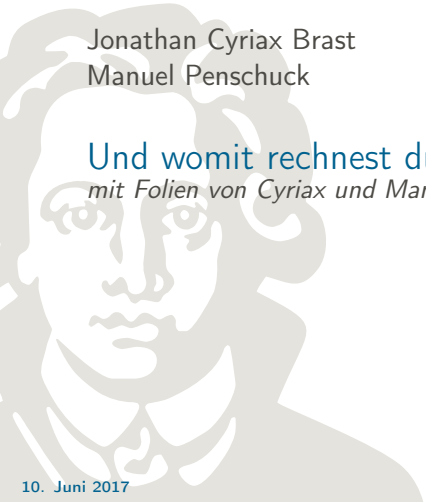
GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Jonathan Cyriax Brast
Manuel Penschuck

Und womit rechnest du so?

mit Folien von Cyriax und Manuel

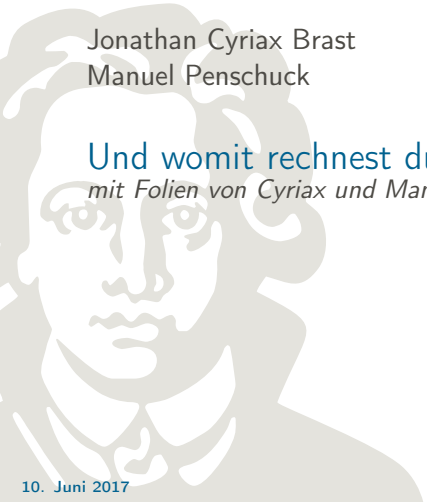


10. Juni 2017

Jonathan Cyriax Brast
Manuel Penschuck

Und womit rechnest du so?

mit Folien von Cyriax und Manuel



10. Juni 2017

Subtraktion

Minus-Rechnen

Selbsttest

Subtraktion

$$7 - 3 = \_$$

Subtraktion

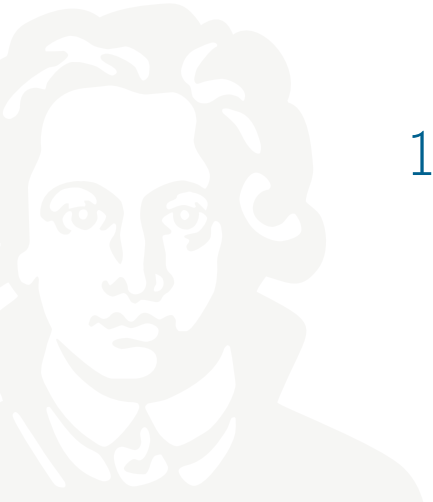
$$7 - 3 = \underline{4}$$

Und noch einer . . .

Subtraktion

$$123 - 45 = \_\_$$

Subtraktion



$$123 - 45 = \underline{78}$$

#NotMyPisa

Unäre Zahlen – römisch für Arme

Wir stellen die Zahl x durch x Einsen dar.

1_{Dez}	$=$	$1_{\text{Unär}}$	6_{Dez}	$=$	$111‘111_{\text{Unär}}$
2_{Dez}	$=$	$11_{\text{Unär}}$	7_{Dez}	$=$	$1‘111‘111_{\text{Unär}}$
3_{Dez}	$=$	$111_{\text{Unär}}$	8_{Dez}	$=$	$11‘111‘111_{\text{Unär}}$
4_{Dez}	$=$	$1‘111_{\text{Unär}}$	9_{Dez}	$=$	$111‘111‘111_{\text{Unär}}$
5_{Dez}	$=$	$11‘111_{\text{Unär}}$	10_{Dez}	$=$	$1‘111‘111‘111_{\text{Unär}}$

Wie beschreiben wir unäre Subtraktion?

$$a - b = c$$

	1	11	111	1‘111	11‘111	⋯
1		1	11	111	1‘111	⋯
11			1	11	111	⋯
111				1	11	⋯
1‘111					1	⋯
11‘111						⋯
⋮	⋮	⋮	⋮	⋮	⋮	⋱

Wie beschreiben wir unäre Subtraktion?

$$a - b = c$$

	1	11	111	1‘111	11‘111	⋯
1	0	1	11	111	1‘111	⋯
11		0	1	11	111	⋯
111			0	1	11	⋯
1‘111				0	1	⋯
11‘111					0	⋯
⋮	⋮	⋮	⋮	⋮	⋮	⋱

Wie beschreiben wir unäre Subtraktion?

$$a - b = c$$

	1	11	111	1‘111	11‘111	⋯
1		1	11	111	1‘111	⋯
11			1	11	111	⋯
111				1	11	⋯
1‘111					1	⋯
11‘111						⋯
⋮	⋮	⋮	⋮	⋮	⋮	⋱

Wie beschreiben wir unäre Subtraktion?

$$a - b = c$$

Unpraktisch für große Zahlen!

Wir brauchen einen Algorithmus

$$a - b = c$$

1. **Wenn** b keine Ziffer mehr hat,
   dann antworte $c \leftarrow a$ und **halte**

2. `Entferne` letzte Ziffer von a
   `Entferne` letzte eine Ziffer von b

3. **Springe** zu 1.

a:	111
b:	11
c:	?

Wir brauchen einen Algorithmus

$$a - b = c$$

→ 1. **Wenn** b keine Ziffer mehr hat,
dann antworte $c \leftarrow a$ und **halte**

2. `Entferne` letzte Ziffer von a
`Entferne` letzte eine Ziffer von b

3. **Springe** zu 1.

a:	111
b:	11
c:	?

Wir brauchen einen Algorithmus

$$a - b = c$$

1. **Wenn** b keine Ziffer mehr hat,
   dann antworte $c \leftarrow a$ und **halte**
2. `Entferne` letzte Ziffer von a
   `Entferne` letzte eine Ziffer von b
3. **Springe** zu 1.

a:	$11\underline{1}$
b:	$1\underline{1}$
c:	?

Wir brauchen einen Algorithmus

$$a - b = c$$

1. **Wenn** b keine Ziffer mehr hat,
   dann antworte $c \leftarrow a$ und **halte**

2. `Entferne` letzte Ziffer von a
   `Entferne` letzte eine Ziffer von b

→ 3. **Springe** zu 1.

a:	11
b:	1
c:	?

Wir brauchen einen Algorithmus

$$a - b = c$$

→ 1. **Wenn** b keine Ziffer mehr hat,
dann antworte $c \leftarrow a$ und **halte**

2. `Entferne` letzte Ziffer von a
`Entferne` letzte eine Ziffer von b

3. **Springe** zu 1.

a:	11
b:	1
c:	?

Wir brauchen einen Algorithmus

$$a - b = c$$

1. **Wenn** b keine Ziffer mehr hat,
   dann antworte $c \leftarrow a$ und **halte**
2. `Entferne` letzte Ziffer von a
   `Entferne` letzte eine Ziffer von b
3. **Springe** zu 1.

a:	$1\underline{1}$
b:	$\underline{1}$
c:	?

Wir brauchen einen Algorithmus

$$a - b = c$$

1. **Wenn** b keine Ziffer mehr hat,
   dann antworte $c \leftarrow a$ und **halte**
2. `Entferne` letzte Ziffer von a
   `Entferne` letzte eine Ziffer von b
3. **Springe** zu 1.

a: 1

b:

c: ?

Wir brauchen einen Algorithmus

$$a - b = c$$

→ 1. **Wenn** b keine Ziffer mehr hat,
dann antworte $c \leftarrow a$ und **halte**

2. `Entferne` letzte Ziffer von a
`Entferne` letzte eine Ziffer von b

3. **Springe** zu 1.

a: 1

b:

c: ?

Wir brauchen einen Algorithmus

$$a - b = c$$

1. **Wenn** b keine Ziffer mehr hat,
→ **dann antworte** $c \leftarrow a$ und **halte**

2. `Entferne` letzte Ziffer von a
`Entferne` letzte eine Ziffer von b

3. **Springe** zu 1.

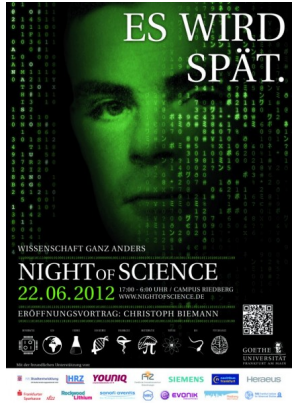
a: 1

b:

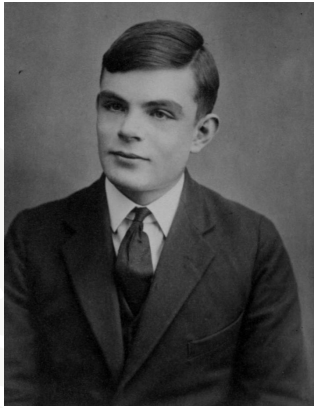
c: 1

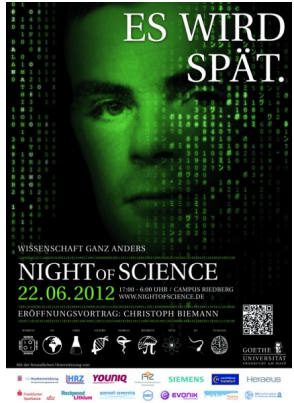
Halt, Stop!

Turing Maschine



Alan M. Turing
1912 – 1954





Alan M. Turing
1912 – 1954

230 A. M. TURING [Nov. 12,

ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM

By A. M. TURING.

[Received 28 May, 1936.—Read 12 November, 1936.]

The "computable" numbers may be described briefly as the real numbers whose expressions as a decimal are calculable by finite means. Although the subject of this paper is ostensibly the computable *numbers*, it is almost equally easy to define and investigate computable functions of an integral variable or a real or computable variable, computable predicates, and so forth. The fundamental problems involved are, however, the same in each case, and I have chosen the computable numbers for explicit treatment as involving the least cumbrous technique. I hope shortly to give an account of the relations of the computable numbers, functions, and so forth to one another. This will include a development of the theory of functions of a real variable expressed in terms of computable numbers. According to my definition, a number is computable if its decimal can be written down by a machine.

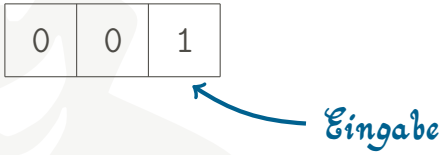
In §§ 9, 10 I give some arguments with the intention of showing that the computable numbers include all numbers which could naturally be regarded as computable. In particular, I show that certain large classes of numbers are computable. They include, for instance, the real parts of all algebraic numbers, the real parts of the zeros of the Bessel functions, the numbers π, e, etc. The computable numbers do not, however, include all definable numbers, and an example is given of a definable number which is not computable.

