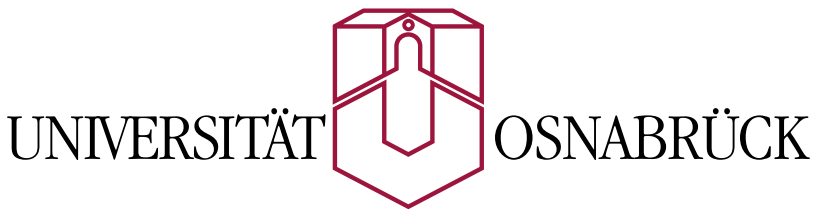
Die Notional Machine

Was ist das und Wozu brauche ich das?

Michael Brinkmeier

Institut für Informatik
AG Didaktik der Informatik

till 2026, 3. März 2026



Was Sie erwartet

Konzepte, Ideen und Gedanken, die ...

- ...seit mehreren Jahren meine Lehre prägen
  - Algorithmen und Datenstrukturen (Informatik A)
  - Informatik für Anwendende
  - schulisch und außerschulische
- ...mir geholfen haben und noch helfen ...
  - Schwierigkeiten von Lernenden zu deuten
  - Konzepte, Werkzeuge und Materialien zu entwickeln
  - Schwerpunkte aus zuschärfen
- basiert auf Beobachtungen, Gesprächen
- **nicht** systematisch evaluiert (ist auch schwierig)
- aber im Einklang mit der Literatur

„Es gibt nichts Praktischeres als eine gute Theorie“

Immanuel Kant / Kurt Lewin

Die fiktive Maschine

Die fiktive Maschine

Hinter jeder Programmiersprache steht eine **fiktive Maschine (notional machine)**, d.h. ein abstraktes Modell des tatsächlichen, technisch realisierten Computers. [Bou86] [RRR03] [Sor13]

Die fiktive Maschine

Hinter jeder Programmiersprache steht eine **fiktive Maschine (notional machine)**, d.h. ein abstraktes Modell des tatsächlichen, technisch realisierten Computers. [Bou86] [RRR03] [Sor13]

Das Modell bestimmt, wie ein Programm abgearbeitet wird. Dabei kann es zwischen verschiedenen Programmiersprachen deutliche Unterschiede bei scheinbar identischen Konzepten geben.

Die fiktive Maschine

Hinter jeder Programmiersprache steht eine **fiktive Maschine (notional machine)**, d.h. ein abstraktes Modell des tatsächlichen, technisch realisierten Computers. [Bou86] [RRR03] [Sor13]

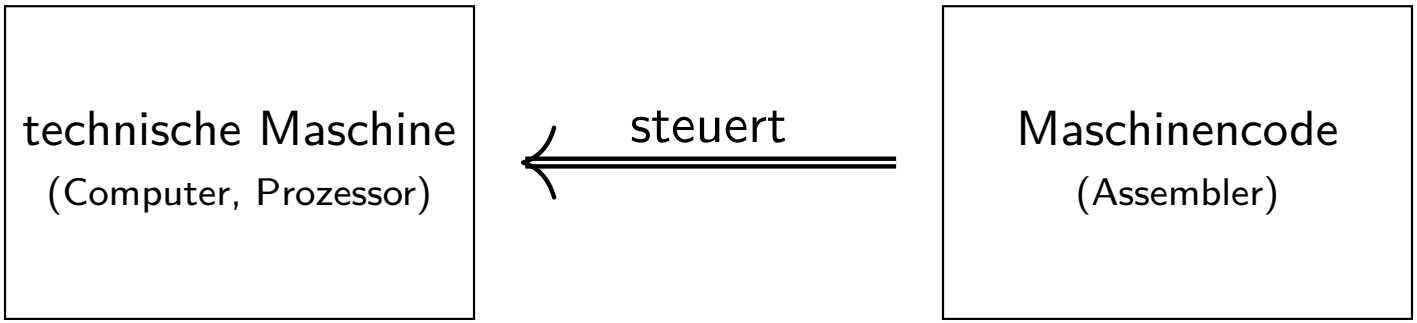
Das Modell bestimmt, wie ein Programm abgearbeitet wird. Dabei kann es zwischen verschiedenen Programmiersprachen deutliche Unterschiede bei scheinbar identischen Konzepten geben.

Besonderheit in Java

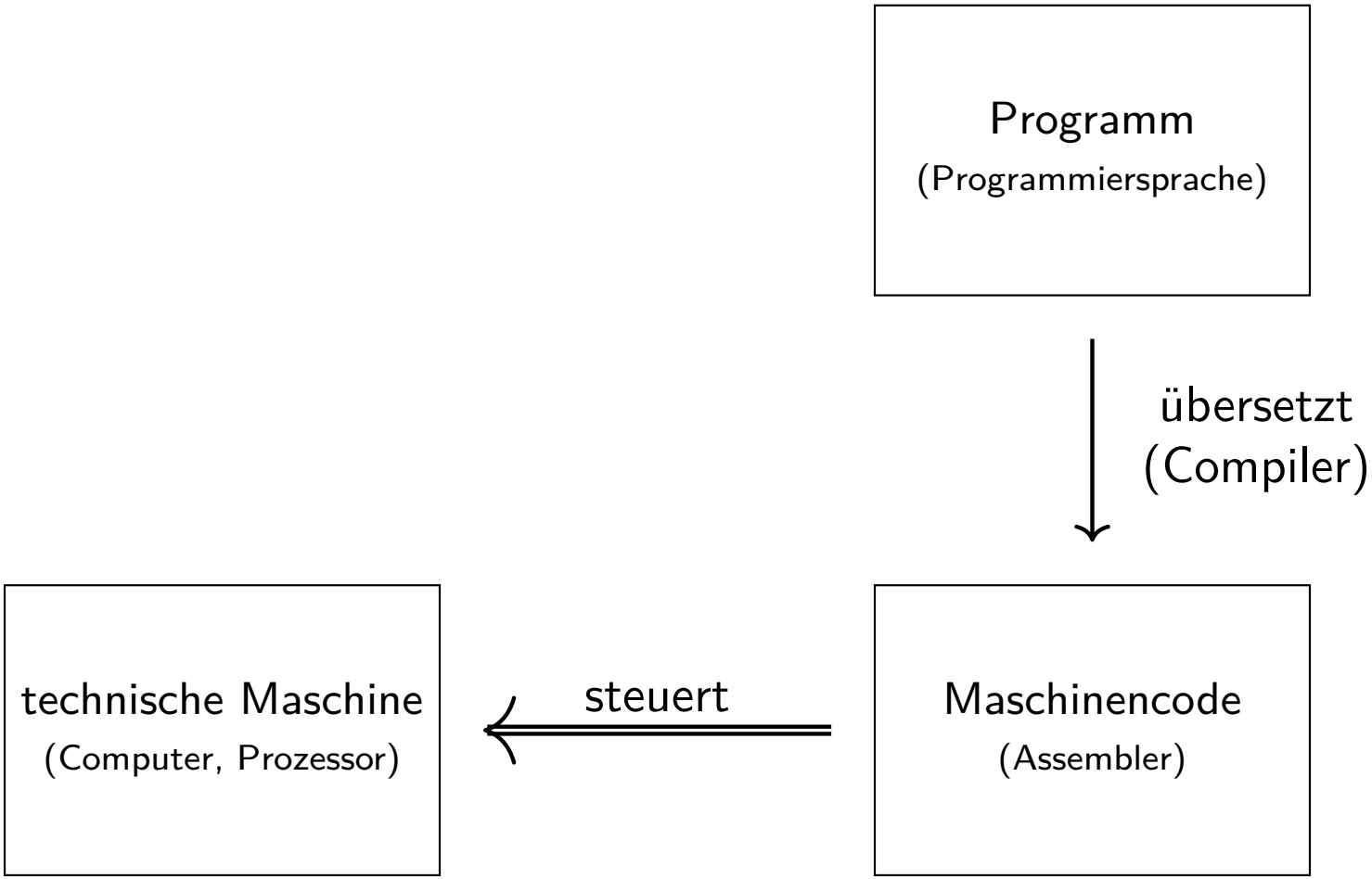
Die JVM ist eine konkrete Implementierung der fiktiven Maschine!

