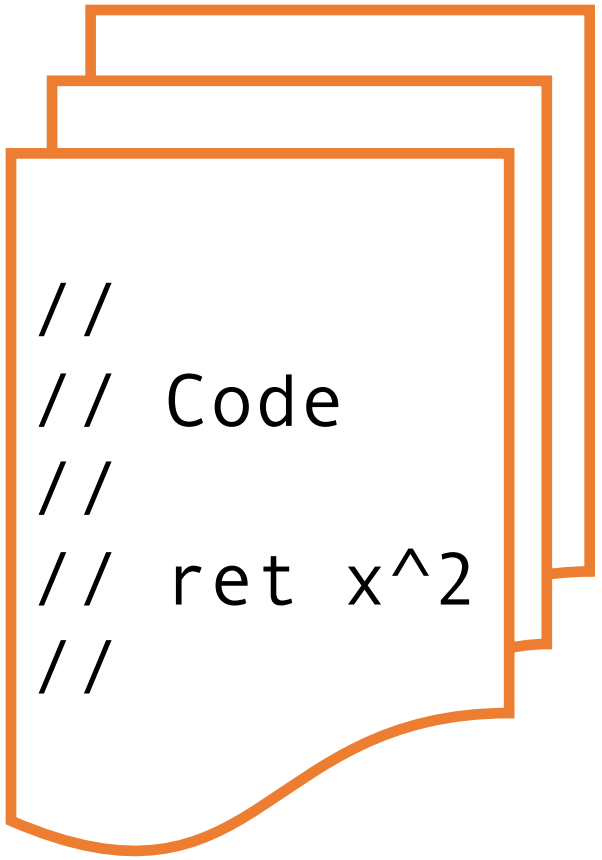
Satz von Rice

René Filip, TINF13B1

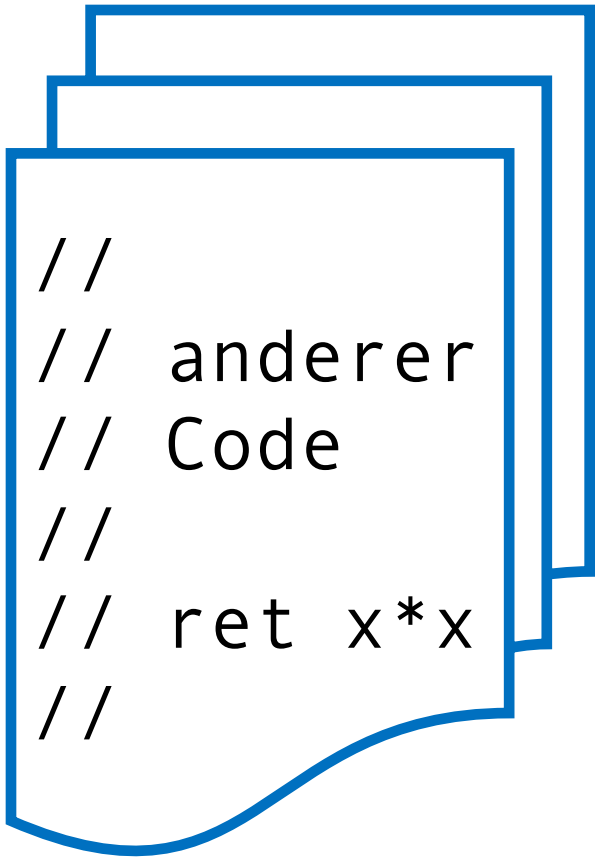
Motivation



Ja / Nein

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

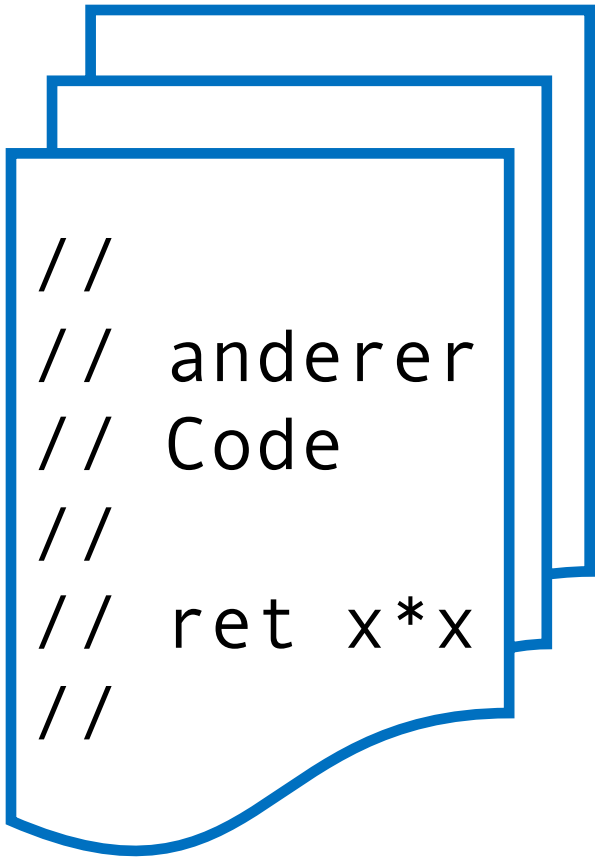
Motivation

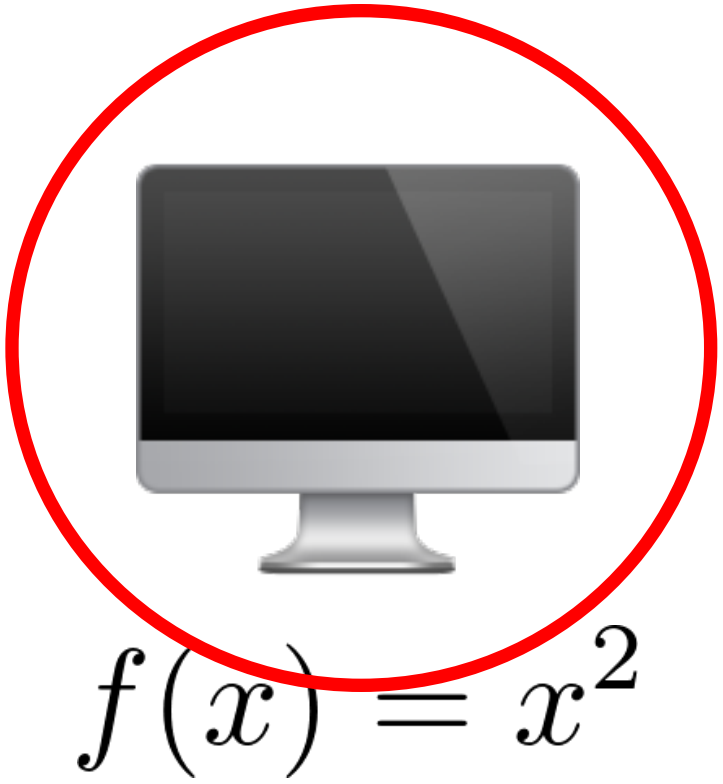


Ja / Nein

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

