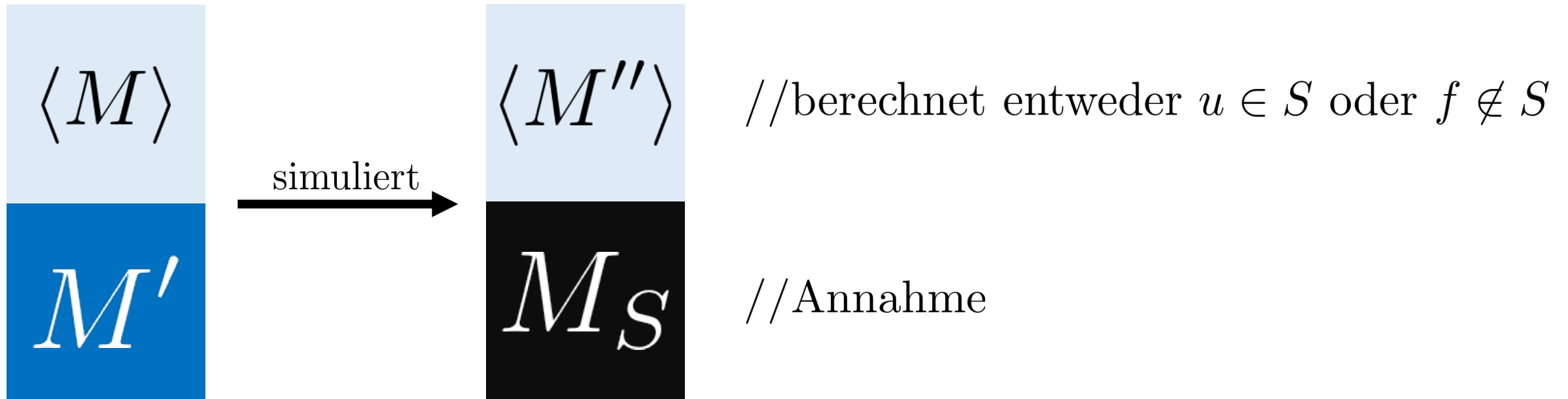
Wenn M auf ε hält, dann darf M' das Programm $\langle M \rangle$ nicht akzeptieren

Wähle f “geschickt”

1. Schritt: Berechnung von Programm $\langle M'' \rangle$



2. Schritt: Simulation von M_S auf $\langle M'' \rangle$



1. **Fall:** M hält nicht auf $\varepsilon \Rightarrow u \in S \Rightarrow M_S$ akzeptiert $\langle M'' \rangle \Rightarrow M'$ akzeptiert $\langle M \rangle$
2. **Fall:** M hält auf $\varepsilon \Rightarrow f \notin S \Rightarrow M_S$ akzeptiert nicht $\langle M'' \rangle \Rightarrow M'$ akzeptiert nicht $\langle M \rangle$
 $\Rightarrow M'$ stets haltend und akzeptiert $\overline{H_\varepsilon}$ ⚡

Hausaufgabe: Was ist mit $u \notin S$?

- Dann wähle ein $f \in S$ (erlaubt, da $S \neq \emptyset$)
 - Beweis von $\overline{L(S)}$, d.h. M_S entscheidet $\overline{L(S)}$
- ➔ Beweis sei dem aufmerksamen Studenten überlassen

Was ist mit $S = \mathcal{R}$?

- Vom Satz per Definition ausgeschlossen
- Von der Bedeutung: Gäbe es eine Code-Checker, mit $S = \mathcal{R}$, dann würde er uns nur sagen, dass unser Code *tatsächlich* eine (TM) berechenbare Funktion berechnet. Wir könnten aber nichts weitere eingrenzen, da jede Funktion aus \mathcal{R} in Frage kommt.