Besser Programmieren mit Mathematik

Michael Sperber

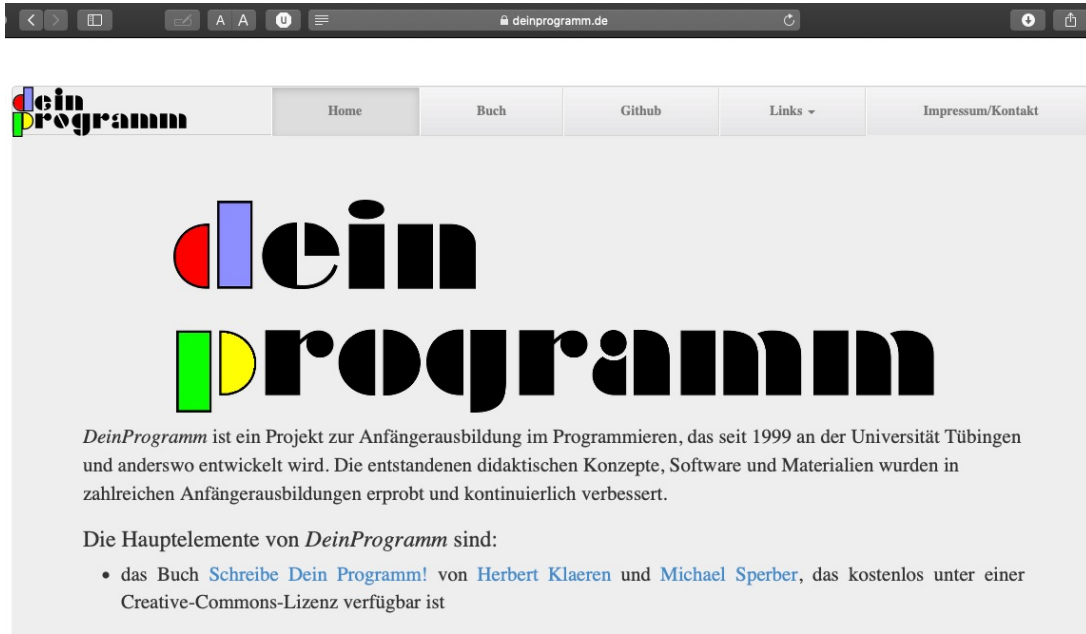
@sperbsen@discuss.systems

Michael Sperber

- Promotion 2001 Uni Tübingen (Bühnenbeleuchtung)
- Geschäftsführer Active Group
- Individualsoftware mit funktionaler Programmierung
- Ausbildung Softwarearchitektur

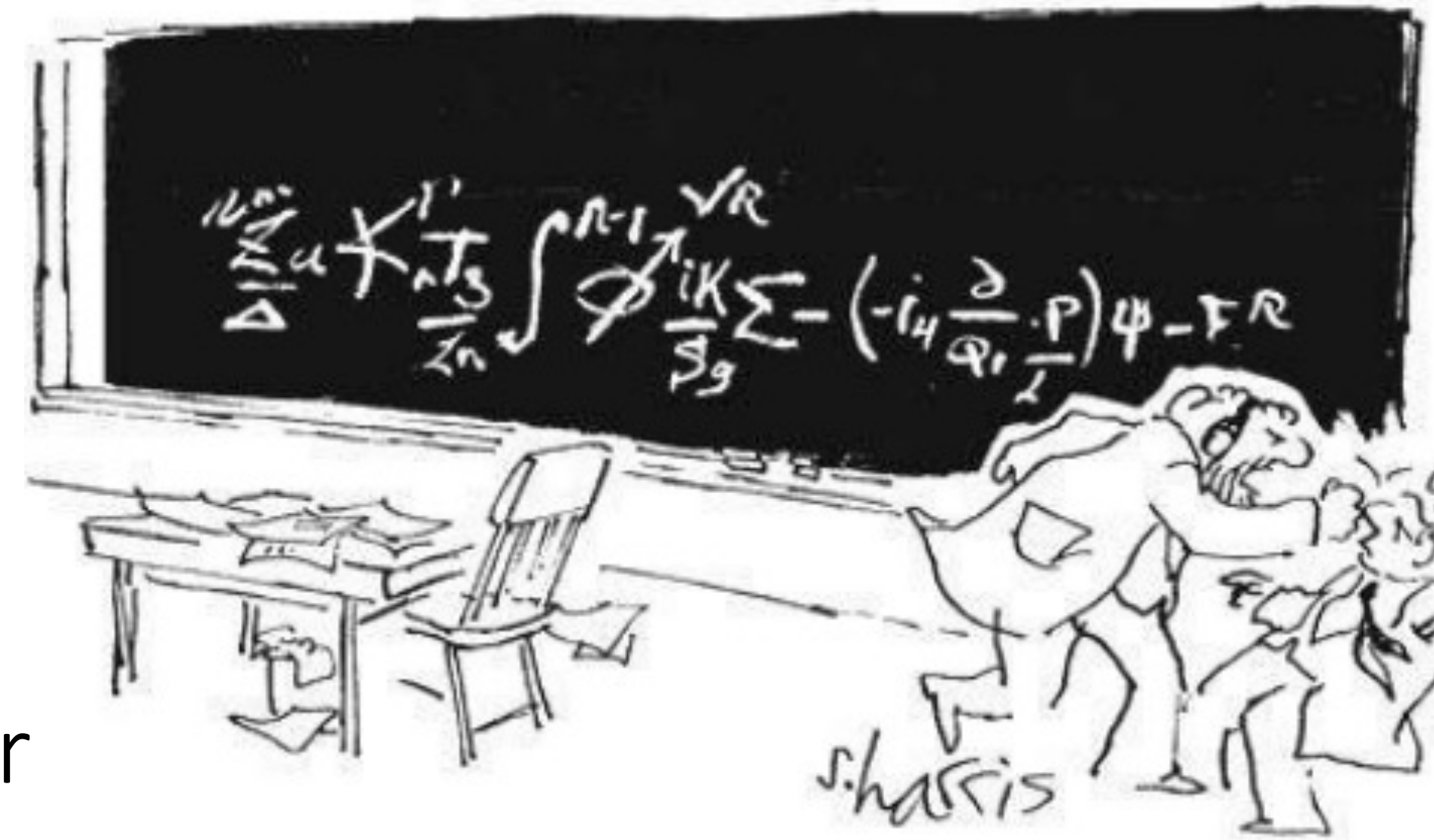
www.deinprogramm.de

- Anfängerausbildung Programmierung:



@ctive group

Mathe für richtig



"You want proof? I'll give you proof!"

@ctive group

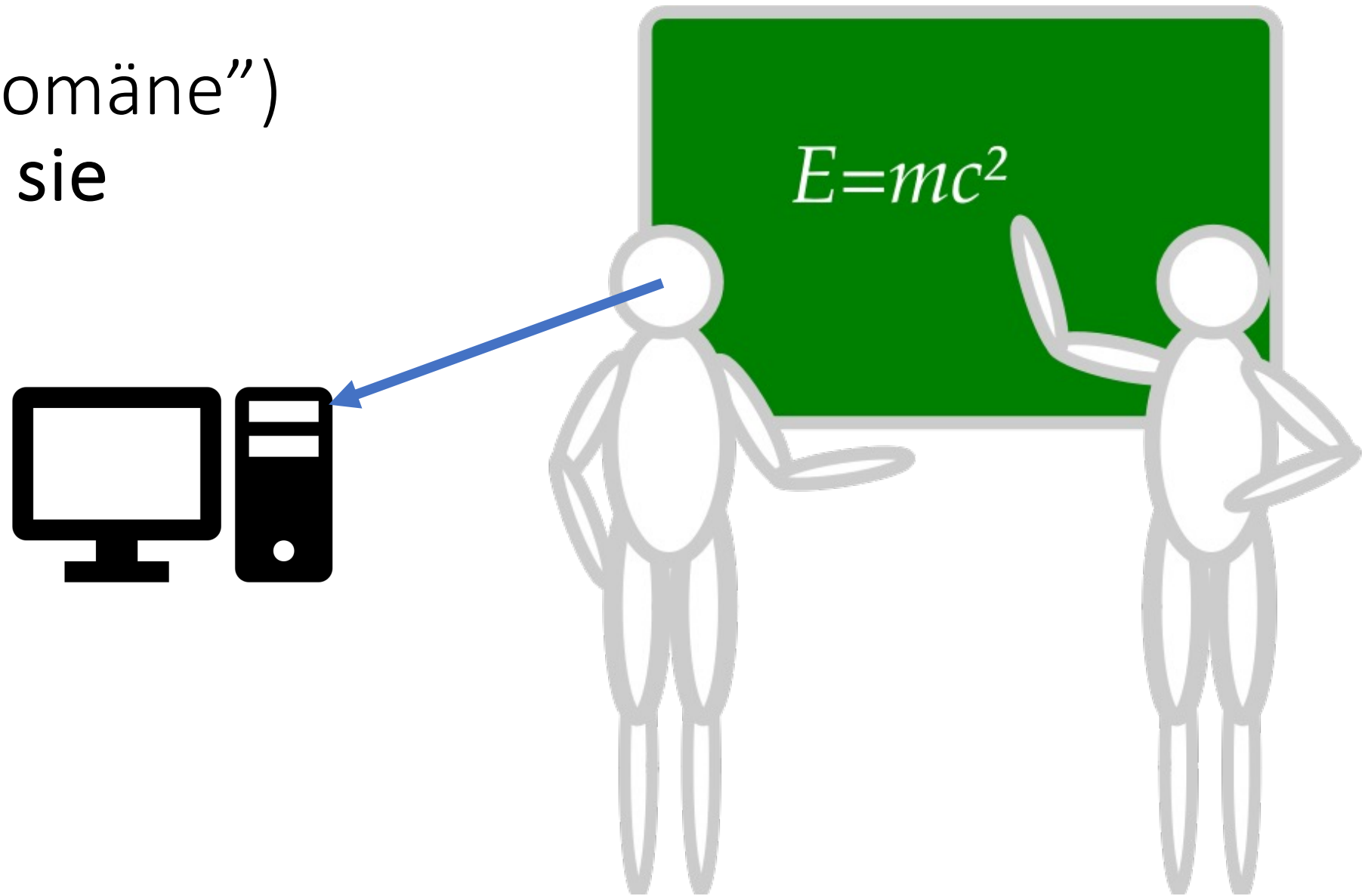
Sidney Harris: You Want Proof? I'll Give You Proof!: More Cartoons, W.H. Freeman, 1990

Mathe als Sprache

“Die Philosophie steht in diesem großen Buch geschrieben, dem Universum, das unserem Blick ständig offen liegt. Aber das Buch ist nicht zu verstehen, wenn man nicht zuvor die Sprache erlernt und sich mit den Buchstaben vertraut gemacht hat, in denen es geschrieben ist. Es ist in der Sprache der Mathematik geschrieben, und deren Buchstaben sind Kreise, Dreiecke und andere geometrische Figuren, ohne die es dem Menschen unmöglich ist, ein einziges Wort davon zu verstehen; ohne diese irrt man in einem dunklen Labyrinth herum.”