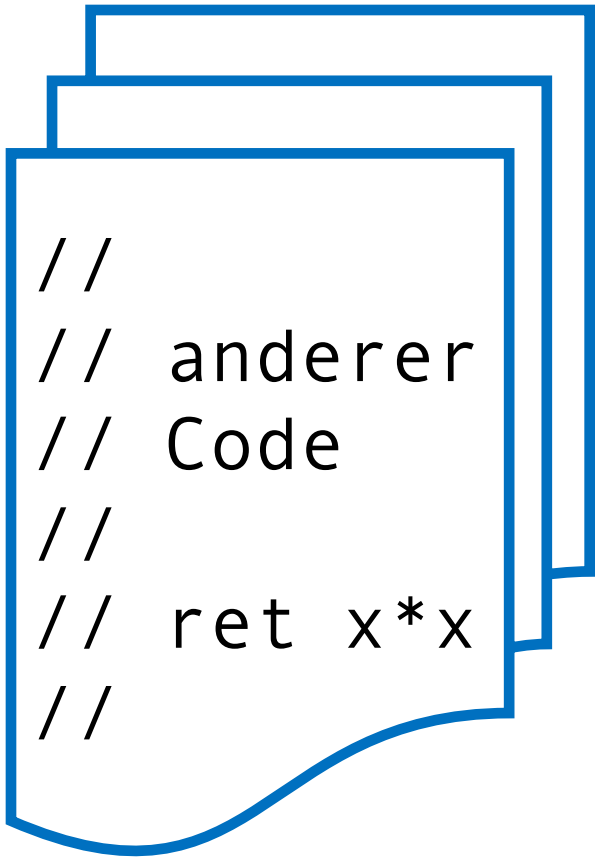
Motivation



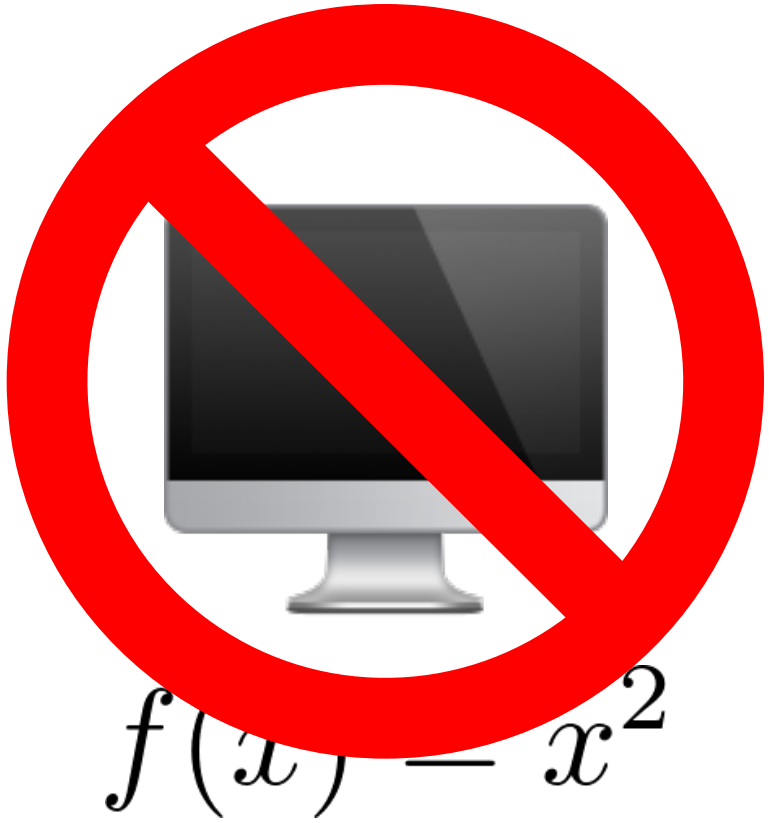


$$f(x) = x^2$$

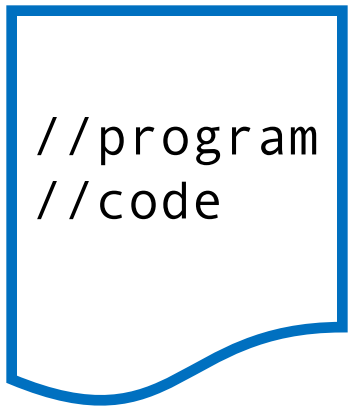
Spoiler



Ja / Nein



Unterschied zum Halteproblem



$f(x) = x^2$

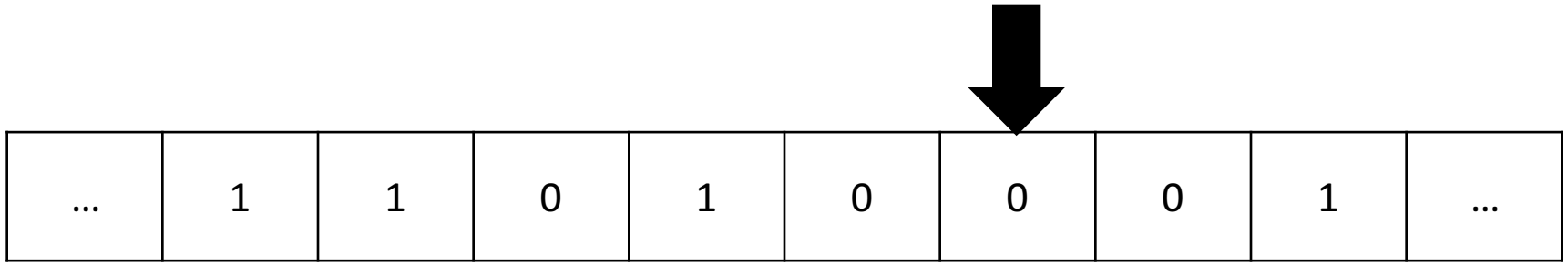
Halteproblem

“Hält mein Programm irgendwan an?”