Bezogen auf Computer

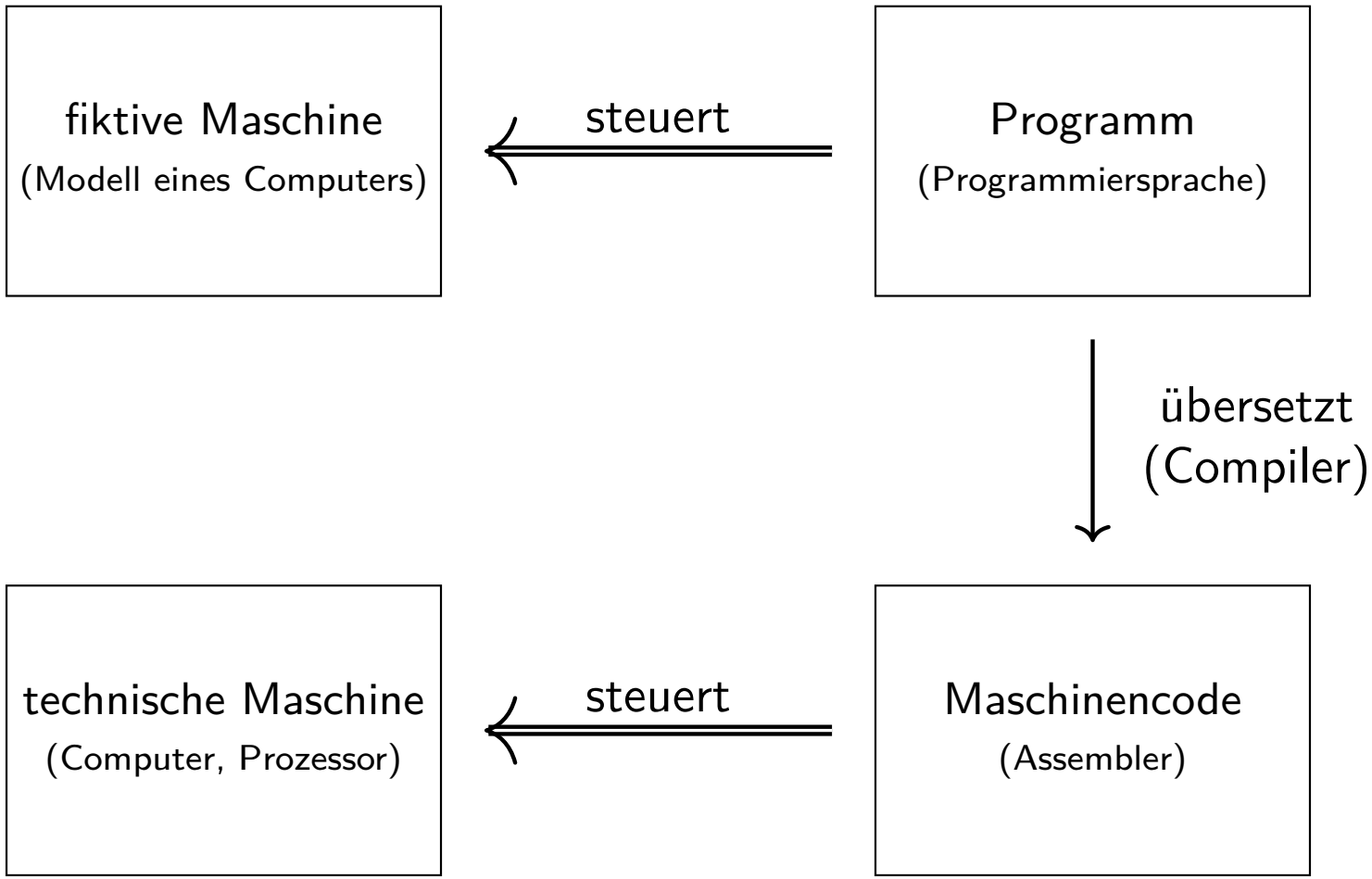
Bezogen auf Computer



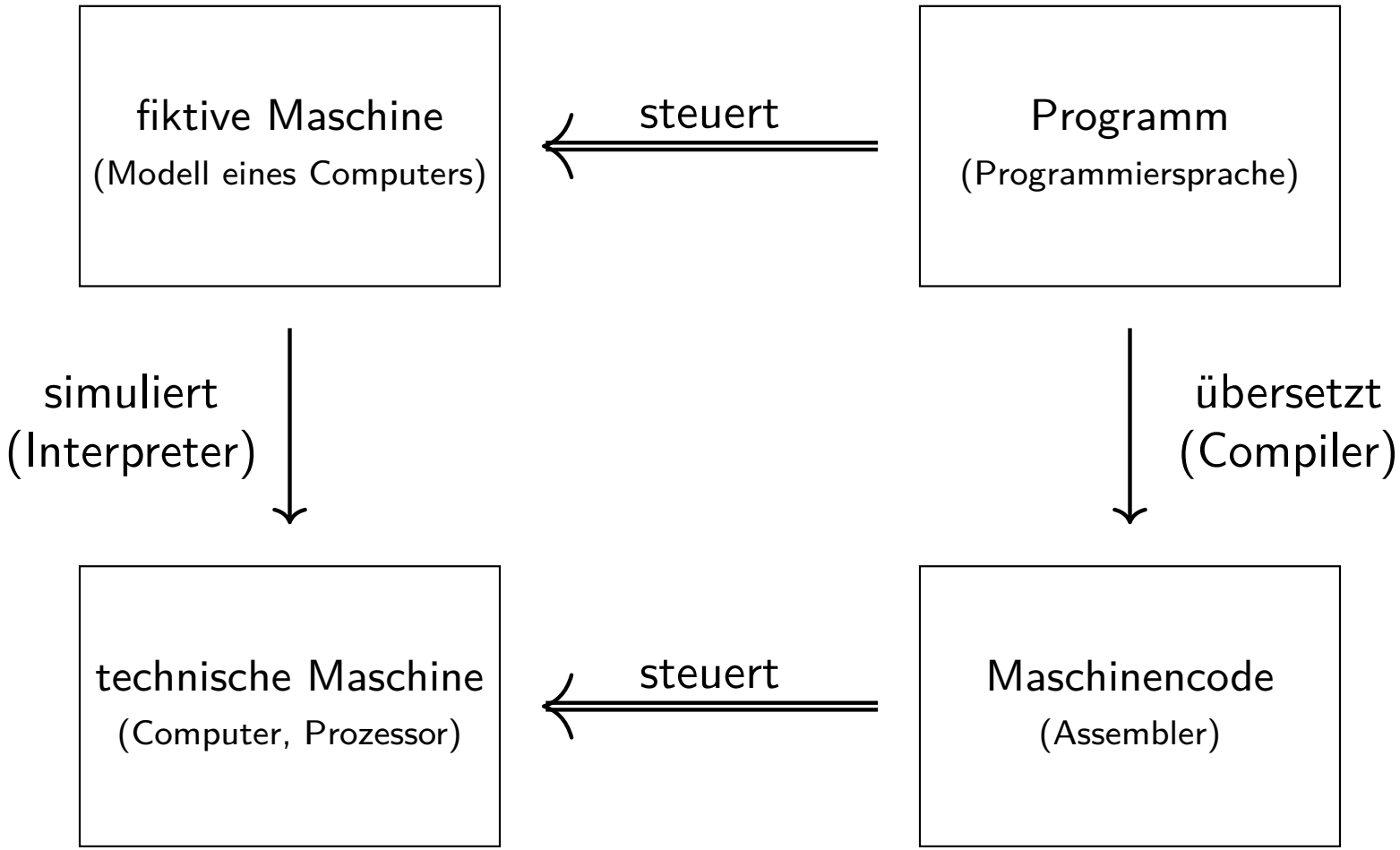
Bezogen auf Computer

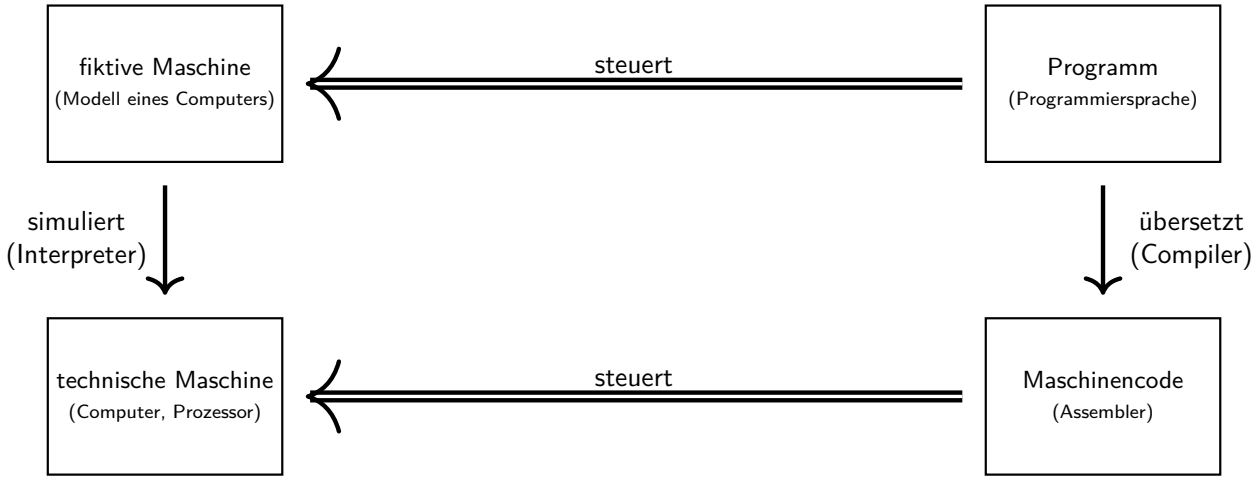