Galileo: Il Saggiatore, 1623

Reden über
die Welt (“Domäne”)
und wie wir sie
sehen

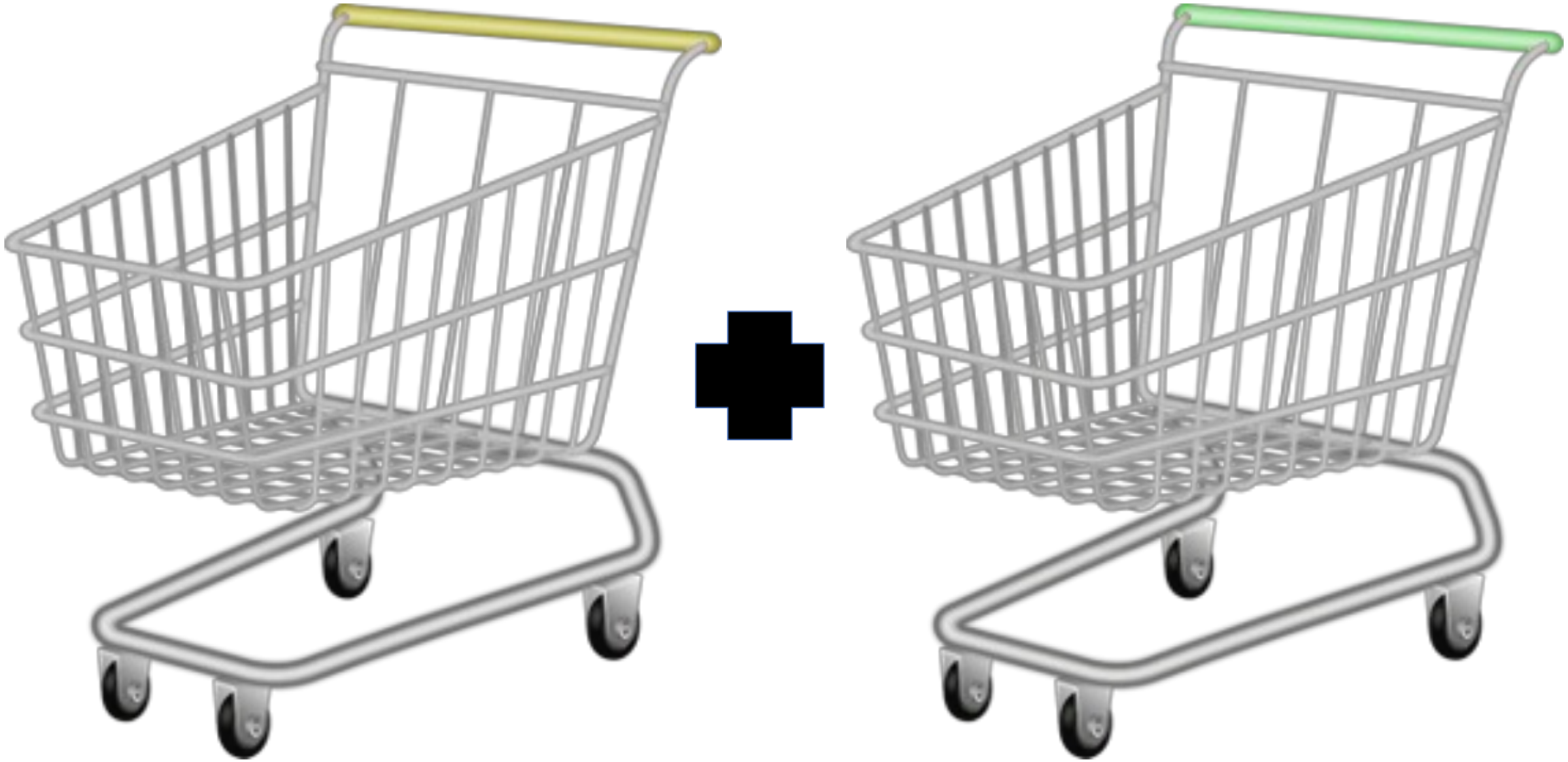


AEKI-Einkaufskorb

- Artikel in den Korb tun — **immer**
- Lieferadresse auswählen
- zur Kasse gehen
- keine Möbel an Packstationen
- keine Eingabedaten verlieren
- zwei Wägen kombinieren

Source: Wikimedia (sofa), Wikimedia (candle diffuser)

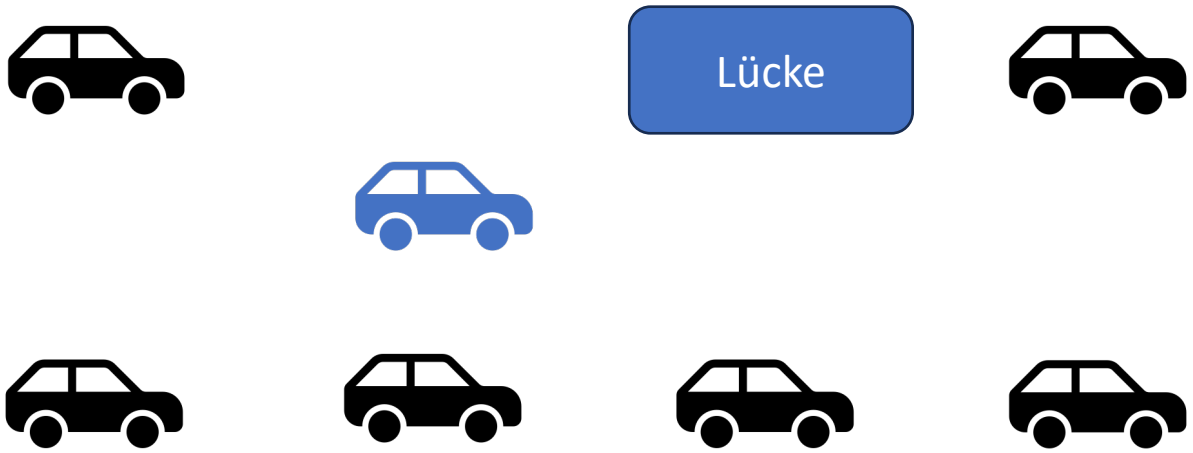
Aus 2 Einkaufskörben einen machen



Gleichungen

$$
\begin{aligned}
x + 2 &= 6 \\
2x^2 + 2y - 2 &= 0 \\
f(5) + f(7) &= 70
\end{aligned}
$$