Satz von Rice

”Berechnet mein Programm die Funktion f?”

Definition Turingmaschine



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

Gödelnummer

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

0	X_1	L	D_1
1	X_2	N	D_2
B	X_3	R	D_3

$$\delta(q_i, X_j) = (q_k, X_l, D_m) \Longrightarrow \mathrm{code}(z) = 0^i 1 0^j 1 0^k 1 0^l 1 0^m$$

$$\langle M \rangle := 111\,\mathrm{code}(1)11\,\mathrm{code}(2)11 \ldots 11\,\mathrm{code}(s)111$$

Universelle Turingmaschine

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

$$\delta(q_i, X_j) = (q_k, X_l, D_m) \Longrightarrow \text{code}(z) = 0^i 1 0^j 1 0^k 1 0^l 1 0^m$$

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

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

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.

$$w \longrightarrow \boxed{\text{TM}} \longrightarrow \begin{cases} q_{\text{end}} \in F, & \text{falls } w \in L \\ q_{\text{end}} \notin F, & \text{falls } w \notin L \end{cases}$$

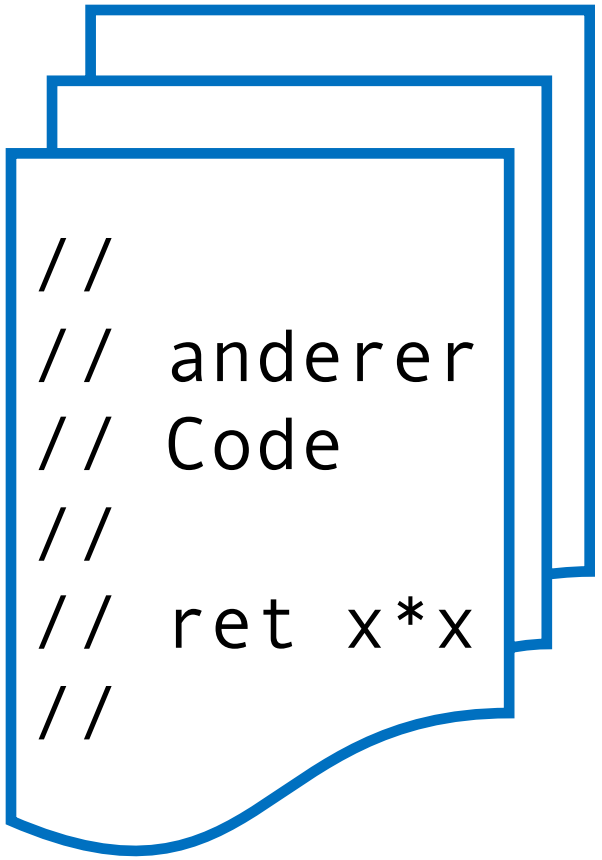
$$\text{existiert TM für Sprache } L \subseteq \Sigma^* ?$$

Konzepte

- Turingmaschine
- Gödelnummer und Universelle Turingmaschine
- Entscheidbarkeit bzw. Rekursive Sprachen
- Universelle Sprache
- Halteproblem
- Spezielles Halteproblem