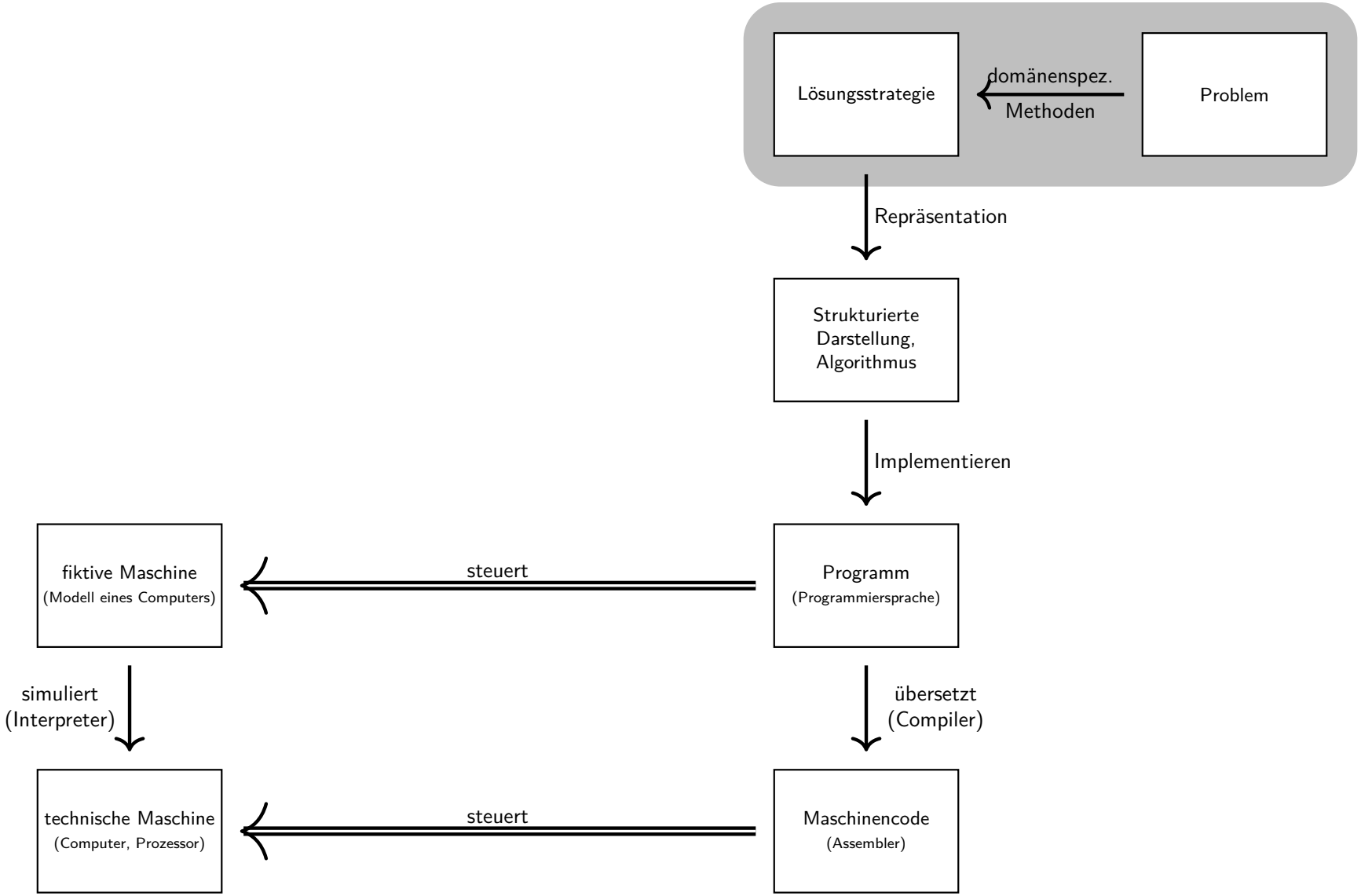
Bezogen auf Computer

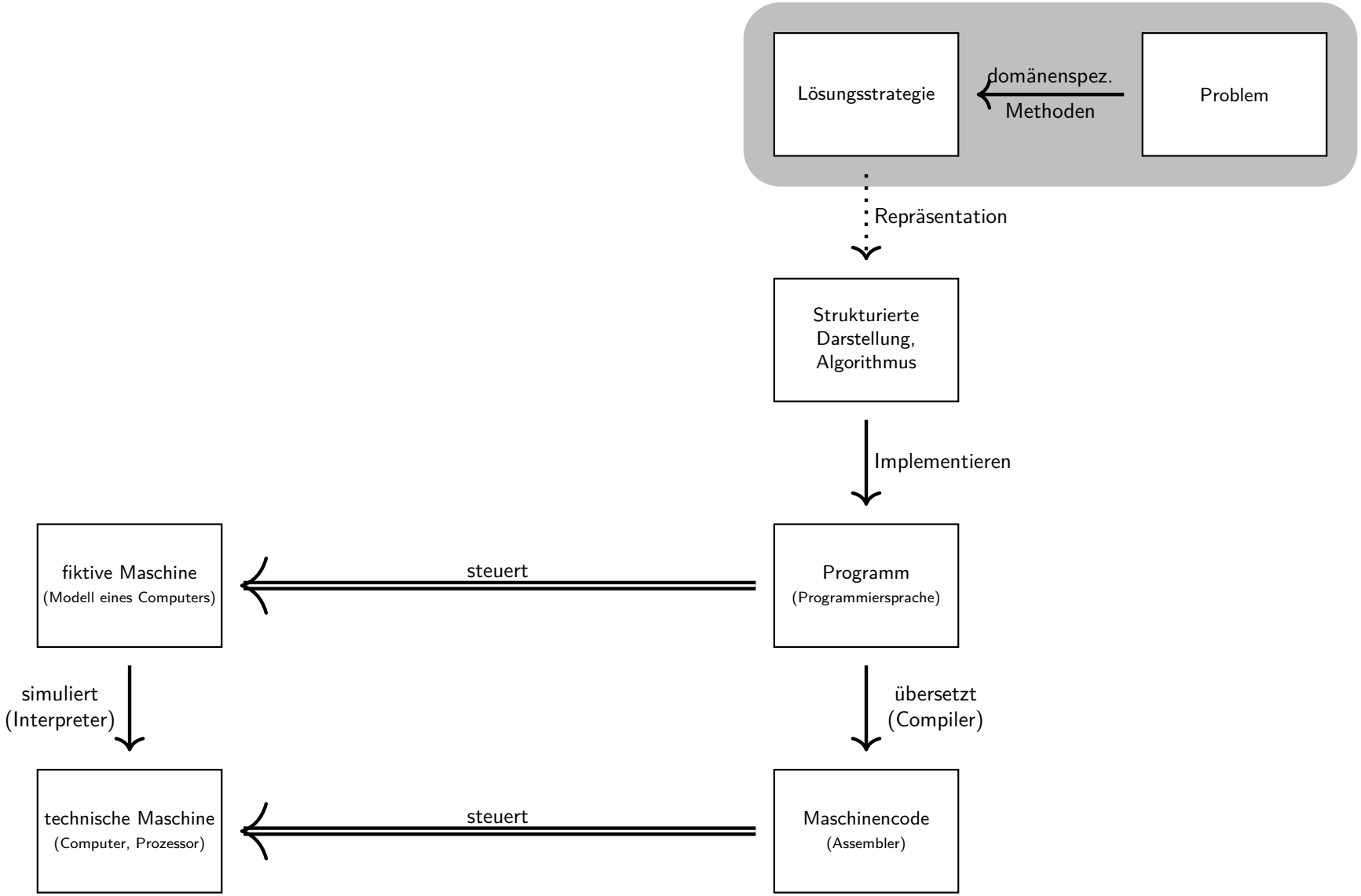


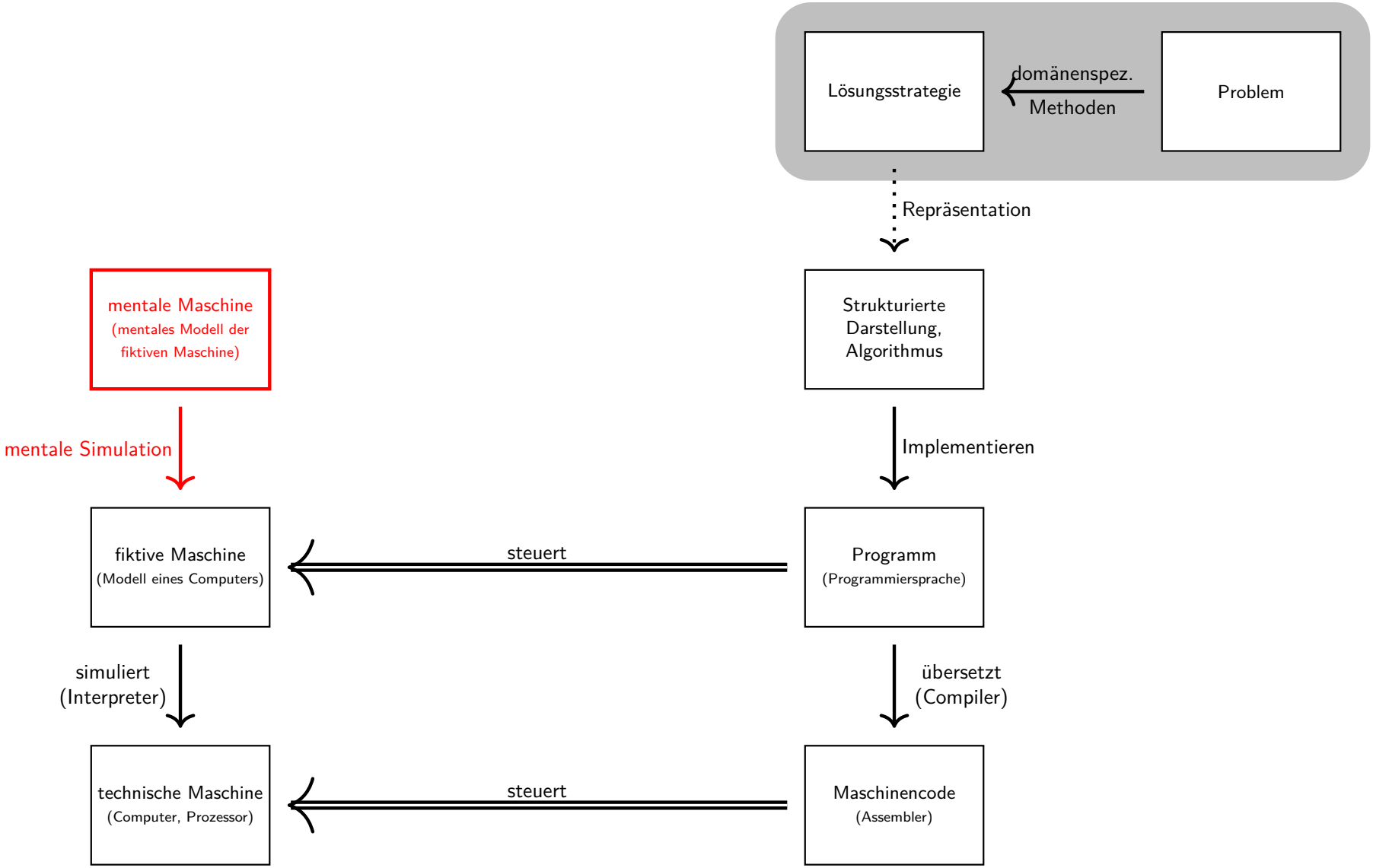
Bezogen auf Computer

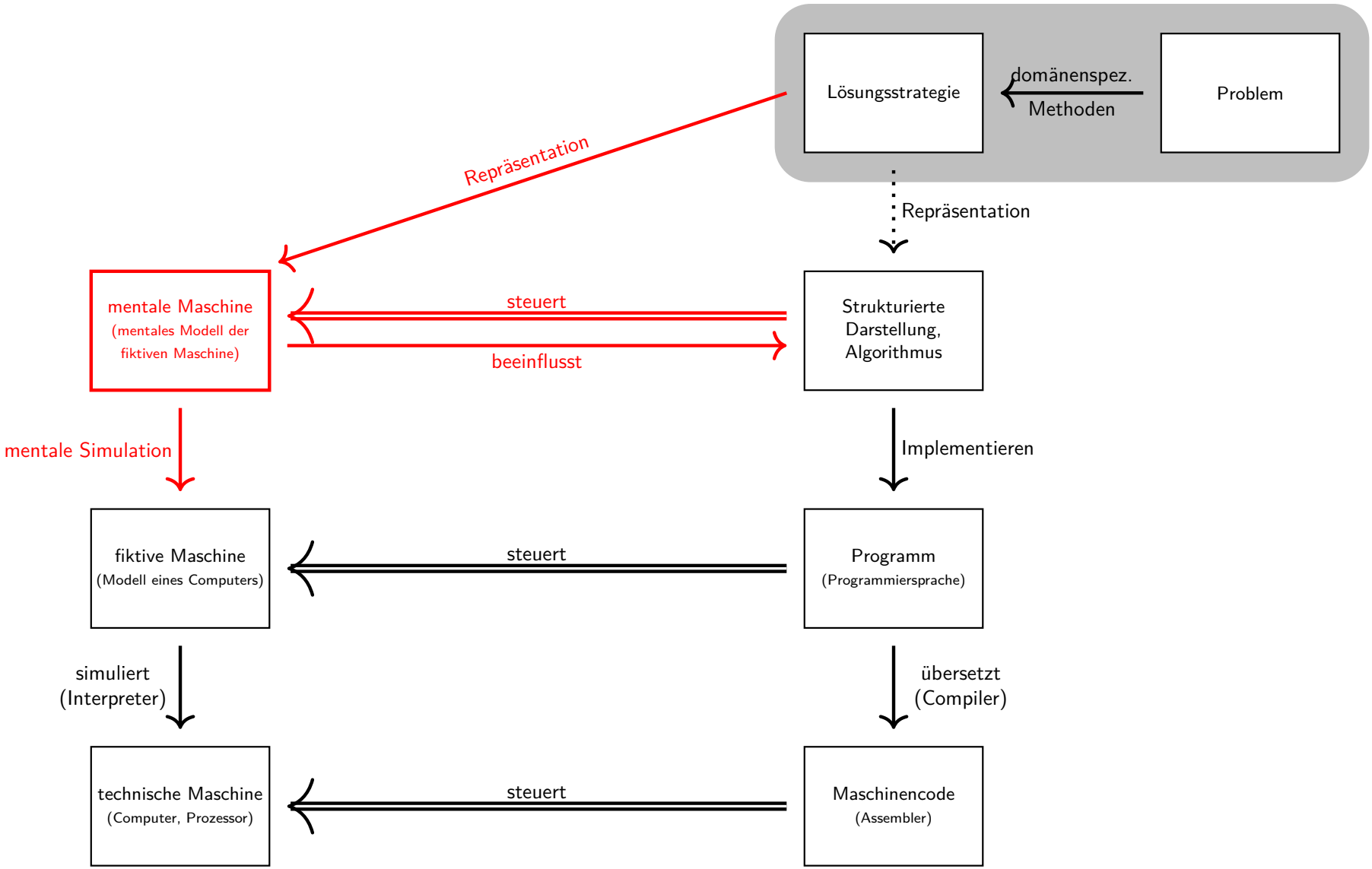



fiktive Maschine
(Modell eines Computers)
steuert
Programm
(Programmiersprache)
simuliert
(Interpreter)
übersetzt
(Compiler)
technische Maschine
(Computer, Prozessor)
steuert
Maschinencode
(Assembler)