Although the class of computable numbers is so great, and in many ways similar to the class of real numbers, it is nevertheless enumerable. In § 8 I examine certain arguments which would seem to prove the contrary. By the correct application of one of these arguments, conclusions are reached which are superficially similar to those of Gödel†. These results

† Gödel, "Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme, I", *Monatshefte Math. Phys.*, 38 (1931), 173–198.

"On Computable Numbers, With an Application to the Entscheidungsproblem"
[A. Turing, 1936/37]

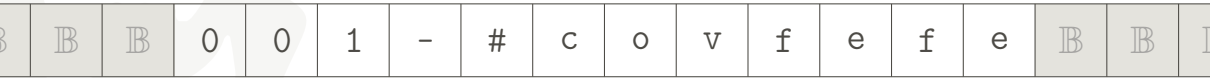
Turing Maschine



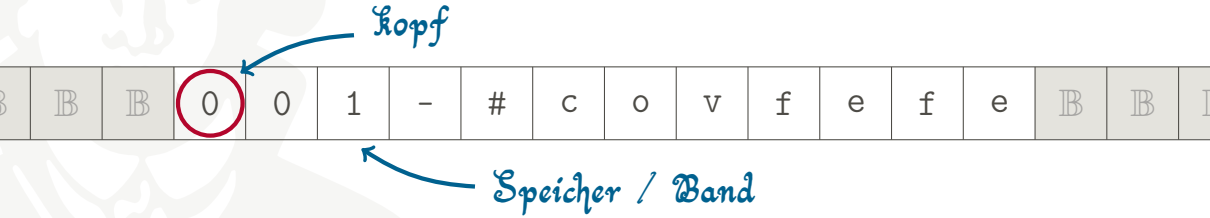
Turing Maschine

Turing Maschine

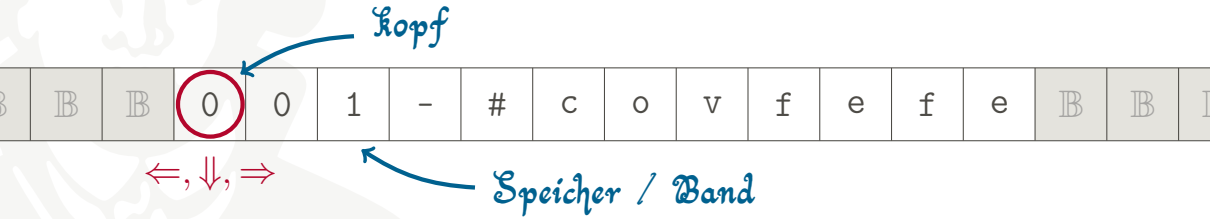
Turing Maschine



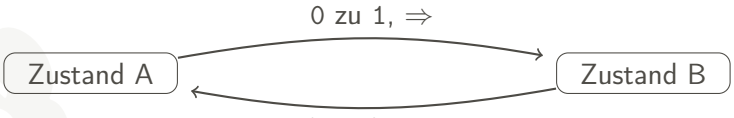
Turing Maschine

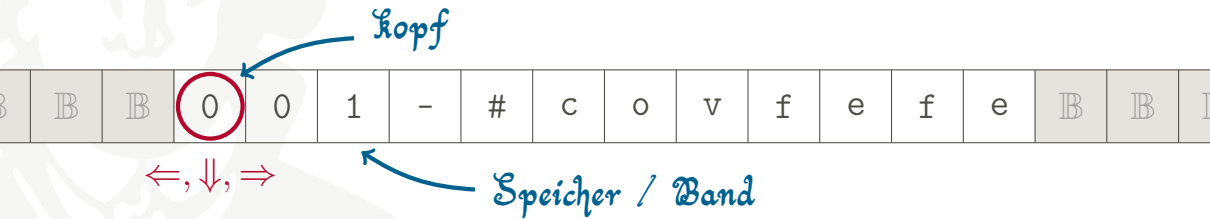


Turing Maschine

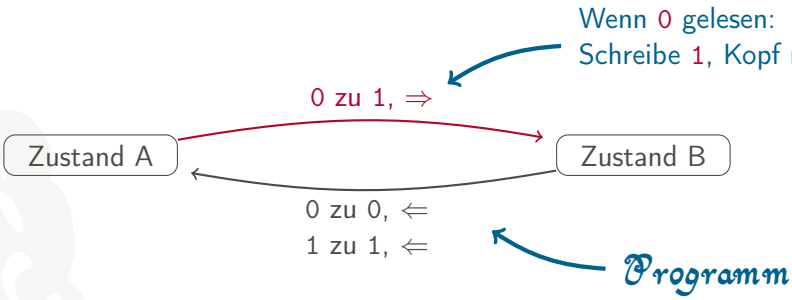