➔ Satz von Rice

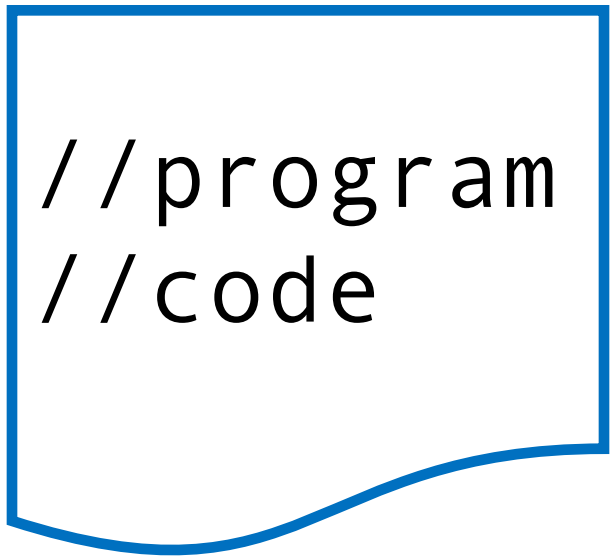
Formalisierung



Ja / Nein

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

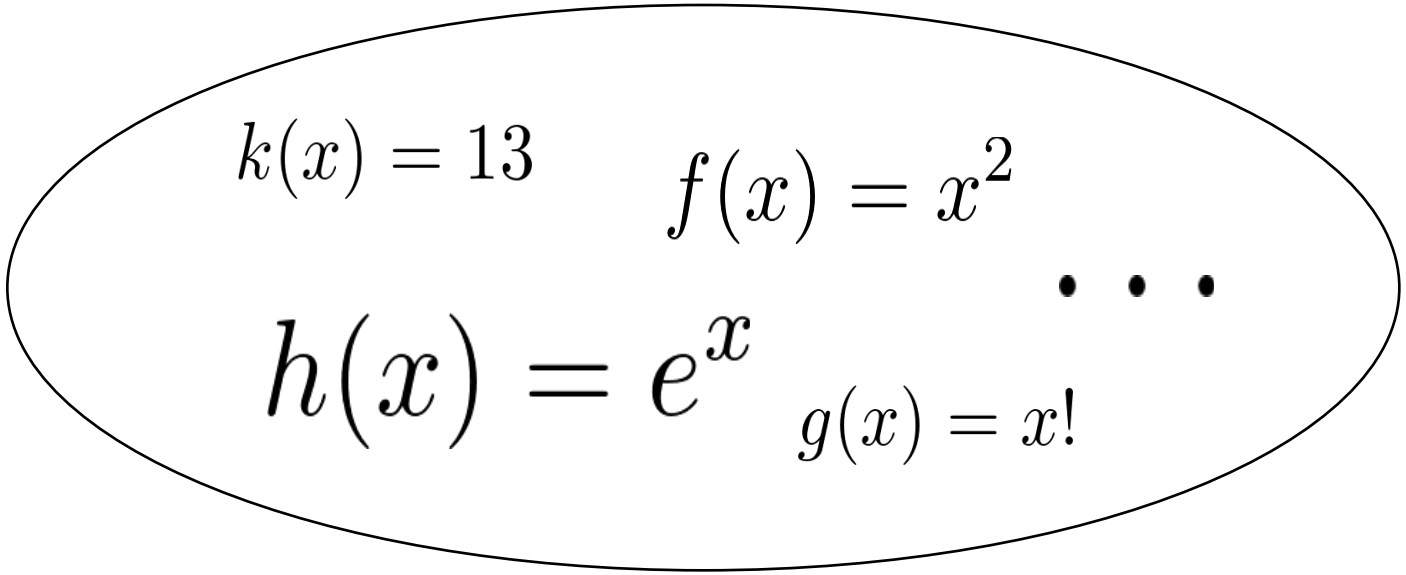
Formalisierung



M: Turing Maschine, äquivalent zum Programmcode

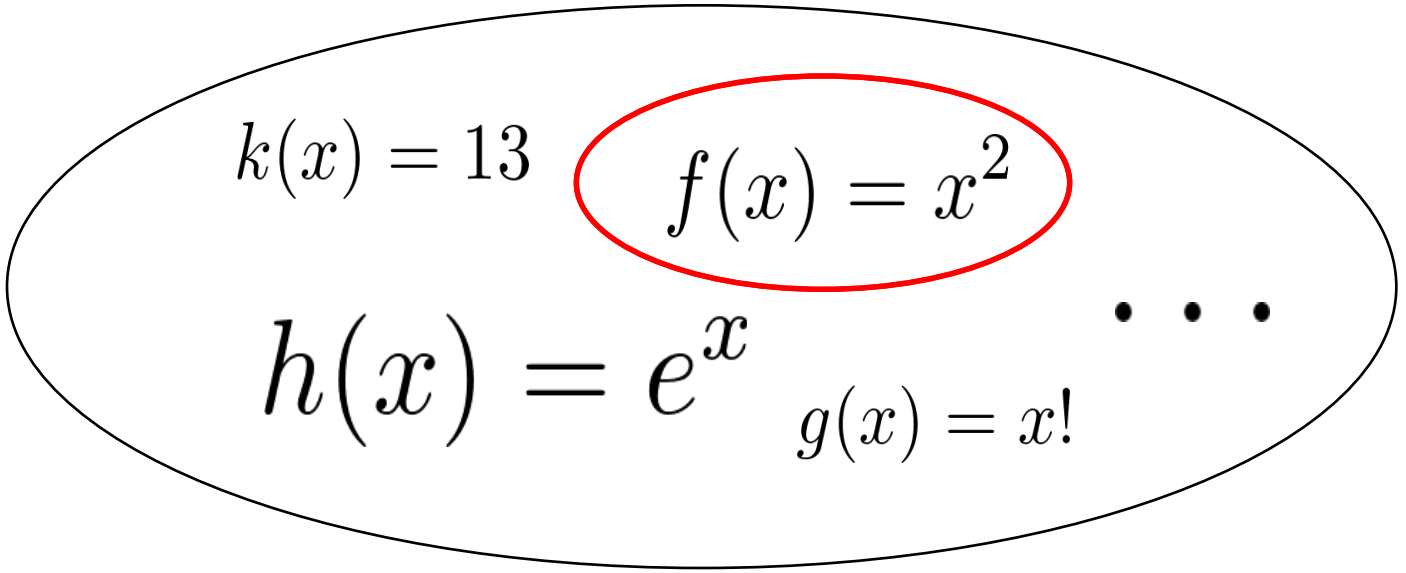
$$\langle M \rangle = 111\,\mathrm{code}(1)11\,\mathrm{code}(2)11\ldots11\,\mathrm{code}(s)111$$