Lösungsstrategie
domänenspez.
Methoden
Problem
Repräsentation
Strukturierte
Darstellung,
Algorithmus
Implementieren
fiktive Maschine
(Modell eines Computers)
steuert
Programm
(Programmiersprache)
simuliert
(Interpreter)
übersetzt
(Compiler)
technische Maschine
(Computer, Prozessor)
steuert
Maschinencode
(Assembler)


Lösungsstrategie
domänenspez.
Methoden
Problem
Repräsentation
Strukturierte
Darstellung,
Algorithmus
Implementieren
fiktive Maschine
(Modell eines Computers)
steuert
Programm
(Programmiersprache)
simuliert
(Interpreter)
übersetzt
(Compiler)
technische Maschine
(Computer, Prozessor)
steuert
Maschinencode
(Assembler)


Lösungsstrategie
domänenspez.
Methoden
Problem
Repräsentation
mentale Maschine
(mentales Modell der
fiktiven Maschine)
Strukturierte
Darstellung,
Algorithmus
mentale Simulation
Implementieren
fiktive Maschine
(Modell eines Computers)
steuert
Programm
(Programmiersprache)
simuliert
(Interpreter)
übersetzt
(Compiler)
technische Maschine
(Computer, Prozessor)
steuert
Maschinencode
(Assembler)


Lösungsstrategie
domänenspez.
Methoden
Problem
Repräsentation
Repräsentation
mentale Maschine
(mentales Modell der
fiktiven Maschine)
steuert
beeinflusst
Strukturierte
Darstellung,
Algorithmus
mentale Simulation
Implementieren
fiktive Maschine
(Modell eines Computers)
steuert
Programm
(Programmiersprache)
simuliert
(Interpreter)
übersetzt
(Compiler)
technische Maschine
(Computer, Prozessor)
steuert
Maschinencode
(Assembler)

Die mentale Maschine

Wenn Sie Algorithmen entwerfen oder Lösungsstrategien in eine strukturierte Form bringen wollen (Programmieren), führen Sie kontinuierlich **mentale Simulationen** des entsprechenden Algorithmus auf dem jeweiligen System durch.

Die mentale Maschine

Wenn Sie Algorithmen entwerfen oder Lösungsstrategien in eine strukturierte Form bringen wollen (Programmieren), führen Sie kontinuierlich **mentale Simulationen** des entsprechenden Algorithmus auf dem jeweiligen System durch.

Dazu verwenden Sie ein **mentales Modell** der fiktiven Maschine, die **mentale Maschine**.

Die mentale Maschine

Wenn Sie Algorithmen entwerfen oder Lösungsstrategien in eine strukturierte Form bringen wollen (Programmieren), führen Sie kontinuierlich **mentale Simulationen** des entsprechenden Algorithmus auf dem jeweiligen System durch.

Dazu verwenden Sie ein **mentales Modell** der fiktiven Maschine, die **mentale Maschine**.

These

Anfänger verfügen – in der Regel – über keine vollständige, kohärente und/oder robuste mentales Maschine. Daher fällt ihnen der Prozess der Umsetzung der Lösungsstrategie in ein strukturiertes Programm schwer.

Die mentale Maschine

Wenn Sie Algorithmen entwerfen oder Lösungsstrategien in eine strukturierte Form bringen wollen (Programmieren), führen Sie kontinuierlich **mentale Simulationen** des entsprechenden Algorithmus auf dem jeweiligen System durch.

Dazu verwenden Sie ein **mentales Modell** der fiktiven Maschine, die **mentale Maschine**.

These

Anfänger verfügen – in der Regel – über keine vollständige, kohärente und/oder robuste mentales Maschine. Daher fällt ihnen der Prozess der Umsetzung der Lösungsstrategie in ein strukturiertes Programm schwer.

Unter Umständen herrschen sogar **Fehlvorstellungen**, wie z.B. beim Konzept der **Variablen**, da die Begriffe bereits durch andere Konzepte besetzt sind.

Die Herausforderung

Problem

Die Bildung der **mentalen Maschine** ist ein sehr subjektiver Prozess, der nur schwer von Außen gesteuert werden kann. Er kann allerdings durch geeignete Aufgaben, Analogien, Visualisierungen und Modelle unterstützt werden.

Die Herausforderung

Problem

Die Bildung der **mentalen Maschine** ist ein sehr subjektiver Prozess, der nur schwer von Außen gesteuert werden kann. Er kann allerdings durch geeignete Aufgaben, Analogien, Visualisierungen und Modelle unterstützt werden.

Konsequenz

Schülerinnen und Schüler müssen aktiv an **Ihrer** mentalen Maschine arbeiten! Und Lehrkräfte sollten diesen Prozess gestalten und begleiten, indem sie ihnen bei der **Konstruktion** einer korrekten, kohärenten und robusten mentalen Maschine helfen.

Didaktische Maschinen

Pädagogische Maschine

Für die Unterstützung der Lernenden bei der Konstruktion ihrer mentalen Maschine können speziell konstruierte fiktive Maschinen genutzt werden, die wir **didaktische** oder **pädagogische Maschinen** nennen.

Man beachte!

Die **fiktive Maschine** einer Programmiersprache ist eine **technische Gegebenheit** und nicht „verhandelbar“.

Didaktische Maschinen

Pädagogische Maschine

Für die Unterstützung der Lernenden bei der Konstruktion ihrer mentalen Maschine können speziell konstruierte fiktive Maschinen genutzt werden, die wir **didaktische** oder **pädagogische Maschinen** nennen.

Man beachte!

Die **fiktive Maschine** einer Programmiersprache ist eine **technische Gegebenheit** und nicht „verhandelbar“.

Achtung!!

In der Literatur hat sich der Begriff der **notional machine** zum Teil so verschoben, dass damit **didaktische Maschinen** gemeint sind!

Die Rolle der didaktischen Maschine

Die Rolle der didaktischen Maschine

- Sie **reduziert** die Komplexität bestimmter Aspekte der fiktiven Maschine
- Sie dient der **Integration** in die mentale Maschine des Lernenden
- Sie ist in der Regel eine enaktive oder ikonische Repräsentation (symbolisch eher selten)

Webseite mit einer Reihe von Beispielen:
`https://notionalmachines.github.io/notional-machines.html`

Vorsicht!

Nicht alles was dort zu finden ist sind gute und sinnvolle pädagogische Maschinen. Einige würde ich auch gar nicht als solche bezeichnen.

Die Rolle der didaktischen Maschine

- Sie **reduziert** die Komplexität bestimmter Aspekte der fiktiven Maschine
- Sie dient der **Integration** in die mentale Maschine des Lernenden
- Sie ist in der Regel eine enaktive oder ikonische Repräsentation (symbolisch eher selten)

Webseite mit einer Reihe von Beispielen:
`https://notionalmachines.github.io/notional-machines.html`

Vorsicht!

Nicht alles was dort zu finden ist sind gute und sinnvolle pädagogische Maschinen. Einige würde ich auch gar nicht als solche bezeichnen.

Was bedeutet das?

Kenne die fiktive Maschine

Kenne die fiktive Maschine!

- Allgemeine Prinzipien
- Spezifika der Programmiersprache
- **Unterschiede** zu anderen Programmiersprachen
- **Best-Practices**

- Scratch
  - quasi-paralleles Multithreading
  - Ereignisse
  - schwache, dynamische und explizite Typisierung
  - mit Prototypen vergleichbar
  - Nachrichten
- Python
  - Singlethread
  - starke, dynamische und implizite Typisierung
  - Klassen
  - Funktionsaufrufe

Die didaktische muss zur fiktiven Maschine passen!

Die didaktische Maschine muss zur fiktiven Maschine passen!

Die diaktische Maschine sollte eine **konsistente** Reduktion der fiktiven Maschine darstellen.

Die didaktische muss zur fiktiven Maschine passen!

Die didaktische Maschine muss zur fiktiven Maschine passen!

Die diaktische Maschine sollte eine **konsistente** Reduktion der fiktiven Maschine darstellen.

Großes Problem

Variablen sind aus Mathe bekannt!
Aber die Konzepte unterscheiden sich deutlich! (in der Mathematik gibt es schon mindestens zwei Variablenkonzepte)
Das führt zu einer falschen Grundlage für die mentale Maschine.

Die didaktische muss zur fiktiven Maschine passen!

Die didaktische Maschine muss zur fiktiven Maschine passen!

Die diaktische Maschine sollte eine **konsistente** Reduktion der fiktiven Maschine darstellen.