Wichtige Eigenschaften



Spur(Lücke) = Spur(Wechsel)

Programmieren vs. Gleichungen

$$f(5) + f(7) = 70$$

(FP FTW!)

```
int y = 7;
int f(int x) {
  int z = y;
  y = x;
  return x*z;
}
```

Substitution

$$
\begin{aligned}
x &= 7 \\
x^2 + 2 \cdot x + 5 &= 7^2 + 2 \cdot 7 + 5 \\
&= 49 + 14 + 5 \\
&= 63 + 5
\end{aligned}
$$

Substitution

$$
\begin{aligned}
x &= 2z \\
x^2 + 2 \cdot x + 5 &= (2z)^2 + 2 \cdot (2z) + 5
\end{aligned}
$$

Algebra

$$a \cdot (b \cdot c) = (a \cdot b) \cdot c$$

Assoziativität

$$a \cdot b = b \cdot a$$

Kommutativität

$$a \cdot (b + c) = a \cdot b + a \cdot c$$

Distributivität

Assoziativität

$$(a + b) + c = a + (b + c)$$

$$(a \cdot b) \cdot c = a \cdot (b \cdot c)$$

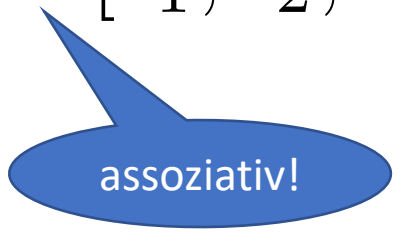
Nicht nur Zahlen

$$(a \vee b) \vee c \quad = \quad a \vee (b \vee c)$$

$$(a \wedge b) \wedge c \quad = \quad a \wedge (b \wedge c)$$

Listen

$$[a_1, a_2, \ldots, a_n] \bowtie [b_1, b_2, \ldots, b_m] = [a_1, a_2, \ldots, a_n, b_1, b_2, \ldots, b_m]$$



Ceci n‘est pas une image!

```
public abstract void drawOval(int x,
                              int y,
                              int width,
                              int height)
```

Draws the outline of an oval. The result is a circle or ellipse that fits within the rectangle specified by the `x`, `y`, `width`, and `height` arguments.

The oval covers an area that is width + 1 pixels wide and height + 1 pixels tall.

Parameters:

`x` - the *x* coordinate of the upper left corner of the oval to be drawn.

`y` - the *y* coordinate of the upper left corner of the oval to be drawn.

`width` - the width of the oval to be drawn.

`height` - the height of the oval to be drawn.

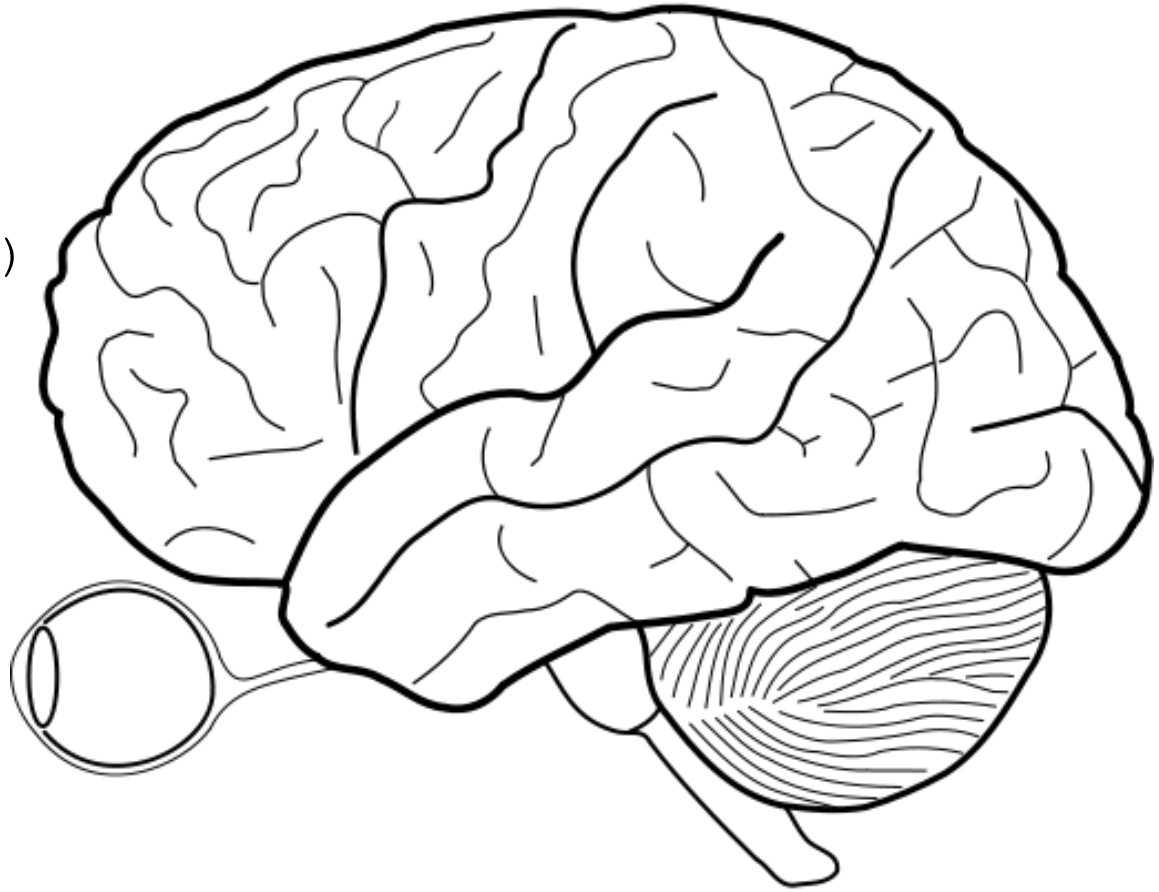
See Also:

`fillOval(int, int, int, int)`

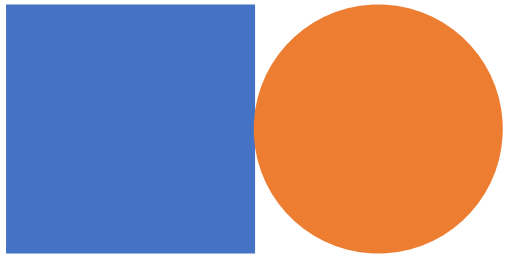
Bilder

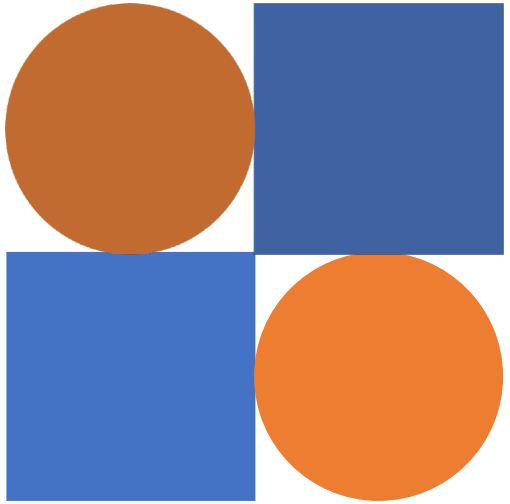
```
public abstract void drawOval(int x,
                              int y,
                              int width,
                              int height)
```

Draws the outline of an oval.

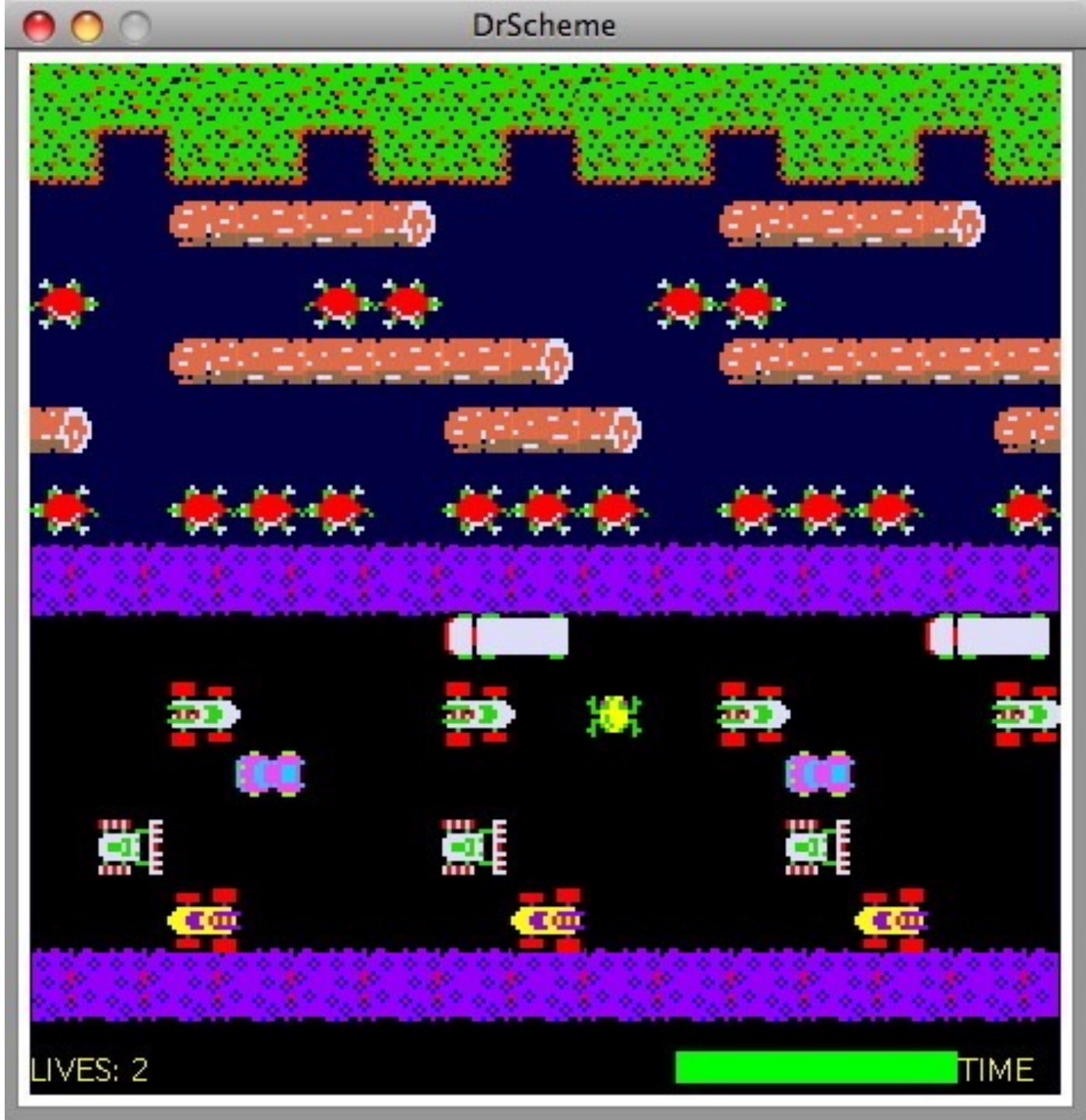


Bilder



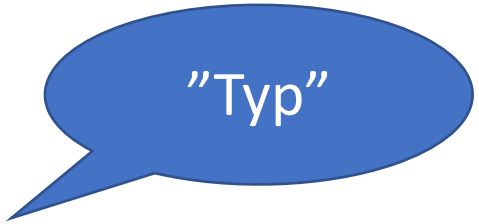


Bilder



Halbgruppe

- Menge(n)
- Operation(en)
  auf diesen Mengen
- Gleichung(en)
  für diese Operationen