Menge der von TM berechenbaren Funktionen $\mathcal{R}$



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

Nicht-triviale Teilmenge S

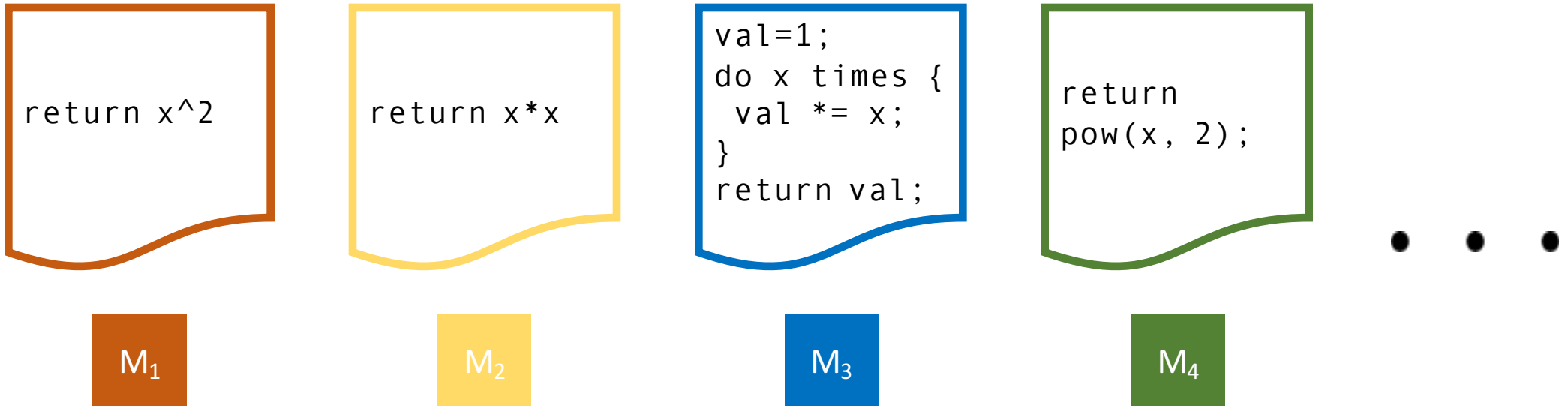


$$\emptyset \neq S \subset \mathcal{R}$$

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

Sprache $L(S)$

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



Formalisierung

Programmcode $\langle M \rangle$

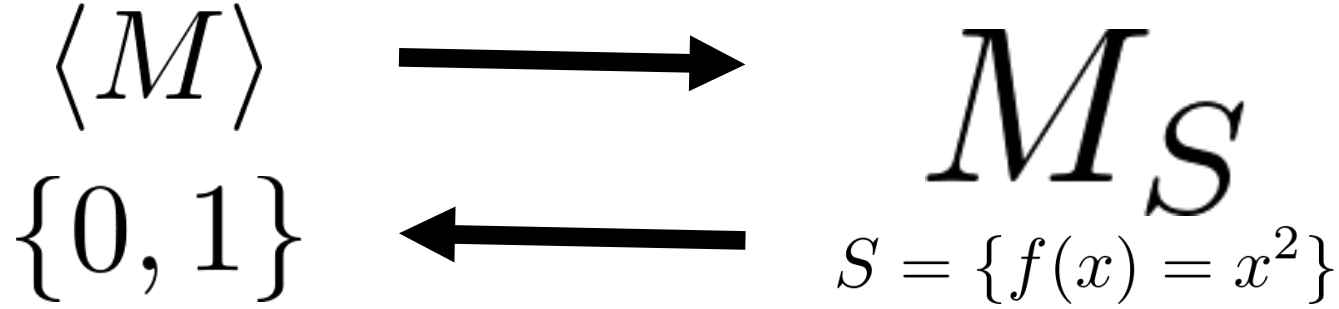
Nein / Ja $\{0, 1\}$

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

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

$$\chi_L(w) := \begin{cases} 1, & \text{falls } w \in L \\ 0, & \text{falls } w \notin L \end{cases}$$

Formalisierung



$$L(S) := \{\langle M \rangle \mid M \text{ berechn}$$

M_S entscheidet $L(S)$

Satz von Rice

$$L(S) := \{\langle M \rangle \mid M \text{ berechn}$$

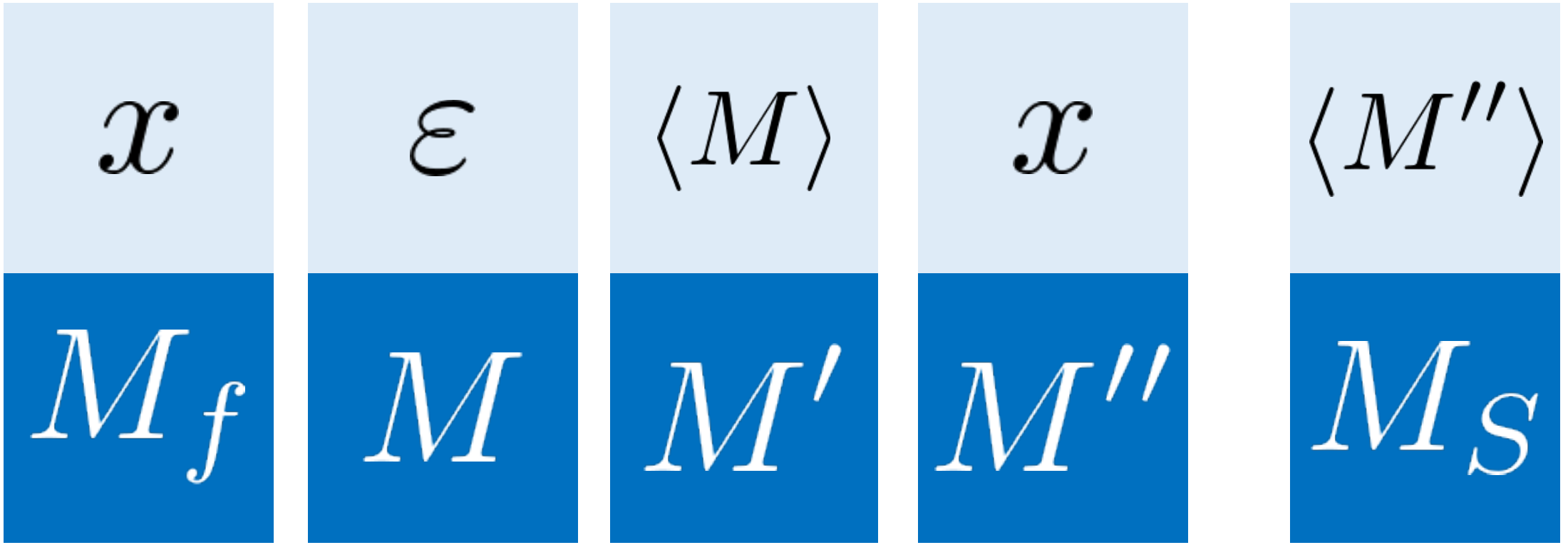
2.6.11 Satz von Rice: Die Sprache $L(S)$ ist nicht rekursiv