Turing Maschine

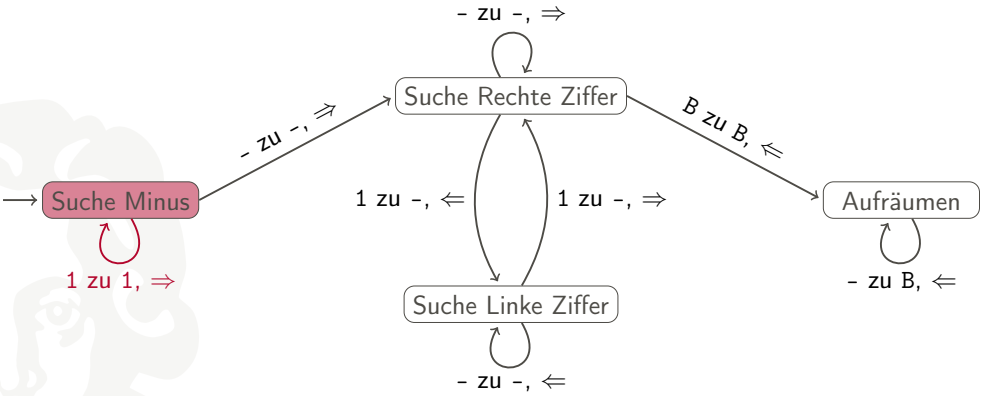




Turing Maschine

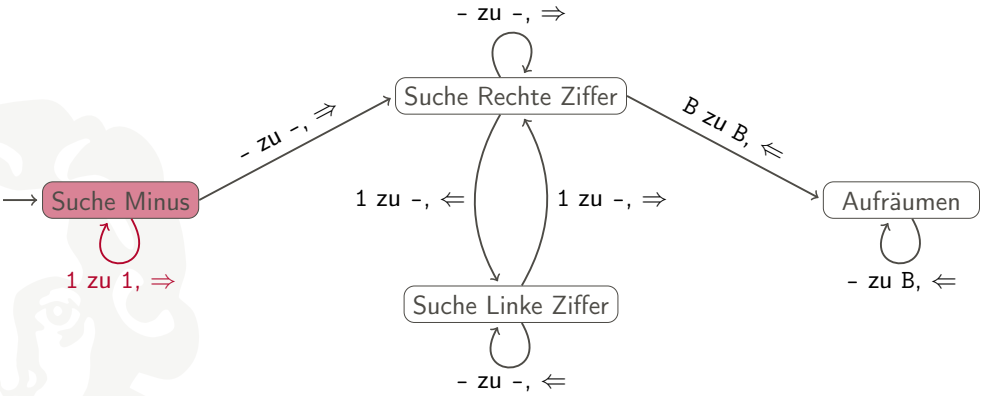


Subtraktion: $111 - 11 = ?$



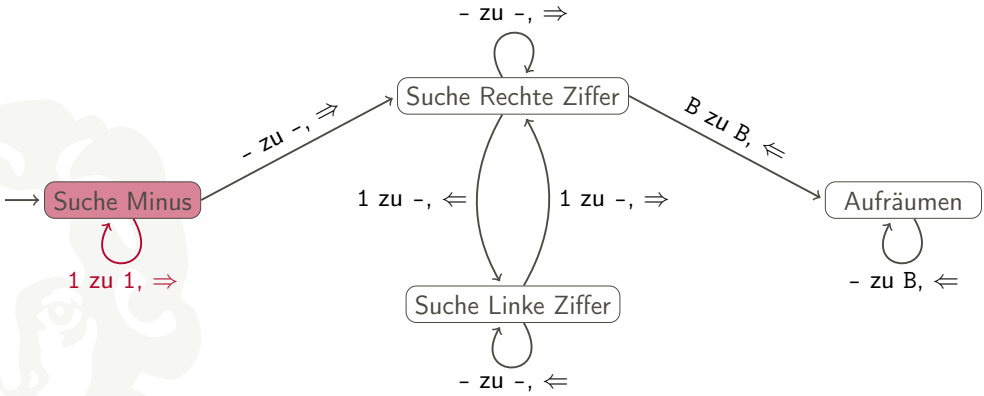
…	𝔹	𝔹	𝔹	𝔹	𝔹	1	1	1	-	1	1	𝔹	𝔹	𝔹	𝔹	𝔹	…

Subtraktion: $111 - 11 = ?$



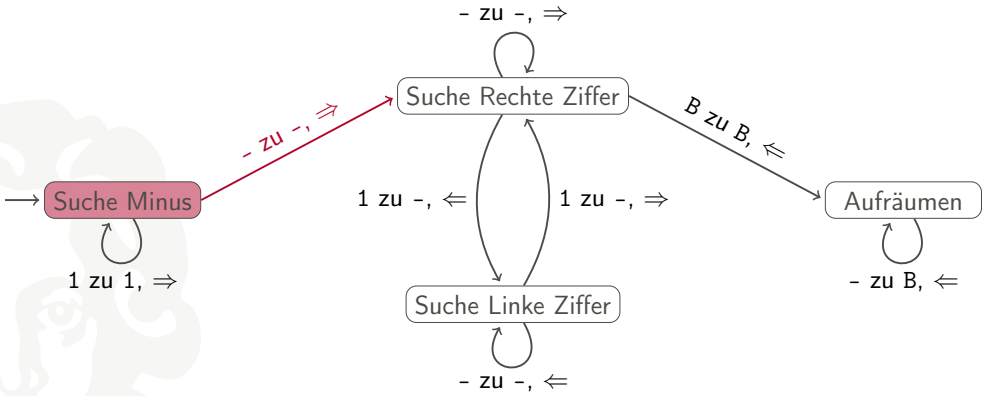
𝔹	𝔹	𝔹	𝔹	𝔹	𝔹	1	1	1	-	1	1	𝔹	𝔹	𝔹	𝔹	𝔹	𝔹

Subtraktion: $111 - 11 = ?$



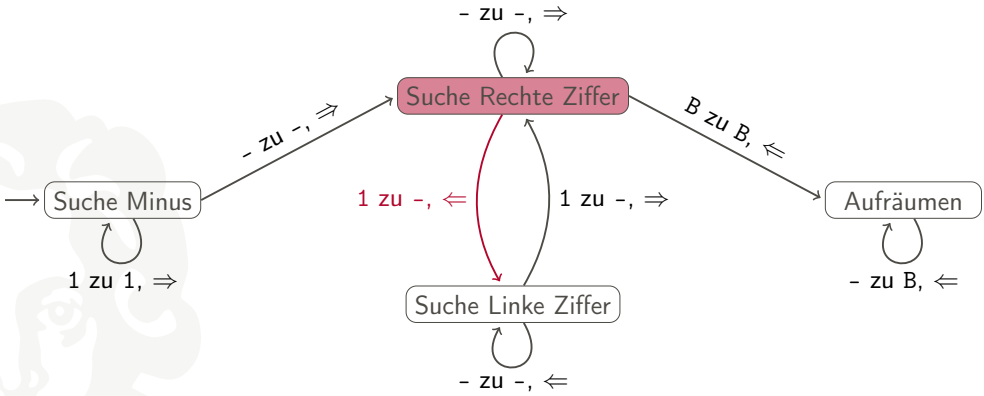
𝔹	𝔹	𝔹	𝔹	𝔹	𝔹	1	1	1	-	1	1	𝔹	𝔹	𝔹	𝔹	𝔹	𝔹

Subtraktion: $111 - 11 = ?$

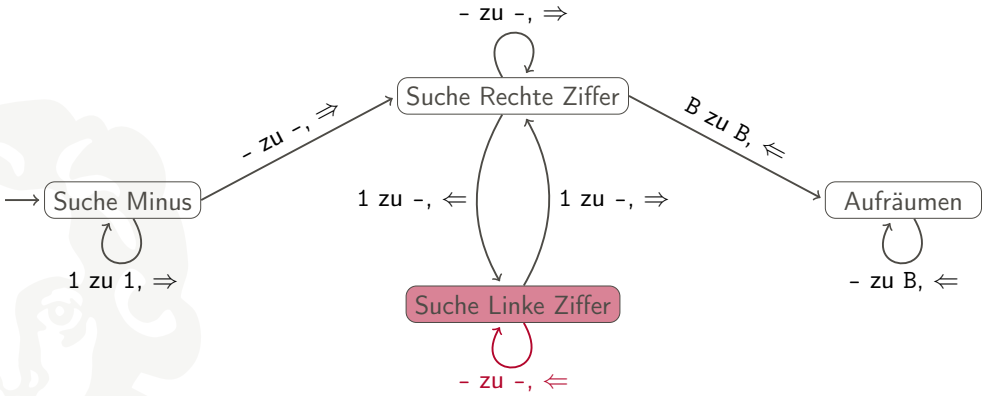


𝔹	𝔹	𝔹	𝔹	𝔹	𝔹	1	1	1	–	1	1	𝔹	𝔹	𝔹	𝔹	𝔹	𝔹

Subtraktion: $111 - 11 = ?$

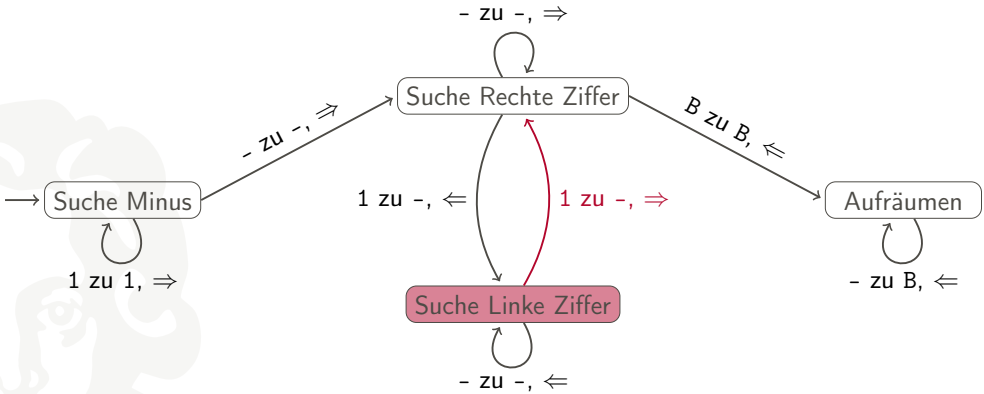


Subtraktion: $111 - 11 = ?$



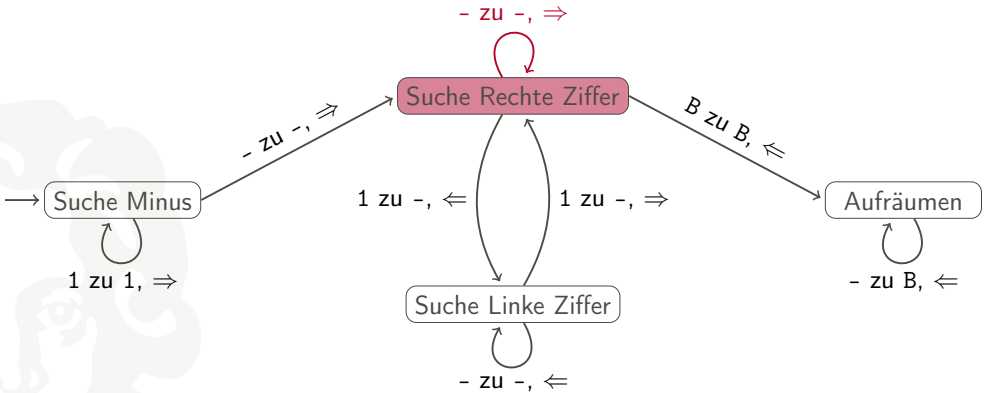
…	𝔹	𝔹	𝔹	𝔹	𝔹	1	1	1	–	–	1	𝔹	𝔹	𝔹	𝔹	𝔹	…

Subtraktion: $111 - 11 = ?$



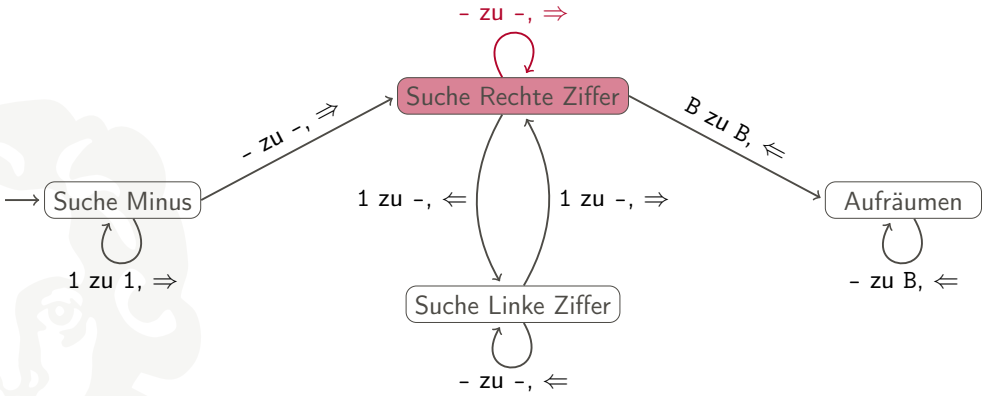
𝔹	𝔹	𝔹	𝔹	𝔹	𝔹	1	1	1	-	-	1	𝔹	𝔹	𝔹	𝔹	𝔹	𝔹

Subtraktion: $111 - 11 = ?$

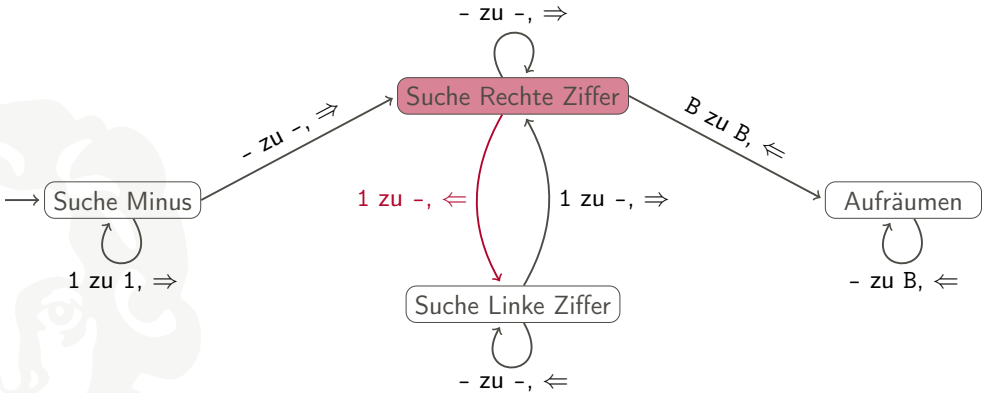