$$M$$

$$\blacklozenge : M \times M \to M$$

$$(a \blacklozenge b) \blacklozenge c = a \blacklozenge (b \blacklozenge c)$$

Halbgruppen

$$M$$

$$\blacklozenge : M \times M \to M$$

$$(a \blacklozenge b) \blacklozenge c = a \blacklozenge (b \blacklozenge c)$$

$$\mathbb{R}, + : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$$

$$(a + b) + c = a + (b + c)$$

$$\mathrm{List}\langle A \rangle, \bowtie : \mathrm{List}\langle A \rangle \times \mathrm{List}\langle A \rangle \to \mathrm{List}\langle A \rangle$$

$$(a \bowtie b) \bowtie c = a \bowtie (b \bowtie c)$$

$$\texttt{image}, \texttt{overlay} : \texttt{image} \times \texttt{image} \to \texttt{image}$$

$$(\texttt{overlay}\ (\texttt{overlay}\ a\ b)\ c) = (\texttt{overlay}\ a\ (\texttt{overlay}\ b\ c))$$

Mathe = Patterns fürs Denken

@ctive group

Monoid

$$M$$

Halbgruppe
+ **neutrales Element**

$$\blacklozenge : M \times M \to M$$

$$\bot : M$$

Neutrales Element

$$(a \blacklozenge b) \blacklozenge c = a \blacklozenge (b \blacklozenge c)$$

$$a \blacklozenge \bot = \bot \blacklozenge a = a$$

Monoiden $\mathbb{R}, +, 0$

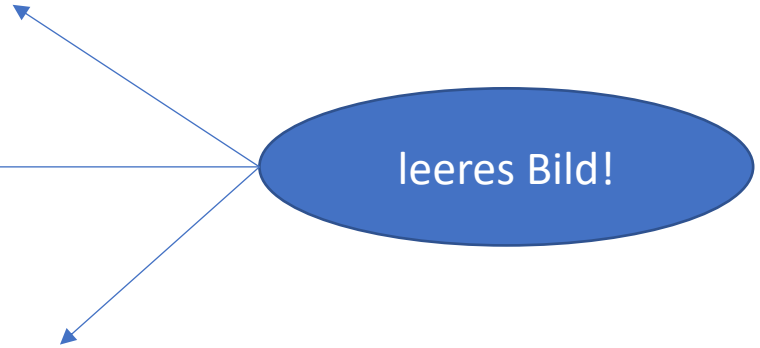
$\mathbb{R}, \cdot, 1$

$\texttt{List}\langle A \rangle, \bowtie, [\,]$

$\texttt{image}, \texttt{beside},$

$\texttt{image}, \texttt{above},$

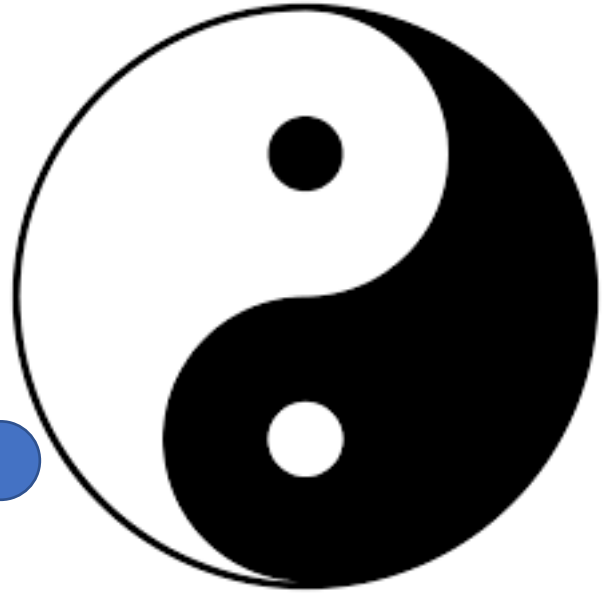
$\texttt{image}, \texttt{overlay},$



Monoiden in freier Wildbahn

- Einkaufskörbe
- Finanzverträge
- Diagramme
- Versicherungsverträge
- Blumensträuße
- Versicherungsansprüche
- Halbleiter-Fabrikationsrouten
- Rechte-Management
- Steuern
- Tabellen
- …