Beweis

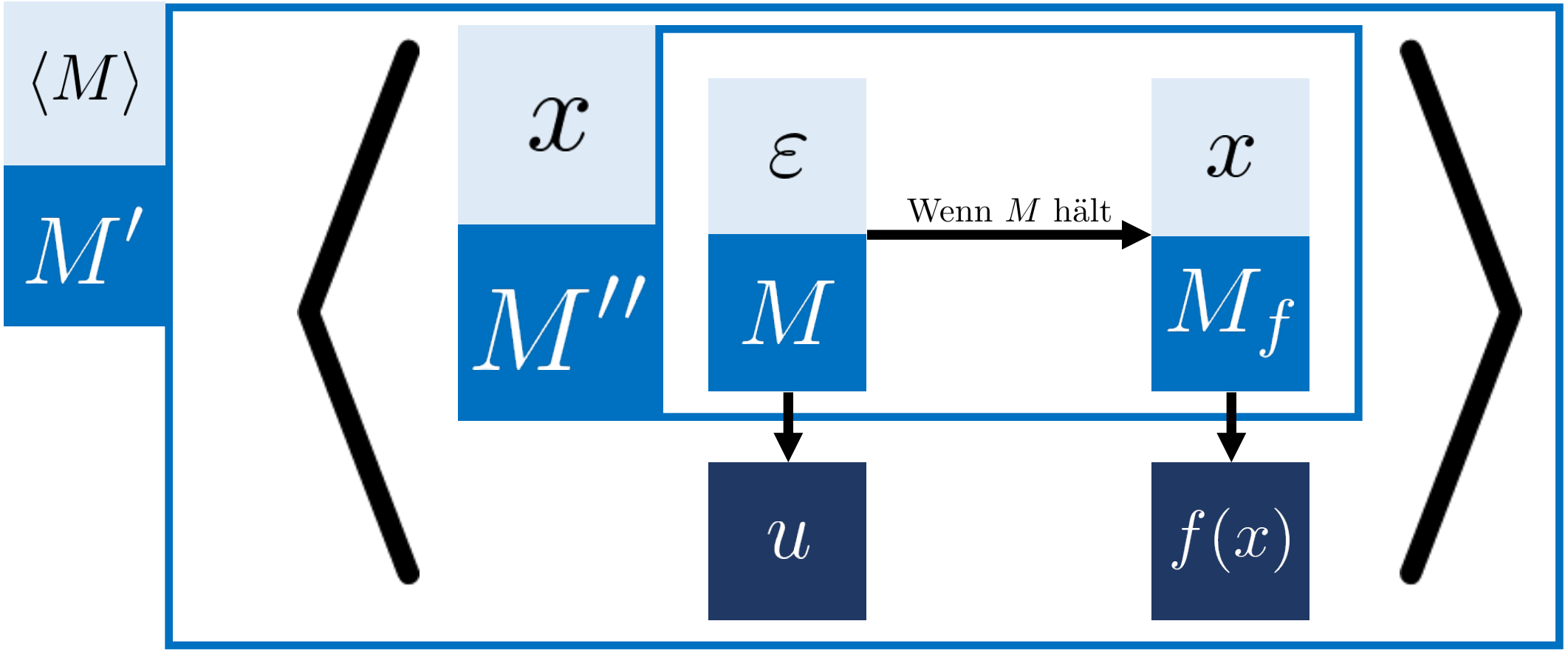
Zutaten:

- 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)


x
ε
$\langle M \rangle$
x
$\langle M'' \rangle$
M_f
M
M'
M''
M_S

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

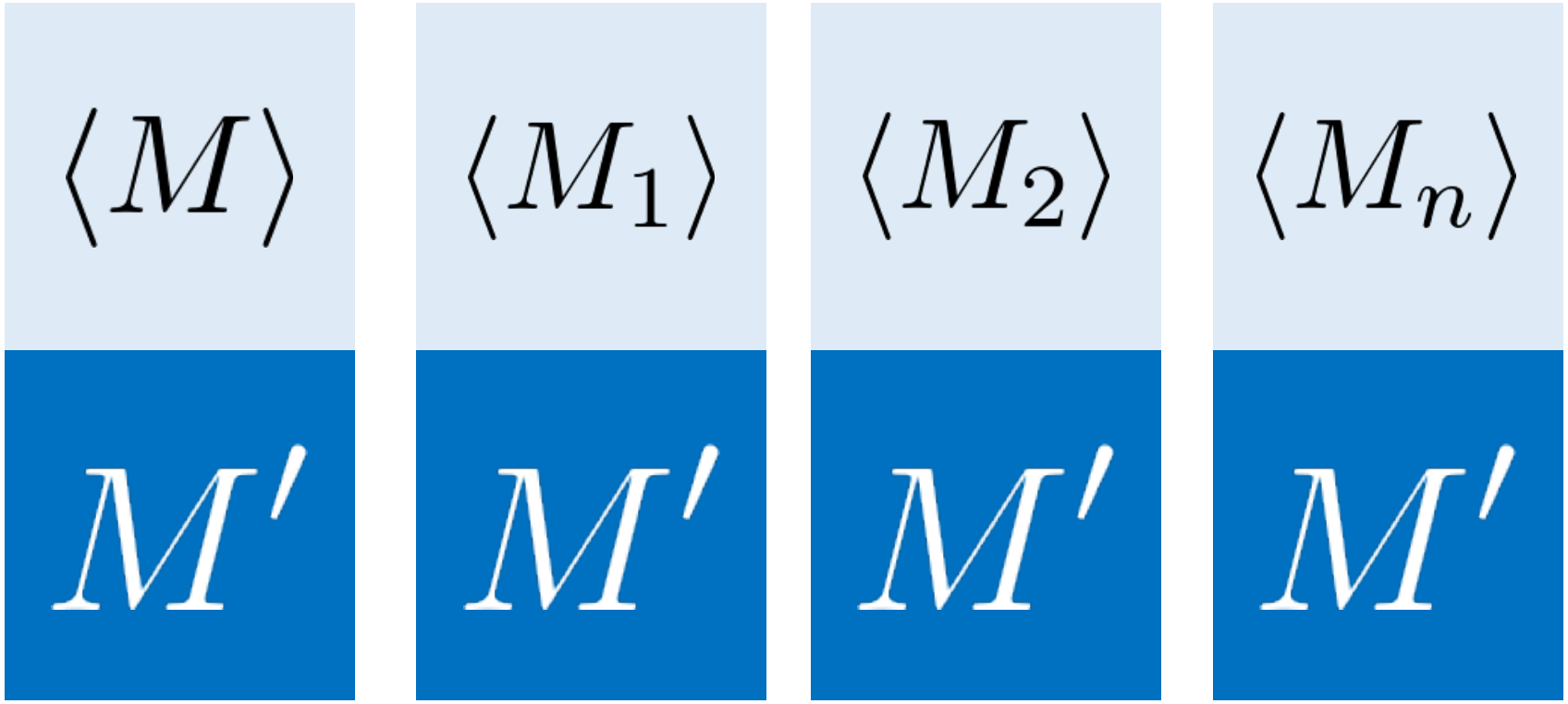


$H_\varepsilon := \{\langle M \rangle \mid M \text{ hält nicht auf der Eingabe } \varepsilon\}$

Lösungsidee

$\overline{H_\varepsilon}$

$\langle M_1 \rangle$



$\overline{H_\varepsilon} = \{\langle M_1 \rangle, \langle M_2 \rangle, \ldots, \langle M_n \rangle\}$