𝔹	𝔹	𝔹	𝔹	𝔹	𝔹	1	1	-	-	-	1	𝔹	𝔹	𝔹	𝔹	𝔹	

Subtraktion: $111 - 11 = ?$

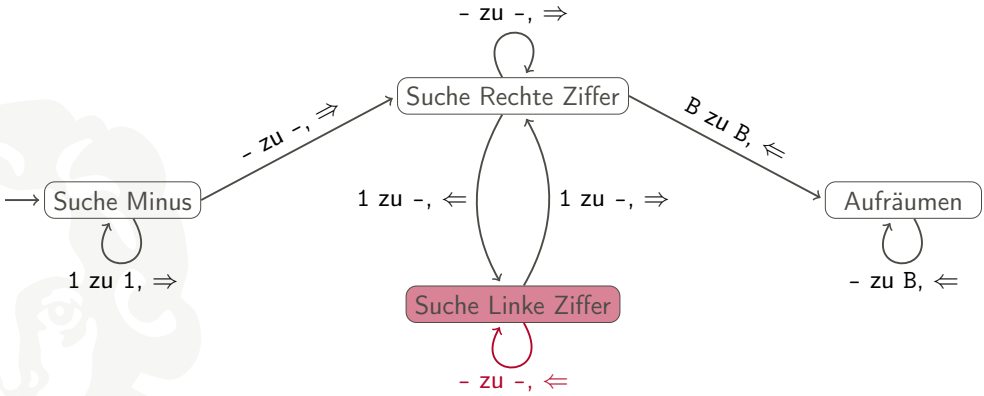


Subtraktion: $111 - 11 = ?$

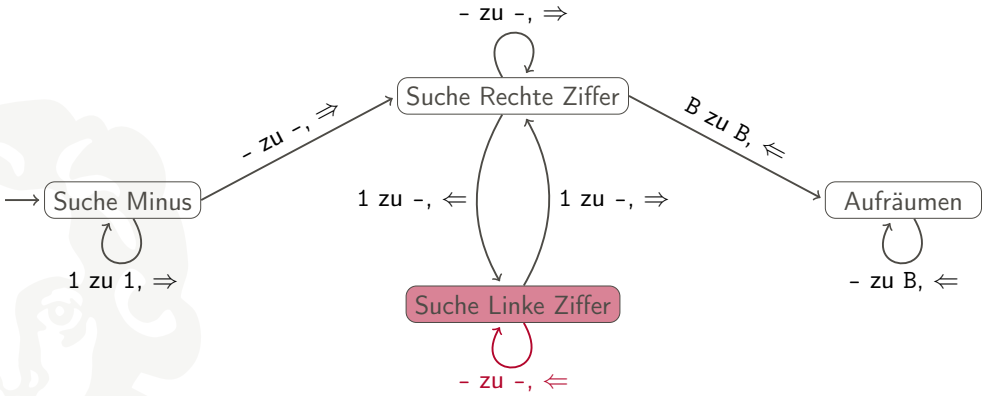


𝔹	𝔹	𝔹	𝔹	𝔹	𝔹	1	1	-	-	-	1	𝔹	𝔹	𝔹	𝔹	𝔹	

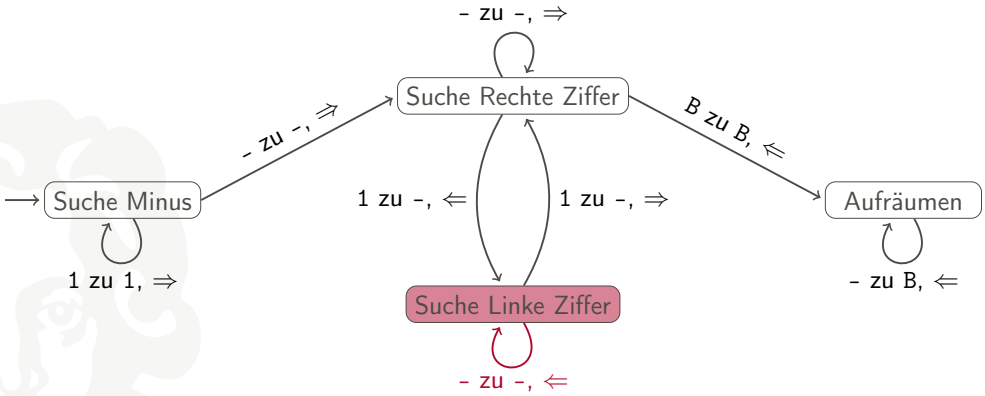
Subtraktion: $111 - 11 = ?$



Subtraktion: $111 - 11 = ?$

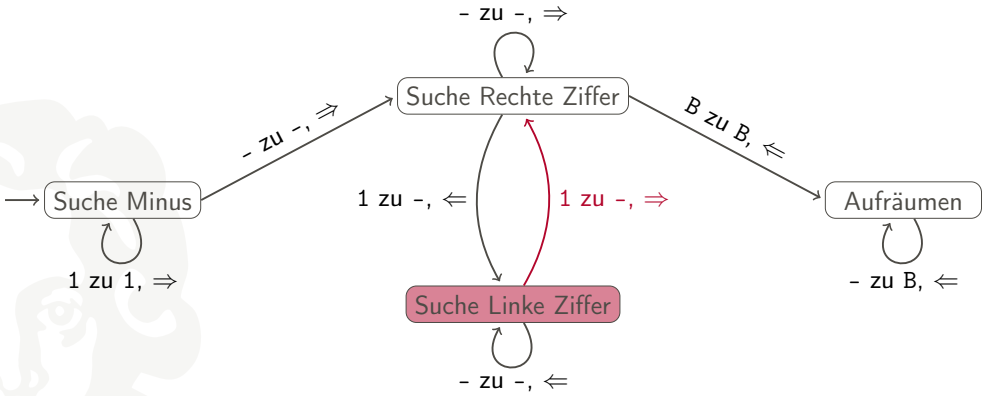


Subtraktion: $111 - 11 = ?$



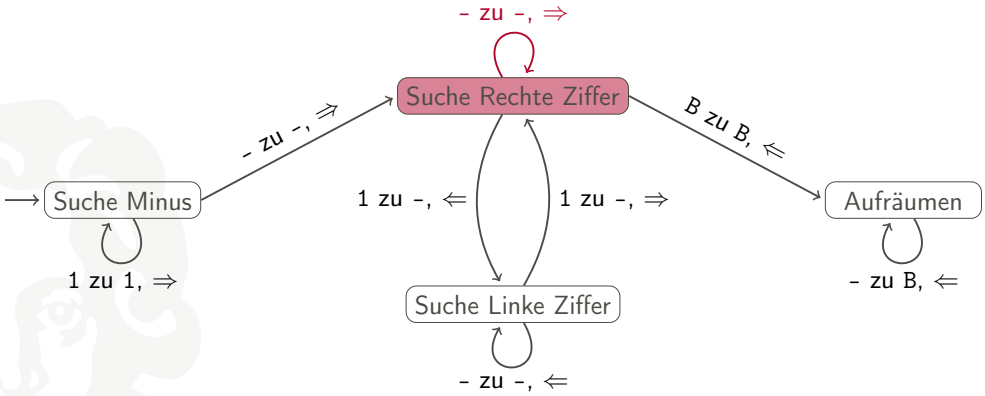
B	B	B	B	B	B	1	1	-	-	-	-	B	B	B	B	B	B

Subtraktion: $111 - 11 = ?$

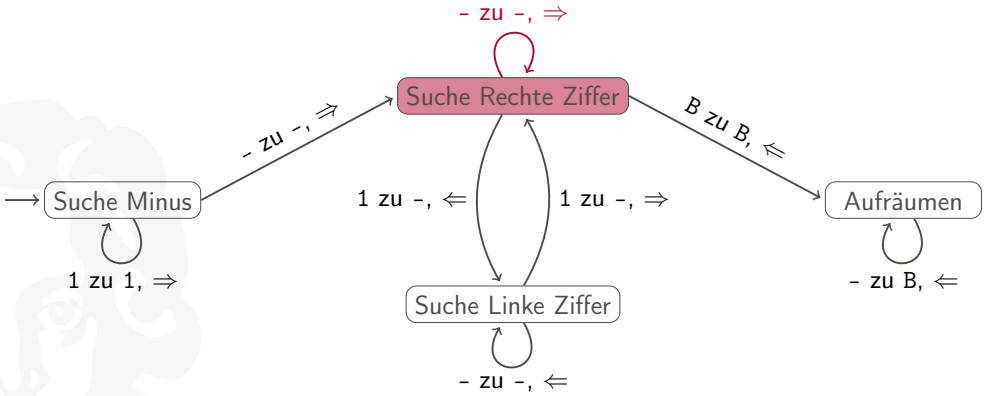


𝔹	𝔹	𝔹	𝔹	𝔹	𝔹	1	1	-	-	-	-	𝔹	𝔹	𝔹	𝔹	𝔹	𝔹

Subtraktion: $111 - 11 = ?$

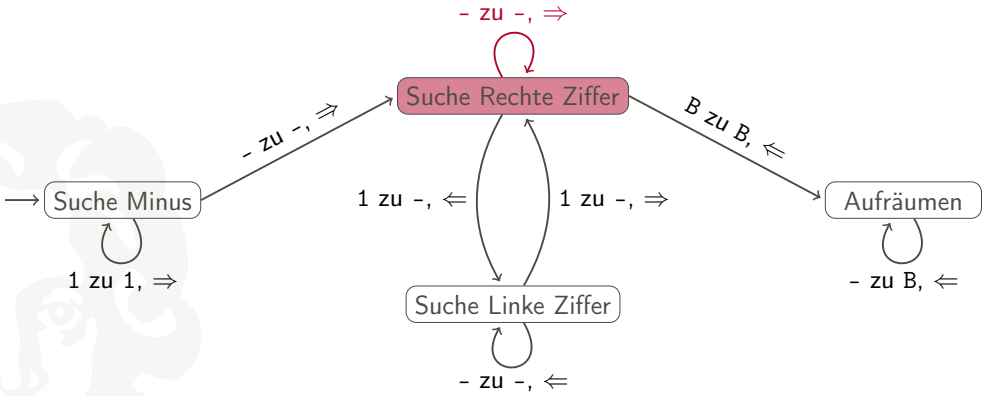


Subtraktion: $111 - 11 = ?$

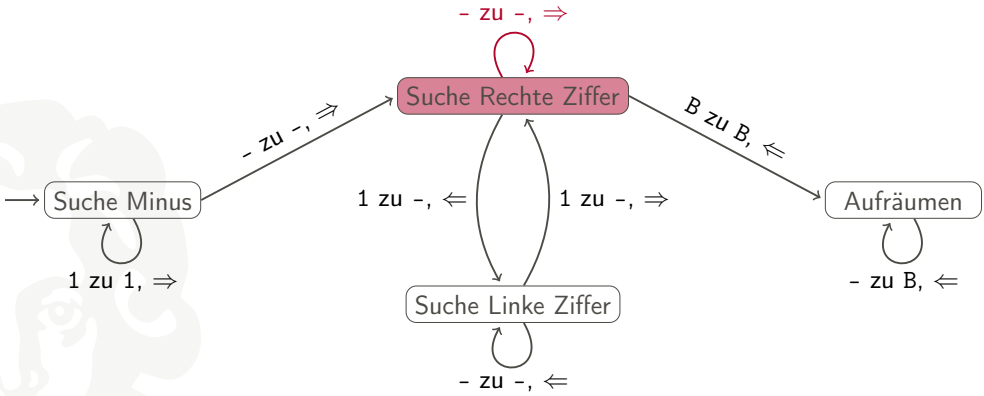


𝔹	𝔹	𝔹	𝔹	𝔹	𝔹	1	–	–	–	–	–	𝔹	𝔹	𝔹	𝔹	𝔹	𝔹

Subtraktion: $111 - 11 = ?$

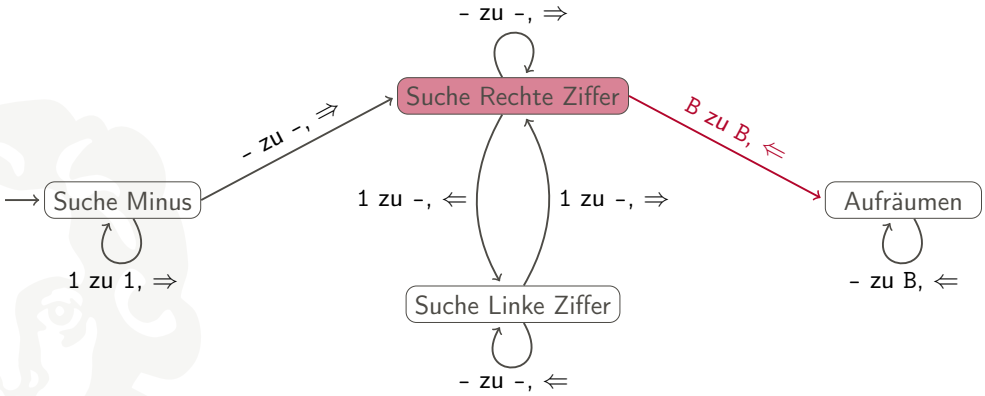


Subtraktion: $111 - 11 = ?$



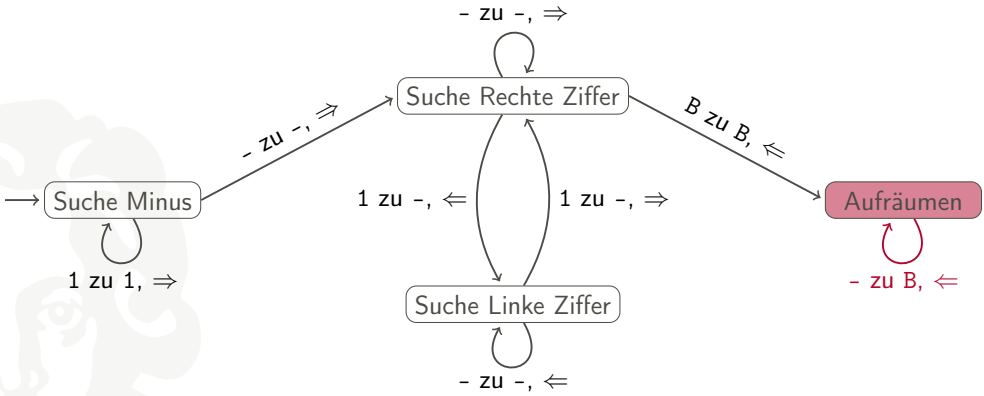