Großes Problem

Variablen sind aus Mathe bekannt!
Aber die Konzepte unterscheiden sich deutlich! (in der Mathematik gibt es schon mindestens zwei Variablenkonzepte)
Das führt zu einer falschen Grundlage für die mentale Maschine.

Schlechtes Beispiel

Das **Schachtelmodell für Variablen** [Bou86]

- Werte werden verschoben, nicht kopiert
- Objekte werden in die Schachteln getan, keine Referenzen
- alte Werte bleiben in der Schachtel (Stack-Misskonzept)

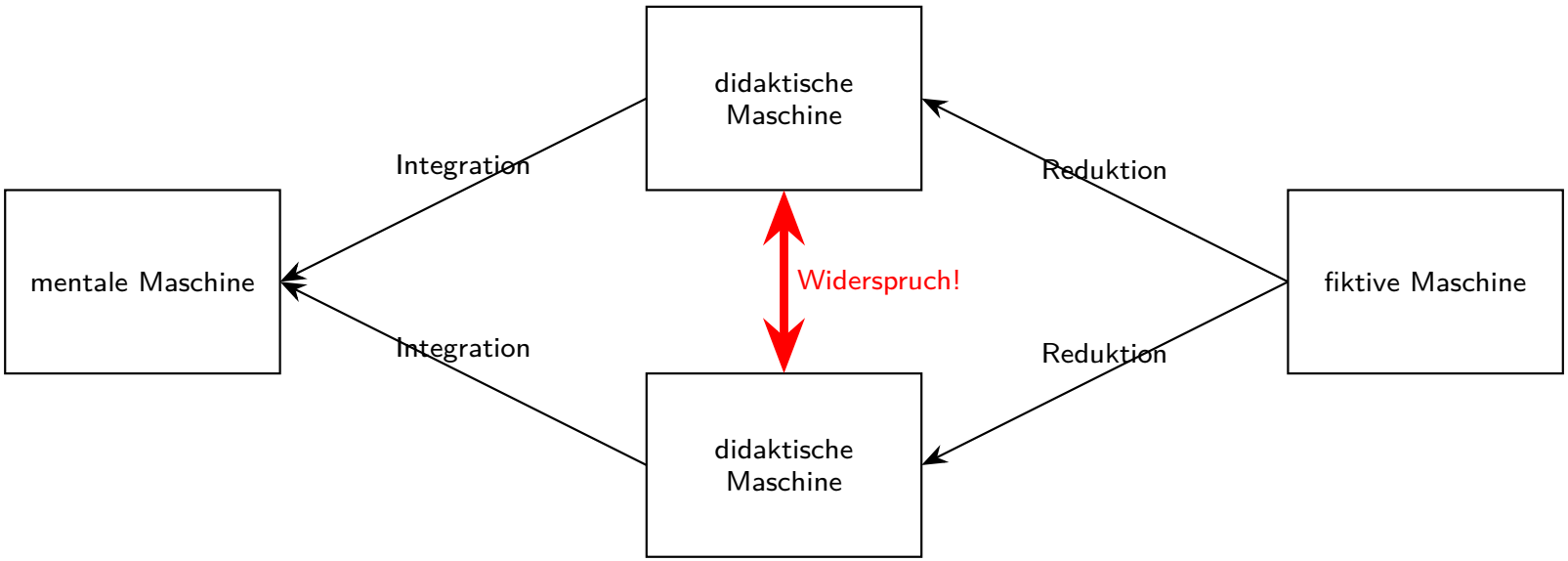
Etwas plakativ . . .

Nicht jede Analogie ist gut oder eine geeignete didaktische Maschine.

Die didaktischen Maschinen müssen kompatibel sein

Die didaktische Maschine muss zur fiktiven Maschine passen!

Die verwendeten diaktischen Maschinen für verschieden Aspekte müssen untereinander kompatibel sein und zu einem konsistenten mentalen Modell beitragen.



Wechsel der Programmiersprache

Wechsel der Programmiersprache

Der Wechsel der Programmiersprache geht immer mit einem Wechsel der fiktiven Maschinen und der mentalen Maschine einher.

- Syntaktische Ähnlichkeit hilft nur bedingt
- Trifft auch auf visuelle Sprachen (Blocksprachen etc.) zu
- Anschlussfähigkeit zu weiterführendem Unterricht sollte gewährleistet sein
- **Best-Practices** widersprechen sich zum Teil (z.B. Scratch [MSABA11])
- Die fiktiven Maschinen sind nicht für den Wechsel gemacht

Visuell und enaktiv

Nutze Visualisierungen und enaktive Maschinen

Die diaktischen Maschinen sollten eine Visualisierung der Abläufe und/oder eine enaktive Simulation ermöglichen.

Visualisierung der Funktionsweise

- Wie funktioniert die fiktive Maschine?
- Wie werden die Abläufe organisiert?
- Welche Regeln müssen beachtet werden?

Visualisierung der Wirkung des Programms

- Microworlds
- Das Programm beeinflusst ein visualisiertes System
- Die Wirkungen werden während des Ablaufs wahrnehmbar
- Es kann ein Zusammenhang zwischen den Abläufen auf der fiktiven Maschine und den Wirkungen auf das gesteuerte System hergestellt werden

Baue die mentale Maschine auf

Baue die mentale Maschine auf

Verwende didaktische Maschinen so, dass die Lernenden sie gut in ihre mentale Maschine integrieren können.

Baue die mentale Maschine auf

Baue die mentale Maschine auf

Verwende didaktische Maschinen so, dass die Lernenden sie gut in ihre mentale Maschine integrieren können.

Vorsicht vor explorativem Vorgehen!

Ein rein exploratives Vorgehen scheint keine tragfähige mentale Maschine zu Bauen (z.B. Scratch [MSABA11]).

Baue die mentale Maschine auf

Baue die mentale Maschine auf

Verwende didaktische Maschinen so, dass die Lernenden sie gut in ihre mentale Maschine integrieren können.

Vorsicht vor explorativem Vorgehen!

Ein rein exploratives Vorgehen scheint keine tragfähige mentale Maschine zu Bauen (z.B. Scratch [MSABA11]).

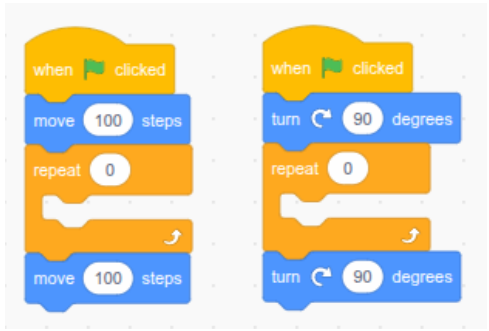
Lernmodell nach [MFR+08]:

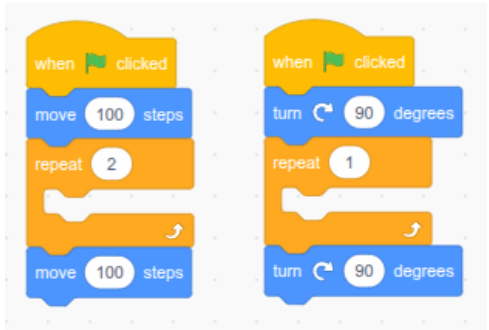
1. **Vorbereitung:** Knüpfe an bekannte Aspekte der bestehenden mentalen Modelle an (müssen dazu bekannt sein)
2. **Kognitiver Konflikt:** Erzeuge ein widersprüchliches Ereignis, dass mit dem aktuellen mentalen Modell nicht erklärt werden kann
3. **Modellkonstruktion:** Löse den Konflikt mithilfe eines angepassten Modells auf (didaktische Maschine)
4. **Anwendung:** Wende den neuen Aspekt in Kombination mit alten Aspekten an (Festigung des mentalen Modells)

Scratch und seine fiktive Maschine

Problem mit der fiktiven Maschine von Scratch

- Identisch aussehende Programme zeigen unterschiedliches Verhalten
- Multithreading und Quasi-Paralellität sind nicht intuitiv verständlich oder nachvollziehbar
- Multithreading und Quasi-Paralellität erfordert bestimmte Problemlösungen, die in anderen Programmiersprachen ungeeignet sind
- Exploratives Vorgehen reicht nicht aus [MSABA11]
- Teilweise ergeben sich grundlegende Misskonzepte [HSAS18], z.B.
  - Probleme mit der Sequentialität von Programmen (56%)
  - Mehrfache Werte (42,0%)
  - Variablenwerte werden nicht gespeichert (33.3%)