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**

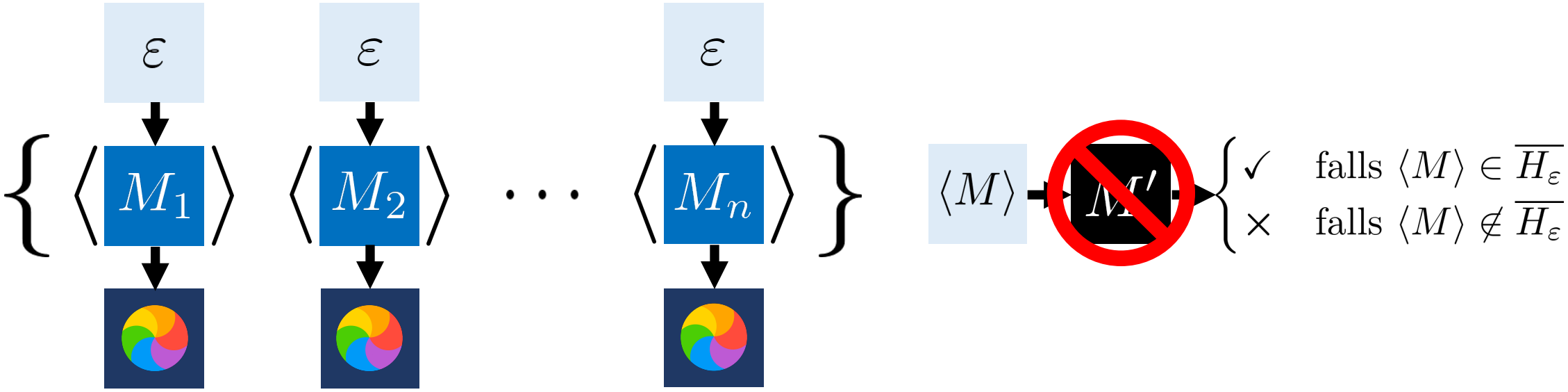
Spezielle Halteproblem

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

2.6.10 Satz: H_ε ist nicht rekursiv.

Das Spezielle Halteproblem

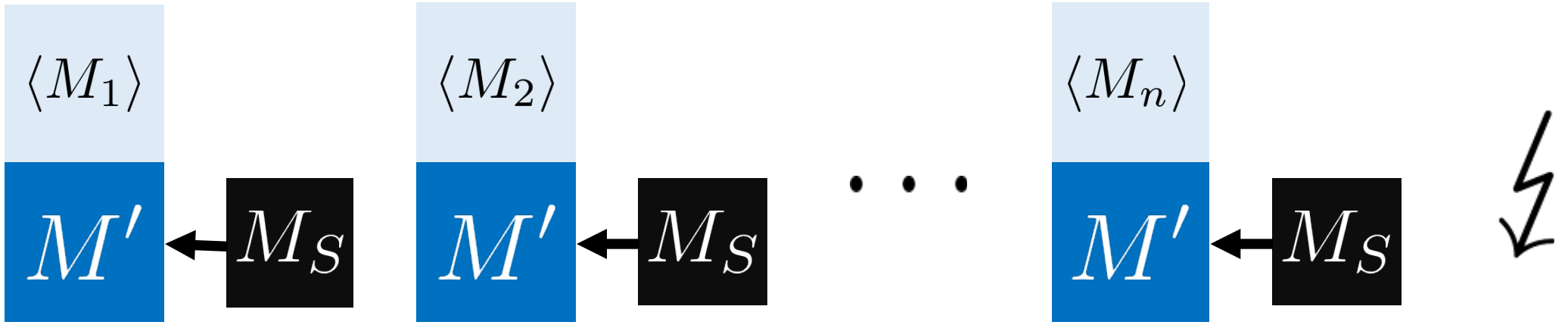
$$\overline{H_\varepsilon} := \{\langle M \rangle \mid M \text{ hält nicht auf der Eingabe } \varepsilon\}$$



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

Lösungsidee

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



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

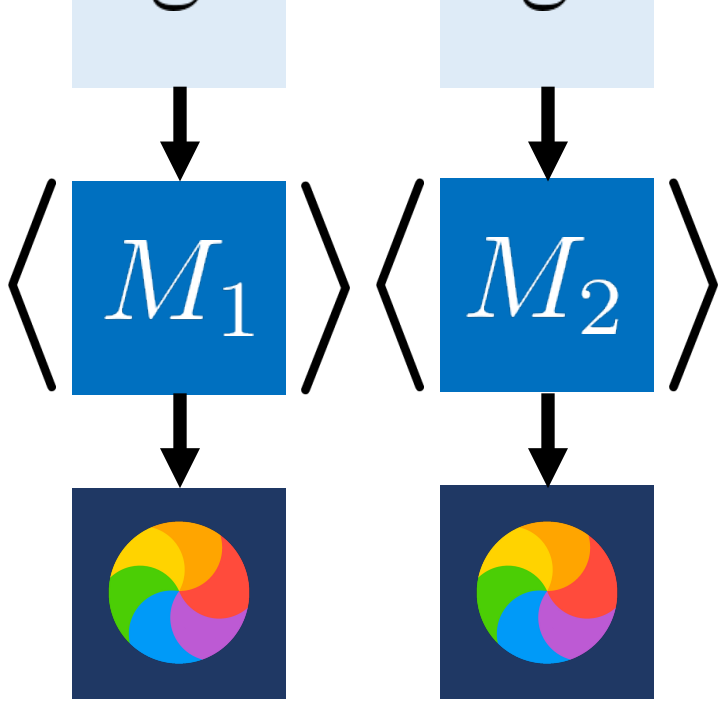
Satz 2.6.10 + Korollar 2.6.3: $\overline{H_\varepsilon}$ ist nicht rekursiv.