…	𝔹	𝔹	𝔹	𝔹	𝔹	1	–	–	–	–	–	𝔹	𝔹	𝔹	𝔹	𝔹	…

Subtraktion: $111 - 11 = ?$



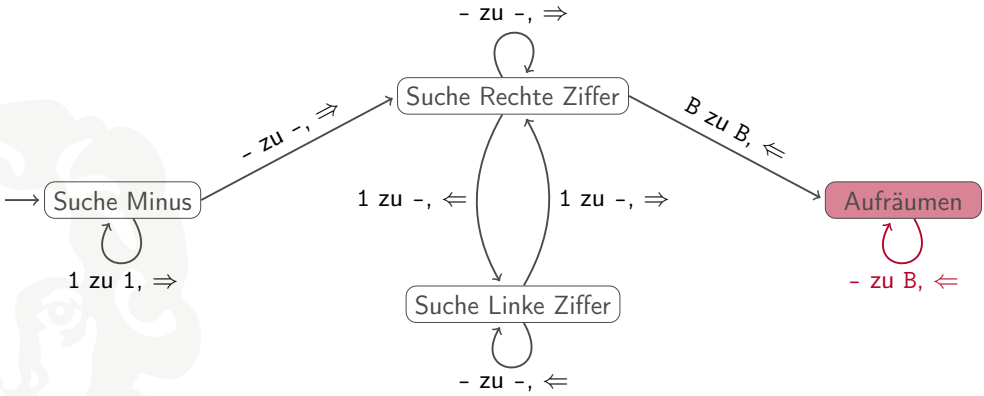
…	𝔹	𝔹	𝔹	𝔹	𝔹	1	–	–	–	–	–	𝔹	𝔹	𝔹	𝔹	𝔹	…

Subtraktion: $111 - 11 = ?$



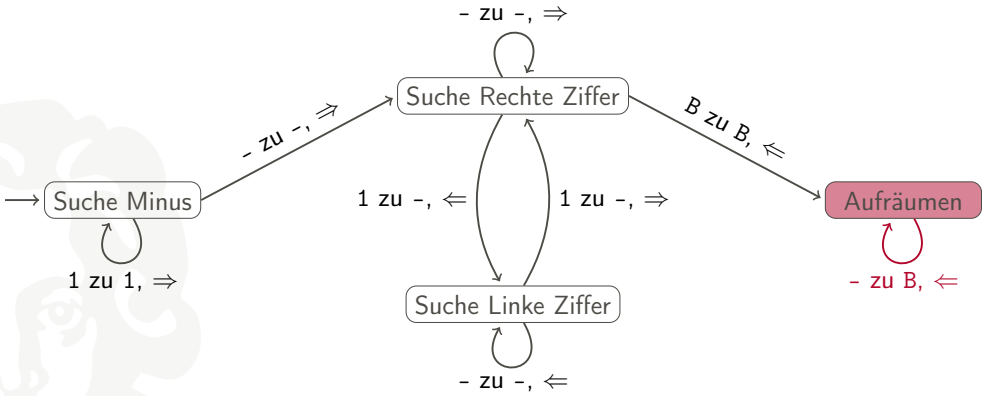
…	𝔹	𝔹	𝔹	𝔹	𝔹	1	–	–	–	–	–	𝔹	𝔹	𝔹	𝔹	𝔹	…

Subtraktion: $111 - 11 = ?$



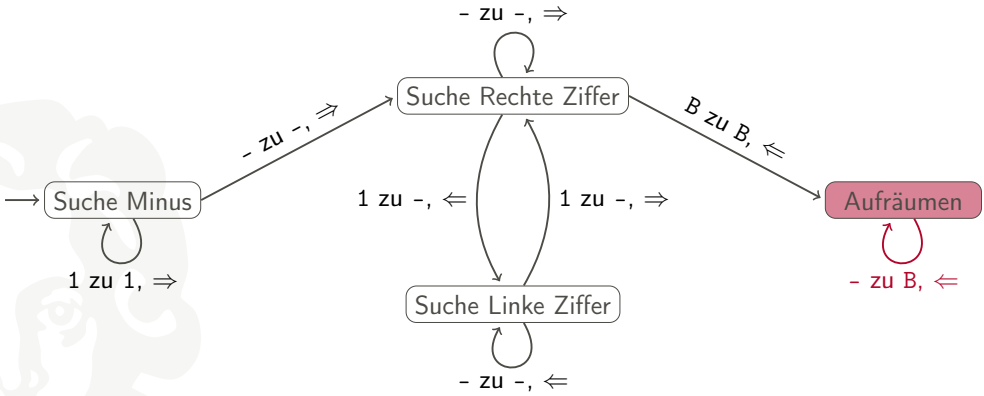
…	𝔹	𝔹	𝔹	𝔹	𝔹	1	–	–	–	–	𝔹	𝔹	𝔹	𝔹	𝔹	𝔹	…

Subtraktion: $111 - 11 = ?$



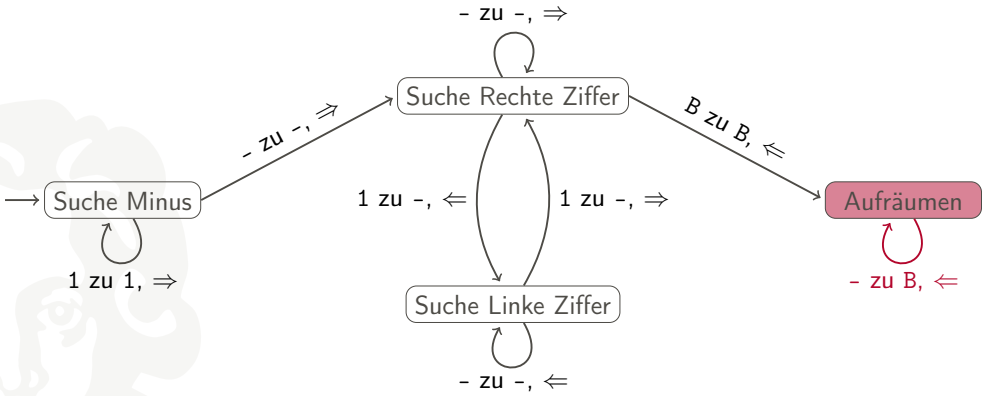
…	𝔹	𝔹	𝔹	𝔹	𝔹	1	–	–	–	𝔹	𝔹	𝔹	𝔹	𝔹	𝔹	𝔹	…

Subtraktion: $111 - 11 = ?$



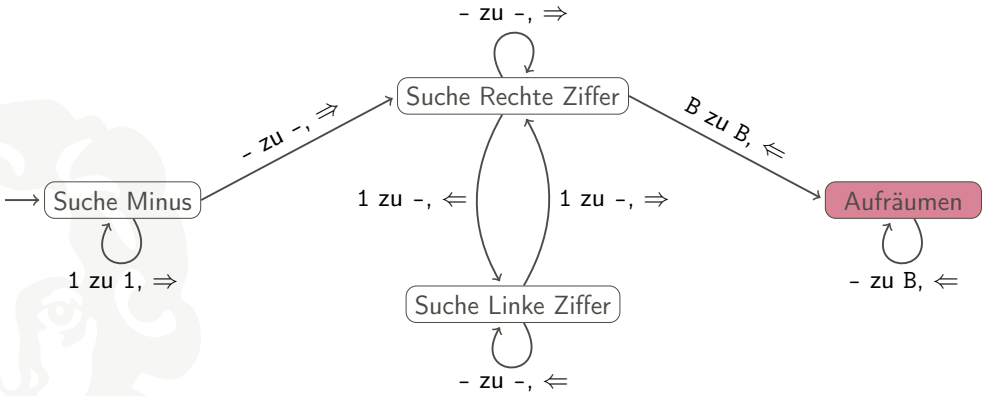
…	𝔹	𝔹	𝔹	𝔹	𝔹	1	–	–	𝔹	𝔹	𝔹	𝔹	𝔹	𝔹	𝔹	𝔹	…

Subtraktion: $111 - 11 = ?$



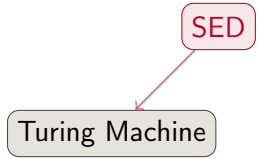
𝔹	𝔹	𝔹	𝔹	𝔹	𝔹	1	–	𝔹	𝔹	𝔹	𝔹	𝔹	𝔹	𝔹	𝔹	𝔹	𝔹

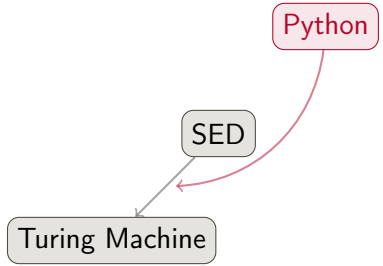
Subtraktion: $111 - 11 = ?$

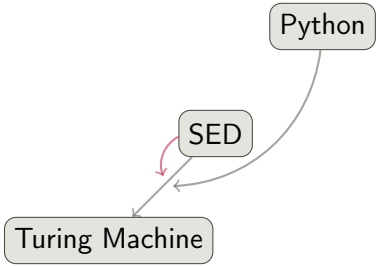


Turing Machine

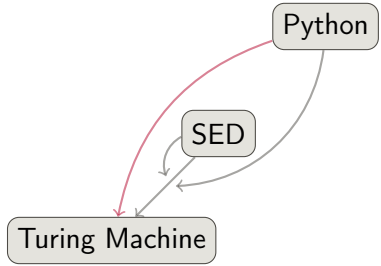
```
s
/ZUSTAND-ALT:(.*)(.)@X(.*)
/ZUSTAND-NEU:\1@\2Y\3/
```