Konflikte zwischen mentaler und fiktiver Maschine

- Unterscheiden sich die mentale und die fiktive Maschine ...
  - werden Programme unter falschen Voraussetzungen erstellt
  - kann scheinbar fehlerhaftes Verhalten nicht erklärt werden
  - ist kein sinnvolles Debugging möglich
  - können die Unterschiede häufig nicht ëntdeckt"werden
- Bleiben Mechanismen der fiktiven Maschine unsichtbar ...
  - kann keine Hypothese über die Funktionsweise gebildet werden
  - bleibt das Verhalten der Maschine rätselhaft

Konflikte zwischen mentaler und fiktiver Maschine

- Unterscheiden sich die mentale und die fiktive Maschine ...
  - werden Programme unter falschen Voraussetzungen erstellt
  - kann scheinbar fehlerhaftes Verhalten nicht erklärt werden
  - ist kein sinnvolles Debugging möglich
  - können die Unterschiede häufig nicht ëntdeckt"werden
- Bleiben Mechanismen der fiktiven Maschine unsichtbar ...
  - kann keine Hypothese über die Funktionsweise gebildet werden
  - bleibt das Verhalten der Maschine rätselhaft

Im schlimmsten Fall

Die Lernenden beginnen „rumzuprobieren“ und nehmen die Tätigkeit der Programmierens als etwas wahr, dass sie nicht verstehen. Das Ganze wird als „nicht-deterministisch“ wahrgenommen.

Paco - Der Papier Computer

Paco - Der Papier Computer

- **Paco** (ein Akronym von Paper Computer) kann hervorragend rechnen. Sieht er eine Rechnung kann er das Ergebnis im Kopf berechnen.

Paco - Der Papier Computer

- **Paco** (ein Akronym von Paper Computer) kann hervorragend rechnen. Sieht er eine Rechnung kann er das Ergebnis im Kopf berechnen.
- Er hat aber ein **schlechtes Gedächtnis**. Werte kann er sich nur während der Abarbeitung einer Anweisung merken. Bei der nächsten Anweisung hat er alle Werte wieder vergessen.

Paco - Der Papier Computer

- **Paco** (ein Akronym von Paper Computer) kann hervorragend rechnen. Sieht er eine Rechnung kann er das Ergebnis im Kopf berechnen.
- Er hat aber ein **schlechtes Gedächtnis**. Werte kann er sich nur während der Abarbeitung einer Anweisung merken. Bei der nächsten Anweisung hat er alle Werte wieder vergessen.
- Er kann hervorragend schriftliche Anweisungen befolgen, wenn sie richtig aufgeschrieben wurden (Syntax).

Paco - Der Papier Computer

- **Paco** (ein Akronym von Paper Computer) kann hervorragend rechnen. Sieht er eine Rechnung kann er das Ergebnis im Kopf berechnen.
- Er hat aber ein **schlechtes Gedächtnis**. Werte kann er sich nur während der Abarbeitung einer Anweisung merken. Bei der nächsten Anweisung hat er alle Werte wieder vergessen.
- Er kann hervorragend schriftliche Anweisungen befolgen, wenn sie richtig aufgeschrieben wurden (Syntax).

Er arbeitet mit . . .

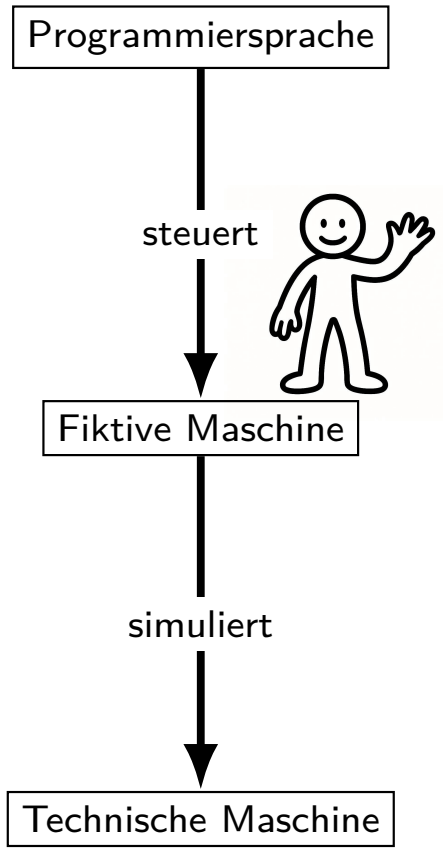
- . . . beliebig vielen Papierbögen
- . . . beliebig viele Klebezettel
- . . . einem Kugelschreiber (nicht radierbar)
- . . . einem Bleistift (radierbar) und
- . . . einem Radiergummi.

Die Idee hinter Paco

Feststellung

Jede Programmiersprache basiert auf einem **Modell** eines Computers, der **fiktiven Maschine**.

- Die Programmiersprache steuert diese **fiktive Maschine**.
- Unterschiede in einigen Aspekten.
- Grundmechanismen sind sehr ähnlich.



Die Idee hinter Paco

Feststellung

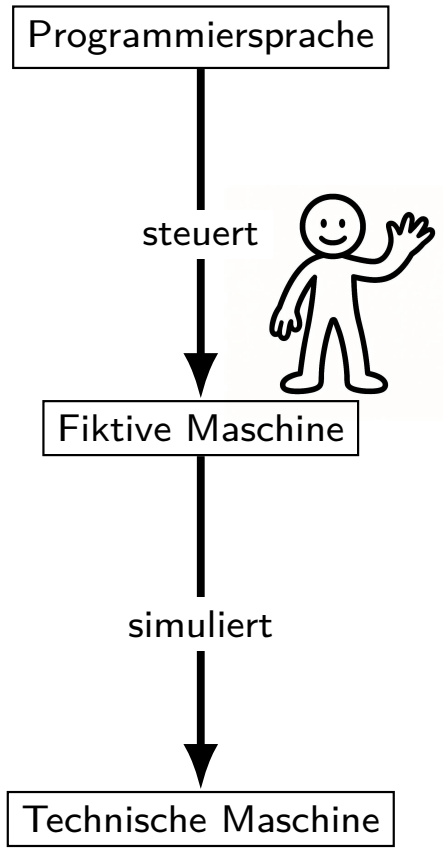
Jede Programmiersprache basiert auf einem **Modell** eines Computers, der **fiktiven Maschine**.

- Die Programmiersprache steuert diese **fiktive Maschine**.
- Unterschiede in einigen Aspekten.
- Grundmechanismen sind sehr ähnlich.

Die Idee von Paco

Paco ist eine **fiktive Maschine**, die auf **menschlichen Handlungen** beruht.

- Die Arbeitsweise ist leicht verständlich und durchführbar.
- Das Verhalten von Paco hat **direkte Entsprechungen** in den fiktiven Maschinen.
- Paco ist auf verschiedene Programmiersprachen adaptierbar.