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

Vorüberlegung

- M1, M2, …, Mn müssen aus \ovelrine{H_\vareps} kommen
- M’ soll entscheiden, ob M1, M2, … auf die Eingabe \vareps hält.
- D.h.
  - Wenn M auf \vareps nicht hält, dann muss M’ das Programm <M> akzeptieren
  - Wenn M nauf \vareps hält, dann darf M’ das Programm <M> nicht akzeptieren

Vorüberlegung

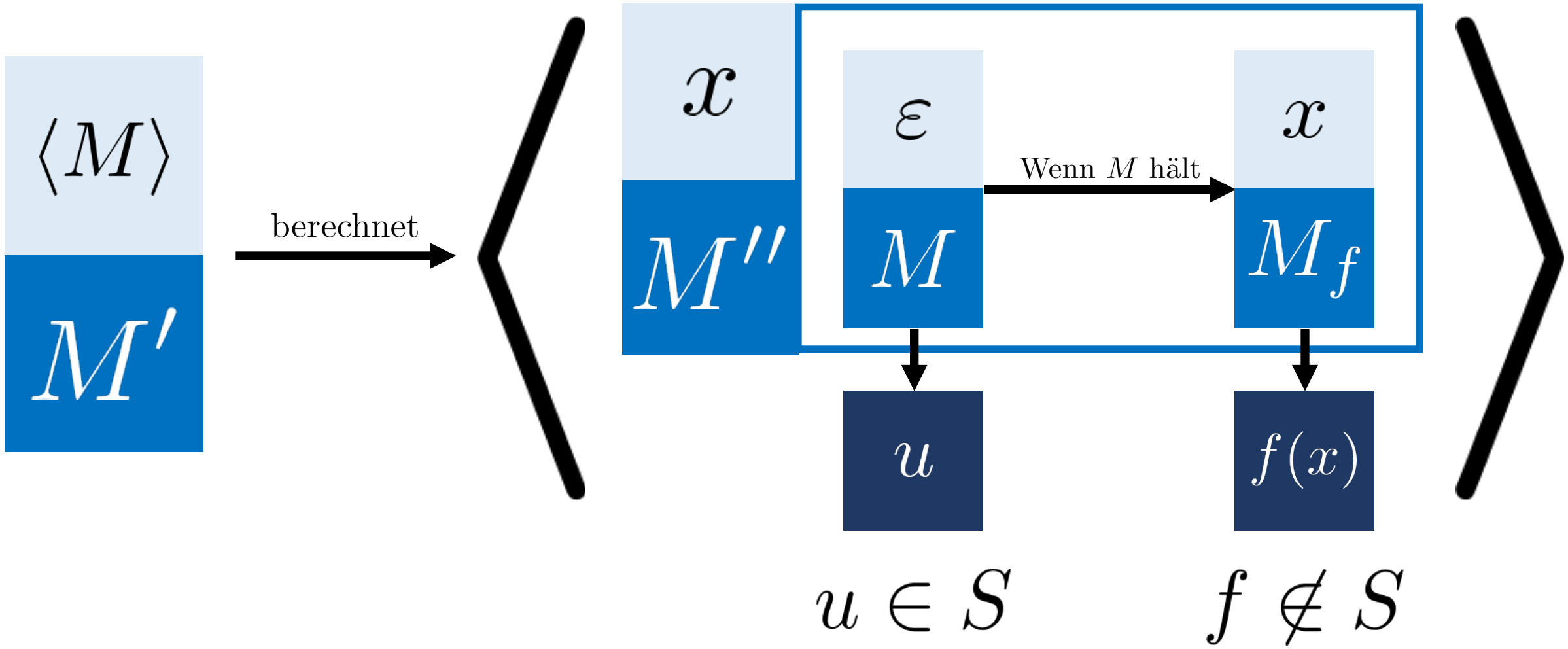


M' entscheidet, ob $M_1, M_2, \ldots, M_n$ auf die Eingabe ε hält

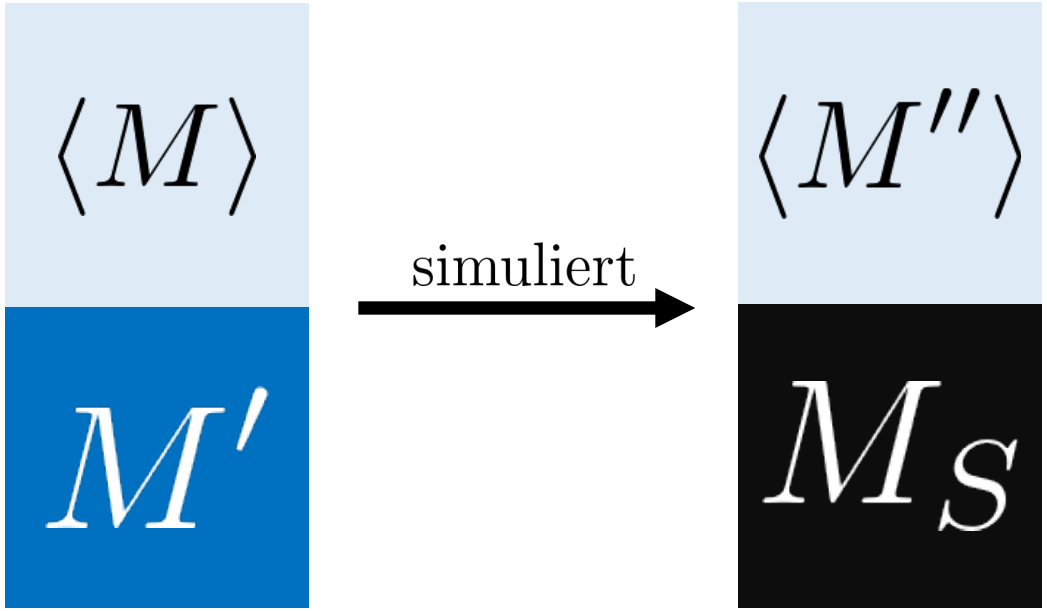
Wenn M auf ε nicht hält, dann muss M' das Programm $\langle M \rangle$ akzeptieren

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

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



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



//berechnet entweder $u \in S$ oder $f \notin S$

//Annahme

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 $\overline{H_\varepsilon}$ ↯

Was ist mit $u \notin S$?

- Wähle $f \in S$
- 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.

Warum “Teilmenge” und nicht “Element”?

- Teilmenge ist allgemeiner. D.h. selbst wenn sich unser Codechecker zwischen unterschiedlichen Funktionen entscheiden müsste (also aus den Elementen der Teilmenge S), kann er noch immer nicht entscheiden, ob unser Programmcode wirklich *irgendeine* dieser Funktionen berechnet.