Tao-Check FTW

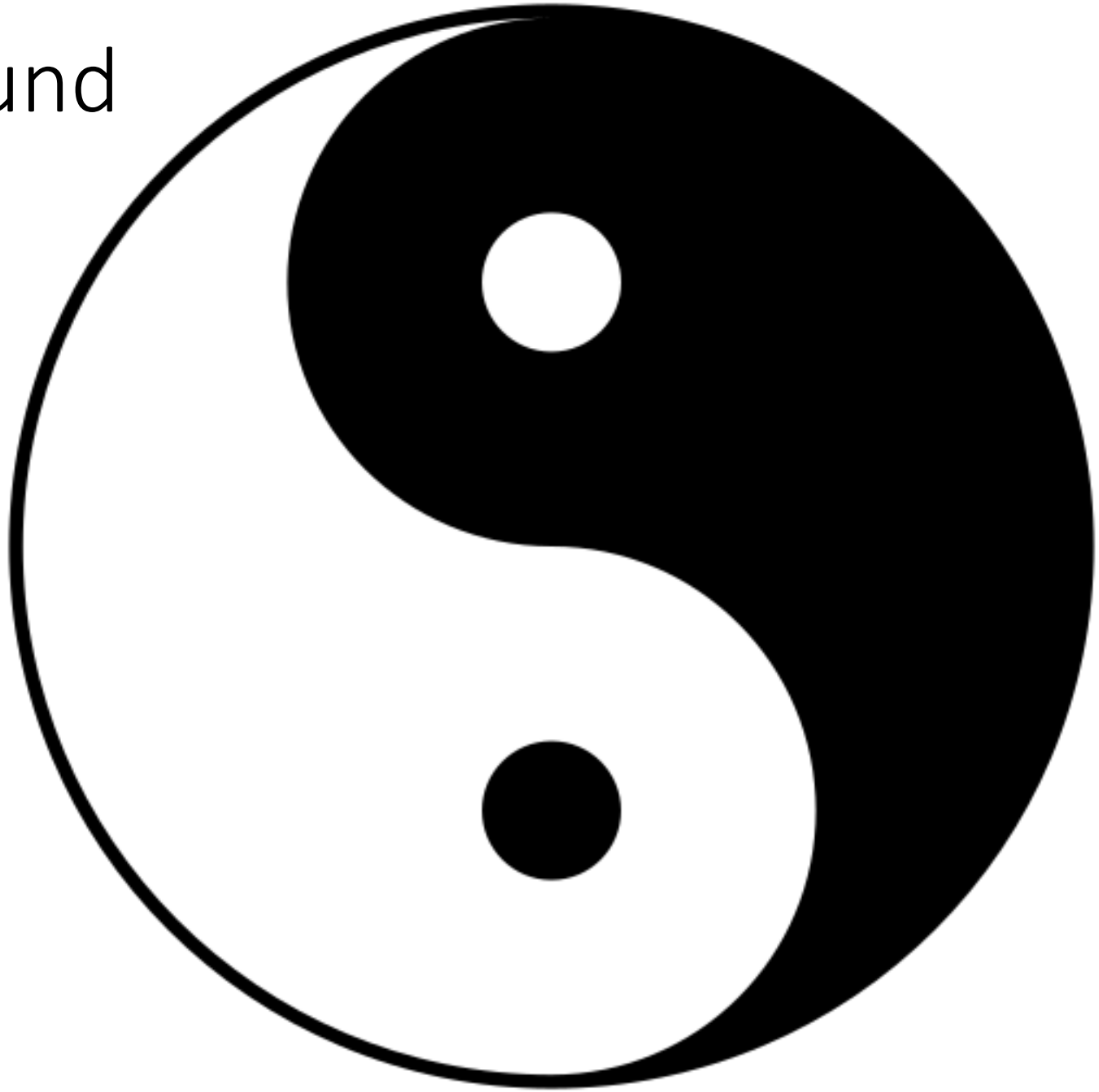


- suche nach binären Operationen
- suche nach neutralem Element

Conal Elliott

@ctive group

Konsistenz und Einfachheit

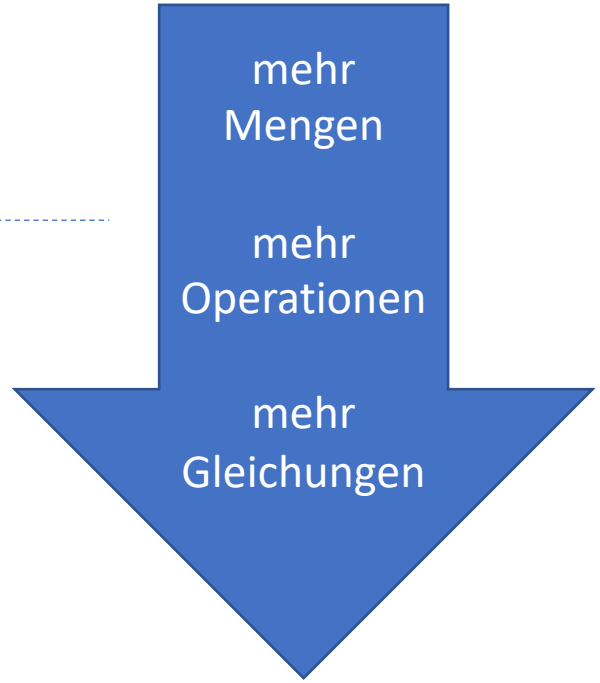


@ctive group

Mehr Struktur

- Halbgruppe
- Monoid

- Gruppe
- Ring
- Körper



Große Monoiden aus kleineren

$$M_1, \blacklozenge_1 : M_1 \times M_1 \to M_1$$
$$M_2, \blacklozenge_2 : M_2 \times M_2 \to M_2$$

$$M_1 \times M_2$$

$$\blacklozenge : (M_1 \times M_2) \times (M_1 \times M_2) \to M_1 \times M_2$$

$$(a_1, a_2) \blacklozenge (b_1, b_2) = (a_1 \blacklozenge_1 b_1,\ a_2 \blacklozenge_2 b_2)$$

Kein Monoid

$$\mathbb{R}$$

$$\max : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$$

$$\max(a, \max(b, c)) = \max(\max(a, b), c)$$

Kein neutrales Element!