Python
SED
Turing Machine

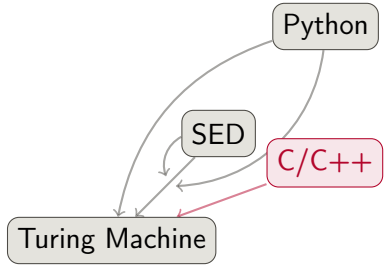

Python
SED
Turing Machine

```
if state=="A":
    if band.read()=="0":
        band.write(1)
        band.goLeft()
        state = "B"
    elif band.read()=="1":
        ...
elif state=="B":
    if band.read()=="0":
    ...
```

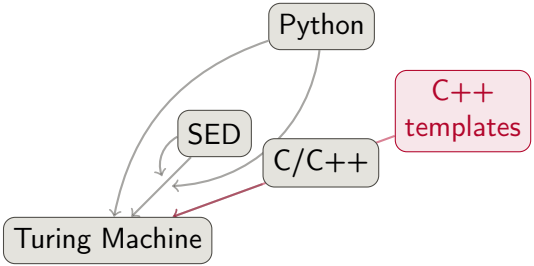


C++

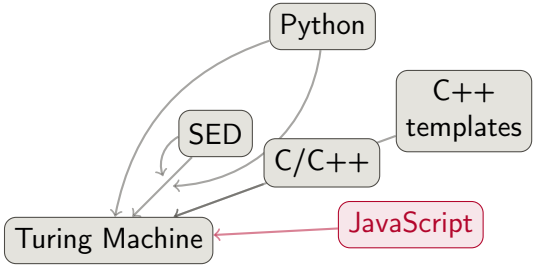
```
if (state=='A') {
    if(band.read()=='0') {
        band.write(1)
        band.goLeft()
        state = "B"
    } else if(band.read()=='1') {
        ...}
} else if(state=="B") {
    if(band.read()=="0") {
    ...}}
```

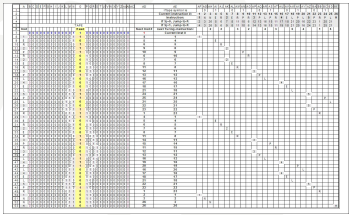


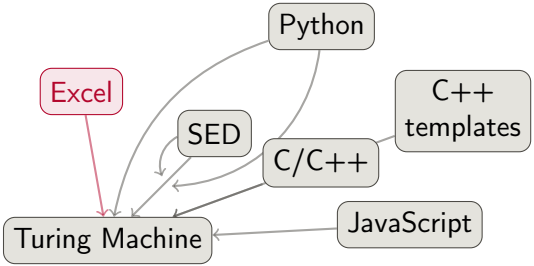

GCC

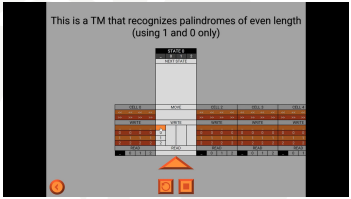

Python
C++
templates
SED
C/C++
Turing Machine

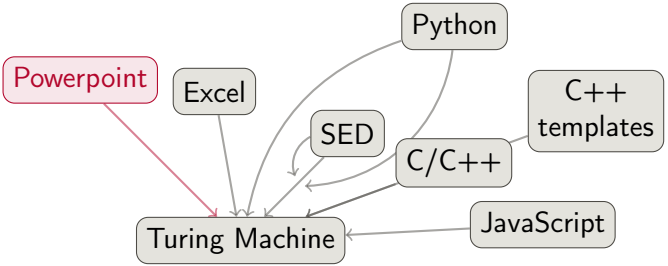

JavaScript

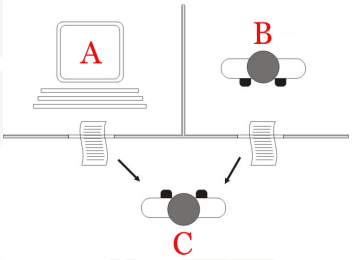

Python
C++
templates
SED
C/C++
Turing Machine
JavaScript

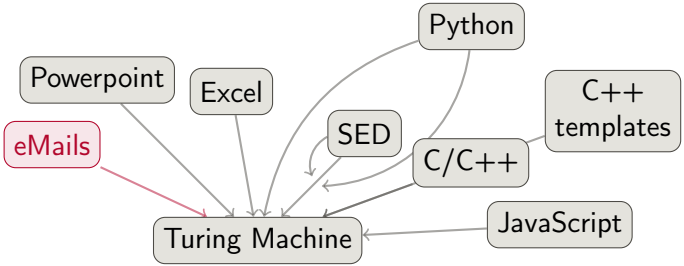


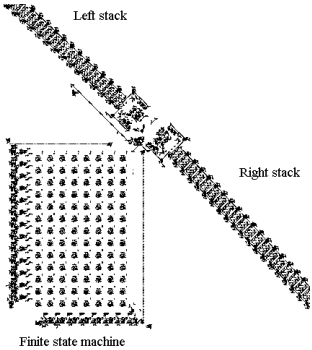

Python
Excel
SED
C++
templates
C/C++
JavaScript
Turing Machine

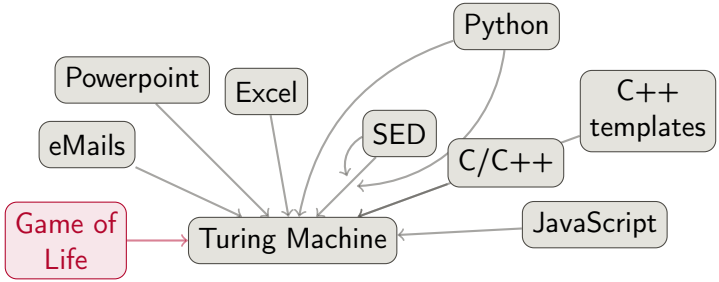

This is a TM that recognizes palindromes of even length
(using 1 and 0 only)

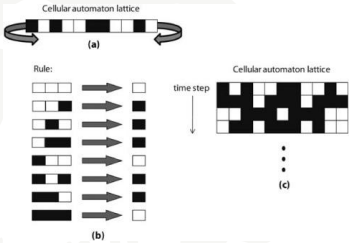

Python
Powerpoint
Excel
C++
templates
SED
C/C++
JavaScript
Turing Machine

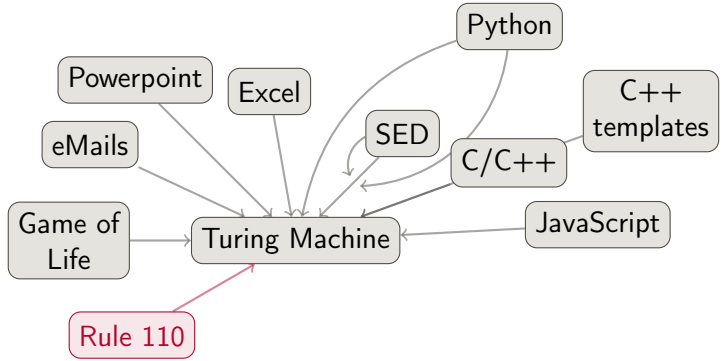

A
B
C

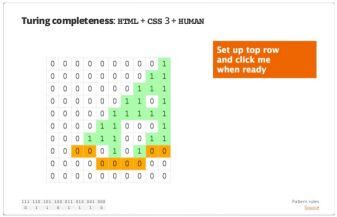

Python
Powerpoint
Excel
C++
templates
SED
eMails
C/C++
JavaScript
Turing Machine

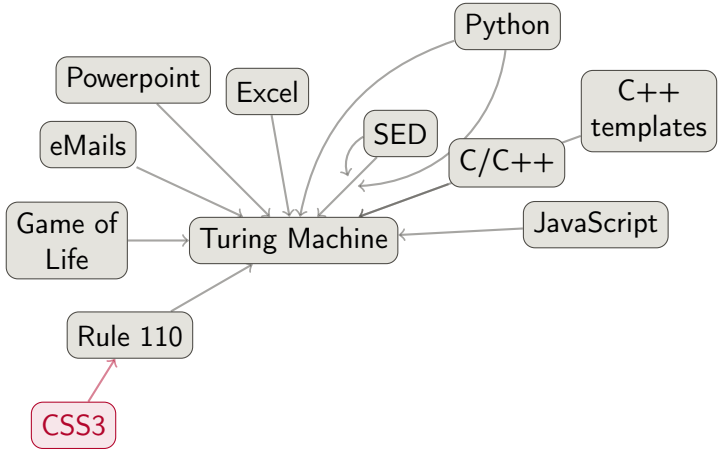

Left stack
Right stack
Finite state machine

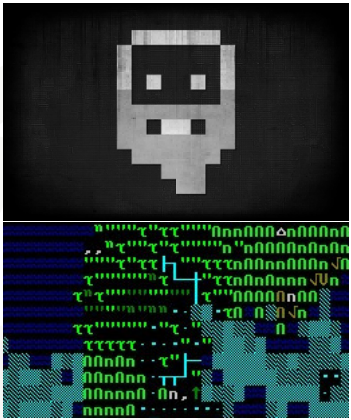

Python
Powerpoint
Excel
C++
templates
SED
eMails
C/C++
Game of
Life
Turing Machine
JavaScript

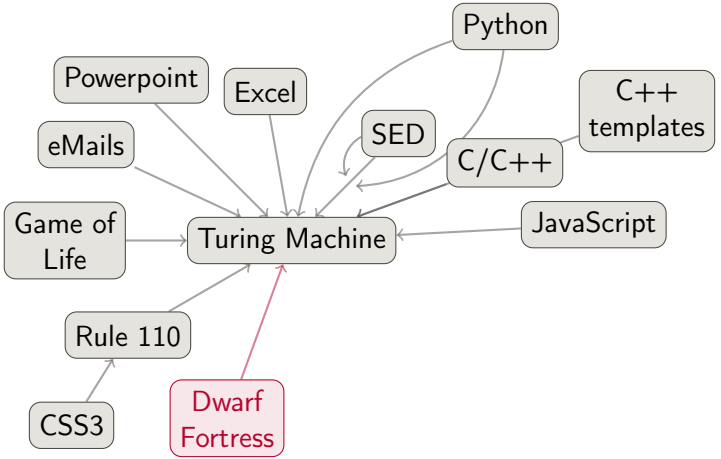

Cellular automaton lattice
(a)
Rule:
(b)
Cellular automaton lattice
time step
(c)