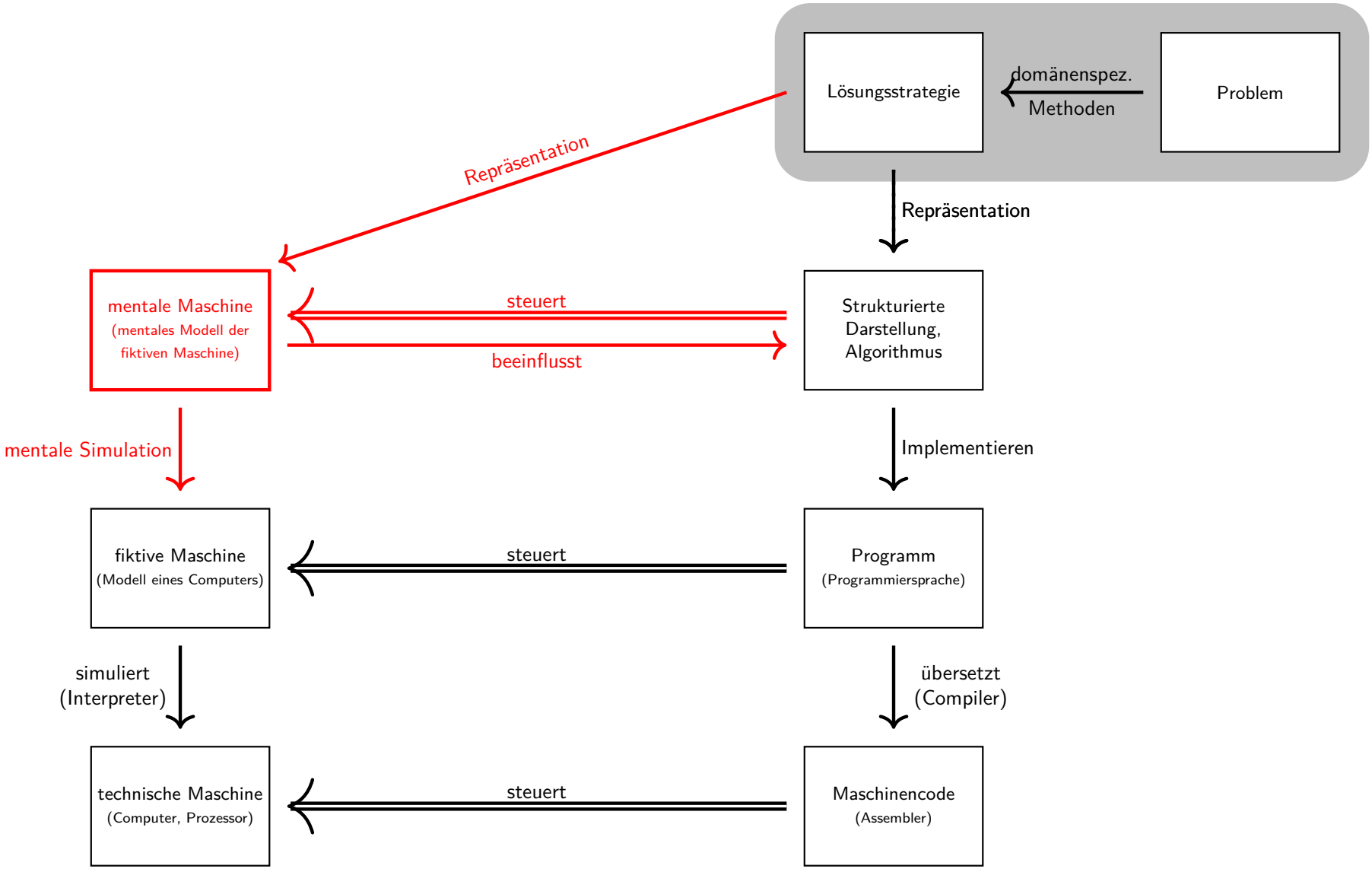


Jenseits des Programmierens

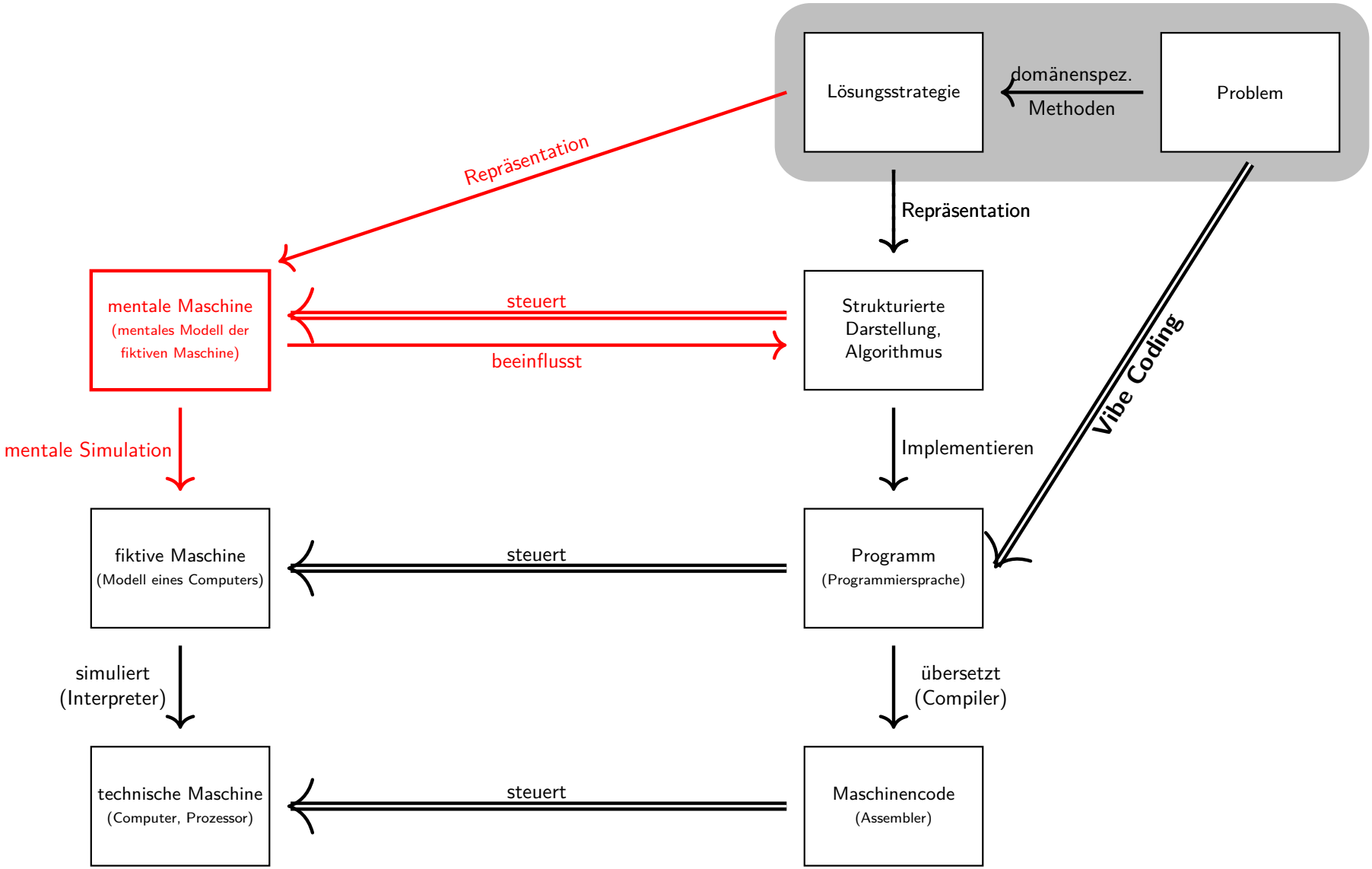
Übertragung auf andere Konzepte

- Datenbanken, bzw. deklarative Sprachen
  - Wie wird ein SQL-Statement abgearbeitet?
  - fester Ablauf
- Tabellenkalkulation, bzw. funktionale Sprachen
  - Wie werden Zellen ausgewertet?
- Netzwerke
  - Nach welchen Regeln verhalten sich die einzelnen Komponenten?
- KI Systeme
  - Was macht ein LLM?
  - Wie funktionieren Agenten-Aufrufe

Noch ein paar Worte zu Vibe Coding



Noch ein paar Worte zu Vibe Coding



Fazit

- Das Konzept hilft Gedanken zu ordnen und den Blick auf die mentalen Modelle der Lerner zu schärfen.

Literatur

Benedict Du Boulay.

Some Difficulties of Learning to Program.
Journal of Educational Computing Research, 2(1):57–73, February 1986.

Felienne Hermans, Alaaeddin Swidan, Efthimia Aivaloglou, and Marileen Smit.

Thinking out of the box: comparing metaphors for variables in programming education.
In *Proceedings of the 13th Workshop in Primary and Secondary Computing Education*, WiPSCE '18, pages 1–8, New York, NY, USA, October 2018. Association for Computing Machinery.

Linxiao Ma, John D. Ferguson, Marc Roper, Isla Ross, and Murray Wood.

Using cognitive conflict and visualisation to improve mental models held by novice programmers.
In *Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '08, page 342–346, New York, NY, USA, 2008. Association for Computing Machinery.

Orni Meerbaum-Salant, Michal Armoni, and Mordechai Ben-Ari.

Habits of programming in scratch.
In *Proceedings of the 16th Annual Joint Conference on Innovation and Technology in Computer Science Education*, ITiCSE '11, page 168–172, New York, NY, USA, 2011. Association for Computing Machinery.

Anthony Robins, Janet Rountree, and Nathan Rountree.

Learning and Teaching Programming: A Review and Discussion.
Computer Science Education, 13(2):137–172, June 2003.

Juha Sorva.

Notional machines and introductory programming education.
ACM Transactions on Computing Education, 13(2):8:1–8:31, July 2013.

Vielen Dank!

Zeit für Fragen!

Michael Brinkmeier
Institut für Informatik
Universität Osnabrück
mbrinkmeier@uni-osnabrueck.de