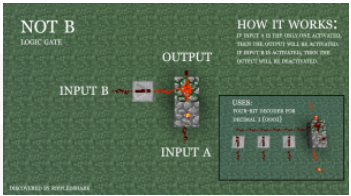

Python
Powerpoint
Excel
C++ templates
eMails
SED
C/C++
Game of Life
Turing Machine
JavaScript
Rule 110

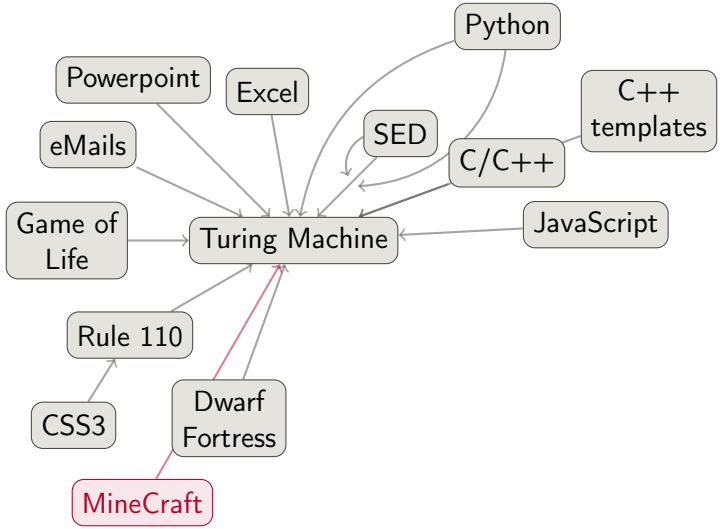

Turing completeness: HTML + CSS 3 + HUMAN
Set up top row and click me when ready
Pattern rules
Source

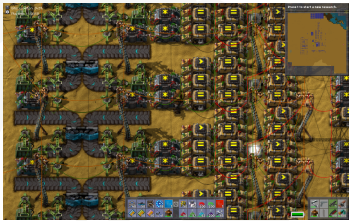

Python
Powerpoint
Excel
C++ templates
SED
eMails
C/C++
Game of Life
Turing Machine
JavaScript
Rule 110
CSS3

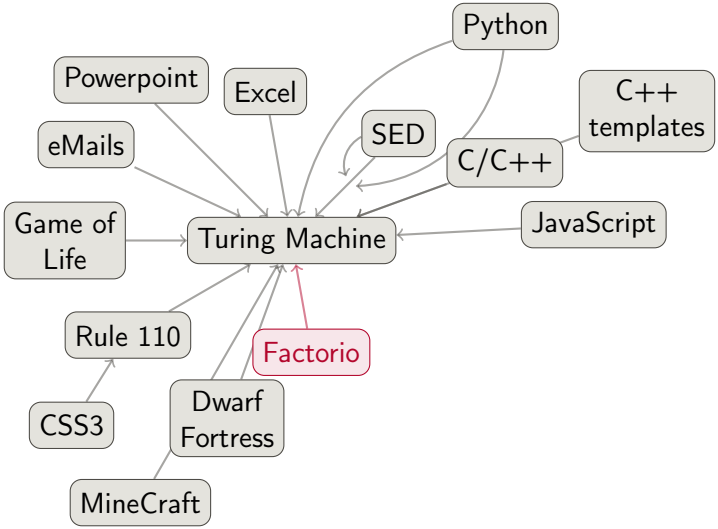


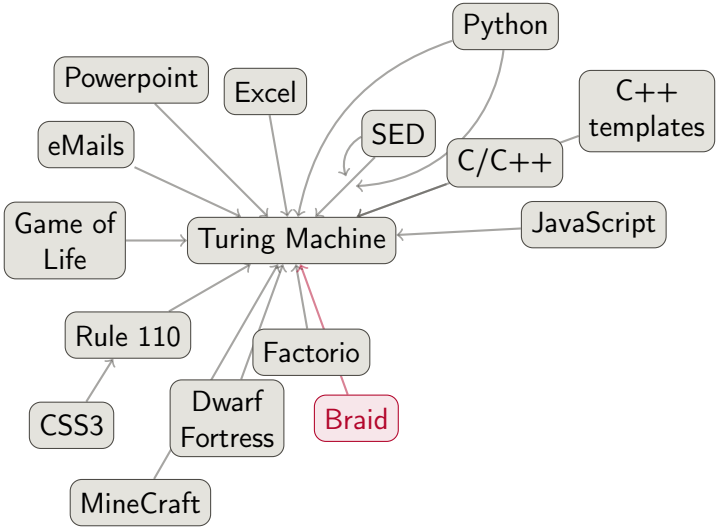

Python
Powerpoint
Excel
C++ templates
SED
eMails
C/C++
Game of Life
Turing Machine
JavaScript
Rule 110
CSS3
Dwarf Fortress

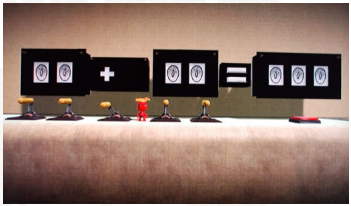

NOT B
LOGIC GATE
OUTPUT
INPUT B
INPUT A
HOW IT WORKS:
USES:

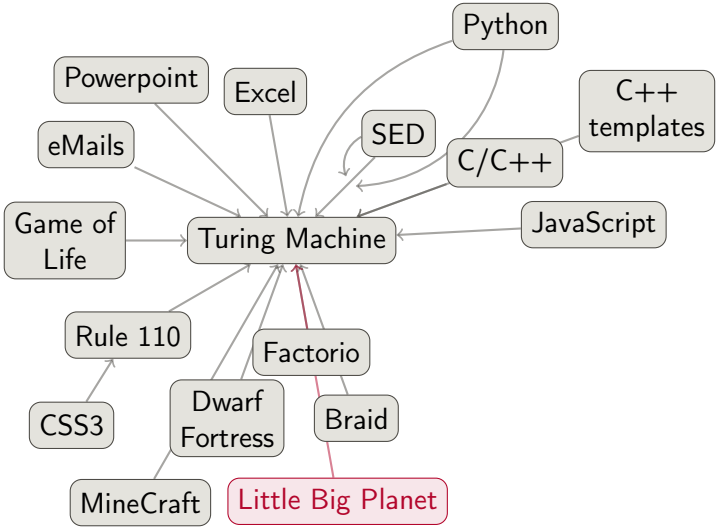

Python
Powerpoint
Excel
C++ templates
SED
eMails
C/C++
Game of Life
Turing Machine
JavaScript
Rule 110
Dwarf Fortress
CSS3
MineCraft

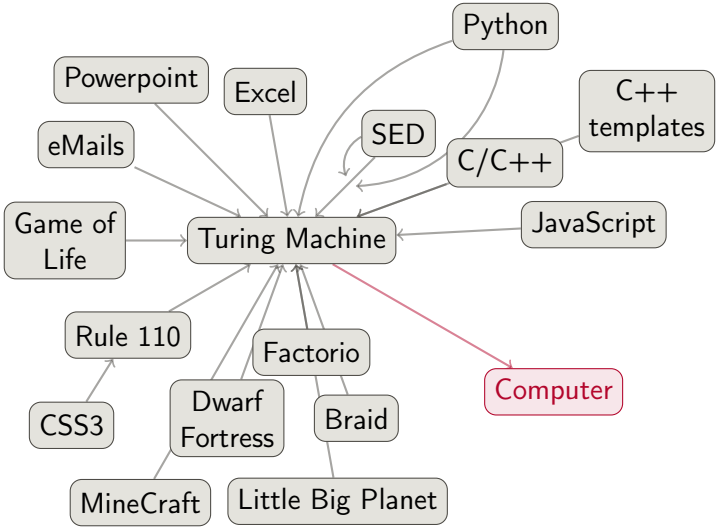


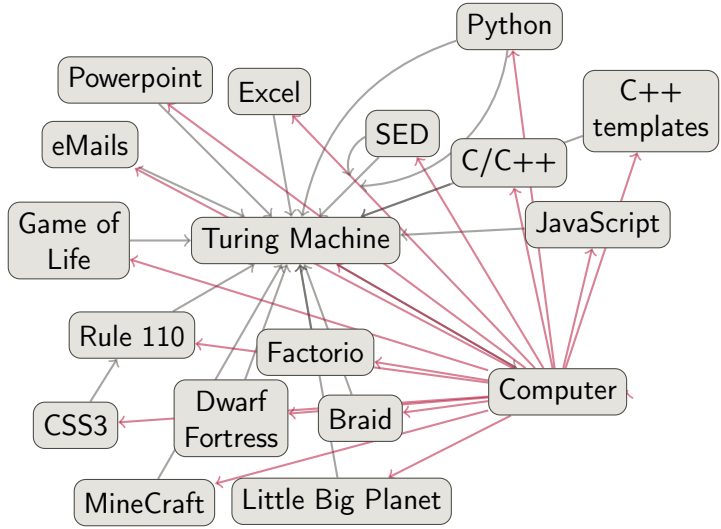

Python
Powerpoint
Excel
C++ templates
SED
eMails
C/C++
Game of Life
Turing Machine
JavaScript
Rule 110
Factorio
CSS3
Dwarf Fortress
MineCraft

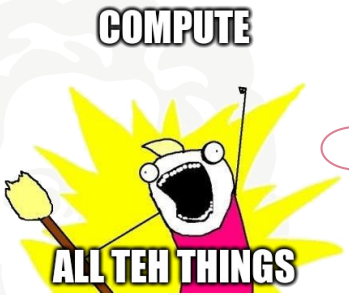

Python
Powerpoint
Excel
C++ templates
SED
eMails
C/C++
Game of Life
Turing Machine
JavaScript
Rule 110
Factorio
CSS3
Dwarf Fortress
Braid
MineCraft

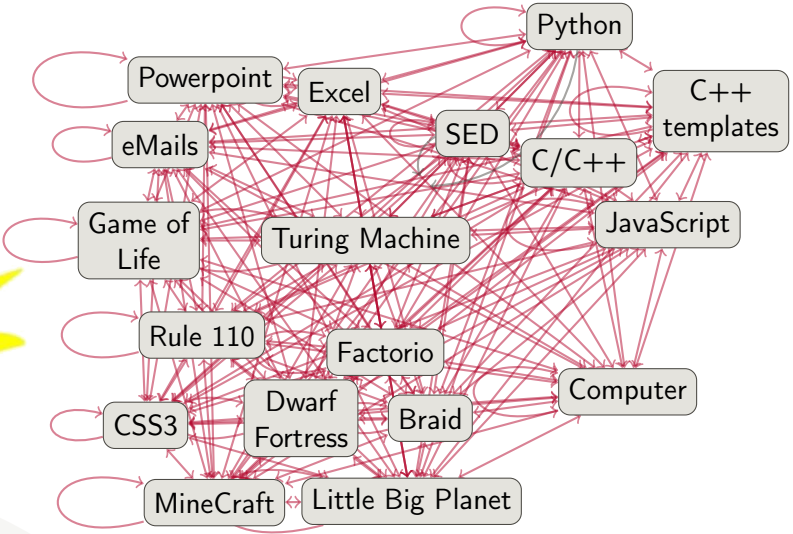



Python
Powerpoint
Excel
C++ templates
SED
eMails
C/C++
Game of Life
Turing Machine
JavaScript
Rule 110
Factorio
CSS3
Dwarf Fortress
Braid
MineCraft
Little Big Planet

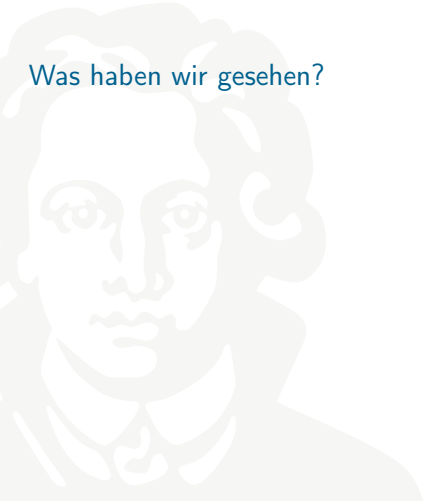

Python
Powerpoint
Excel
C++ templates
SED
eMails
C/C++
Game of Life
Turing Machine
JavaScript
Rule 110
Factorio
Computer
CSS3
Dwarf Fortress
Braid
MineCraft
Little Big Planet


Python
Powerpoint
Excel
C++
templates
SED
eMails
C/C++
Game of
Life
Turing Machine
JavaScript
Rule 110
Factorio
Computer
Dwarf
Fortress
CSS3
Braid
MineCraft
Little Big Planet


COMPUTE
ALL TEH THINGS

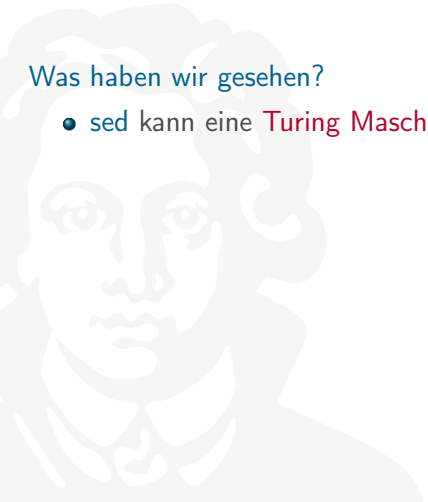

Python
Powerpoint
Excel
C++ templates
eMails
SED
C/C++
Game of Life
Turing Machine
JavaScript
Rule 110
Factorio
Computer
CSS3
Dwarf Fortress
Braid
MineCraft
Little Big Planet

Warum sind Turing Maschinen *wirklich* nützlich?



Was haben wir gesehen?

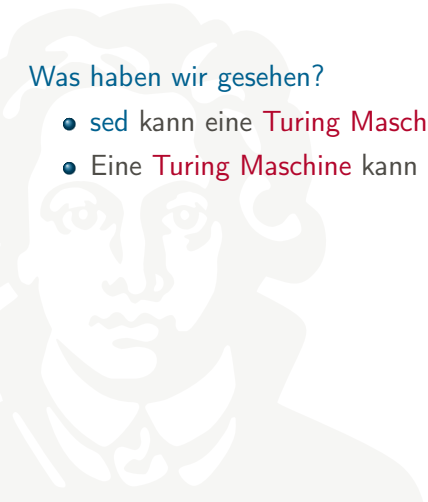
Warum sind Turing Maschinen *wirklich* nützlich?



Was haben wir gesehen?

- sed kann eine Turing Maschine ausführen

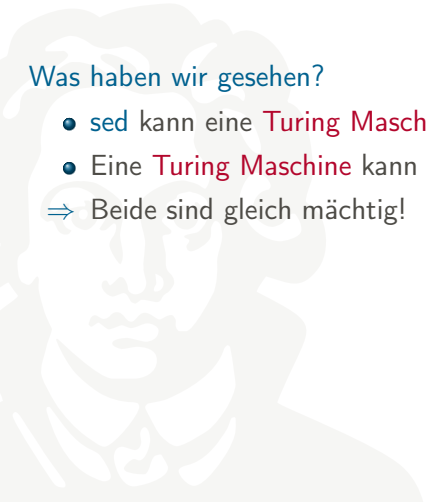
Warum sind Turing Maschinen *wirklich* nützlich?



Was haben wir gesehen?

- sed kann eine Turing Maschine ausführen
- Eine Turing Maschine kann sed ausführen

Warum sind Turing Maschinen *wirklich* nützlich?

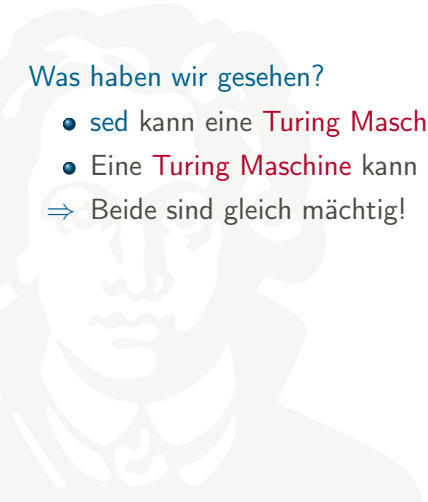


Was haben wir gesehen?

- sed kann eine Turing Maschine ausführen
- Eine Turing Maschine kann sed ausführen

$\Rightarrow$ Beide sind gleich mächtig!

Warum sind Turing Maschinen *wirklich* nützlich?



Was haben wir gesehen?

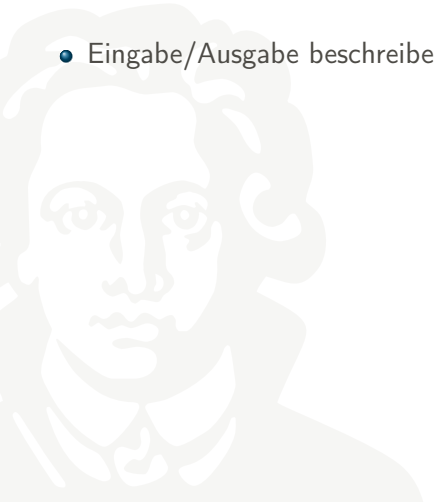
- sed kann eine Turing Maschine ausführen
- Eine Turing Maschine kann sed ausführen

$\Rightarrow$ Beide sind gleich mächtig!

} **Turing Vollständig**

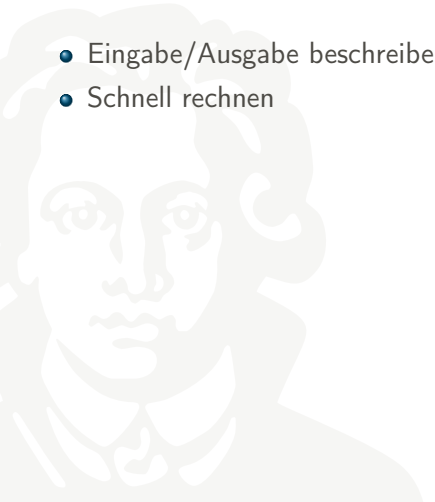
Was können Turing Maschinen **nicht**?

- Eingabe/Ausgabe beschreiben



Was können Turing Maschinen **nicht**?

- Eingabe/Ausgabe beschreiben
- Schnell rechnen



Was können Turing Maschinen **nicht**?

- Eingabe/Ausgabe beschreiben
- Schnell rechnen

- Für ein beliebiges Programm P entscheiden,
  - ...ob P fertig wird

Was können Turing Maschinen **nicht**?

- Eingabe/Ausgabe beschreiben
- Schnell rechnen

- Für ein beliebiges Programm P entscheiden,
  - ...ob P fertig wird
  - ...ob P eine spezielle Zeile ausführt

Was können Turing Maschinen **nicht**?

- Eingabe/Ausgabe beschreiben
- Schnell rechnen

- Für ein beliebiges Programm P entscheiden,
  - ...ob P fertig wird
  - ...ob P eine spezielle Zeile ausführt
  - ...ob P eine nicht triviale Eigenschaft erfüllt

Was ist nicht Turing-Vollständig?

- Reine Daten
  - (Werte-)Tabellen
  - HTML, XML
  - Rechenschieber

Was ist nicht Turing-Vollständig?

- Reine Daten
  - (Werte-)Tabellen
  - HTML, XML
  - Rechenschieber
- Endliche Automaten
  - Turing Maschine mit schreibgeschüztem Band
  - Turing Maschine, die den Kopf nur nach rechts bewegen darf
  - Reguläre Sprachen
  - sed ohne Sprünge

Was ist nicht Turing-Vollständig?

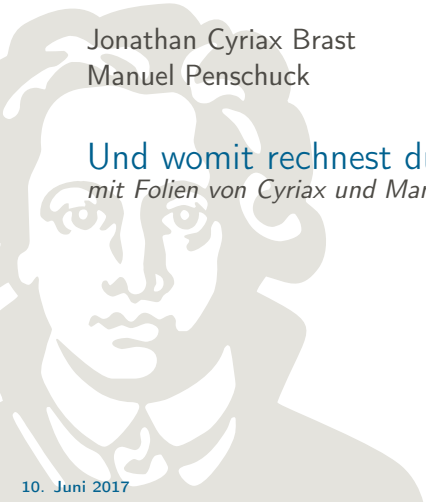
- Reine Daten
  - (Werte-)Tabellen
  - HTML, XML
  - Rechenschieber
- Endliche Automaten
  - Turing Maschine mit schreibgeschüztem Band
  - Turing Maschine, die den Kopf nur nach rechts bewegen darf
  - Reguläre Sprachen
  - sed ohne Sprünge
- ...

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Jonathan Cyriax Brast
Manuel Penschuck

Und womit rechnest du so?

mit Folien von Cyriax und Manuel



http://nos.manuel.jetzt

10. Juni 2017