Halbgruppen zu Monoiden

$$M$$

$$\blacklozenge : M \times M \to M$$

$$M_\perp = M \cup \{\perp\}$$

Optional<M> in Java

$$\blacklozenge_\perp : M_\perp \times M_\perp \to M_\perp$$

$$a \blacklozenge_\perp \perp = \perp \blacklozenge_\perp a = a$$

$$a \blacklozenge_\perp b = a \blacklozenge b \qquad a, b \neq \perp$$

Halbgruppe in Haskell

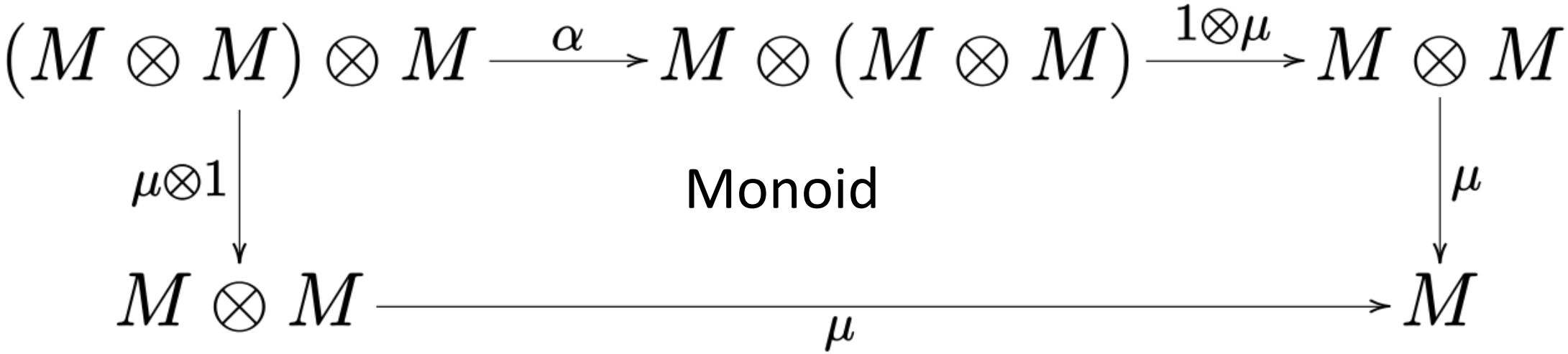
```
class Semigroup a where
   (<>) :: a -> a -> a

(d1 <> d2) <> d3 = d1 <> (d2 <> d3)
```

Monoid in Haskell

```
class Semigroup a => Monoid a where
  mempty :: a
```

```
mempty <> d = d <> mempty = d
```

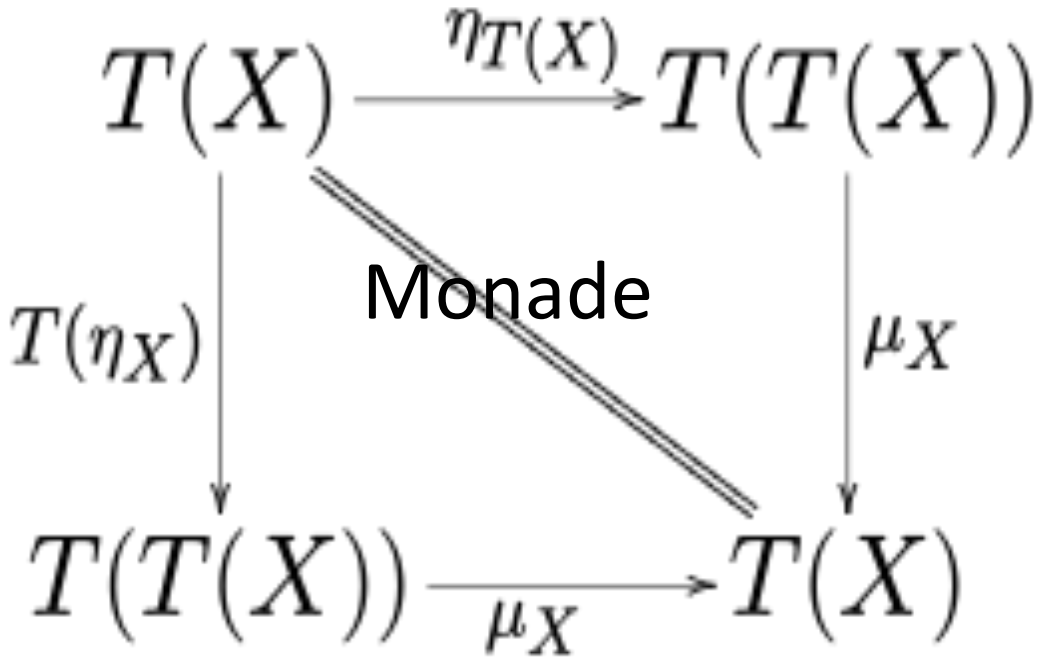
Kategorientheorie - Abstrakter Nonsens



Monoid

Bildquelle: Wikipedia (Monoid)

Mehr Abstraktion



- Funktor
- applikativer Funktor
- Monade
- …

Bildquelle: Wikipedia (Monad)

2022 ACM Software System Award

Gernot Heiser, University of New South Wales; **Gerwin Klein**, Proofcraft; **Harvey Tuch**, Google; **Kevin Elphinstone**, University of New South Wales; **June Andronick**, Proofcraft; **David Cock**, ETH Zurich; **Philip Derrin**, Qualcomm; **Dhammika Elkaduwe**, University of Peradeniya; **Kai Engelhardt**; **Toby Murray**, University of Melbourne; **Rafal Kolanski**, Proofcraft; **Michael Norrish**, Australian National University; **Thomas Sewell**, University of Cambridge; and **Simon Winwood**, Galois, receive the **ACM Software System Award** for the development of the first industrial-strength, high-performance operating system to have been the subject of a complete, mechanically-checked proof of full functional correctness.

AWARDS & RECOGNITION

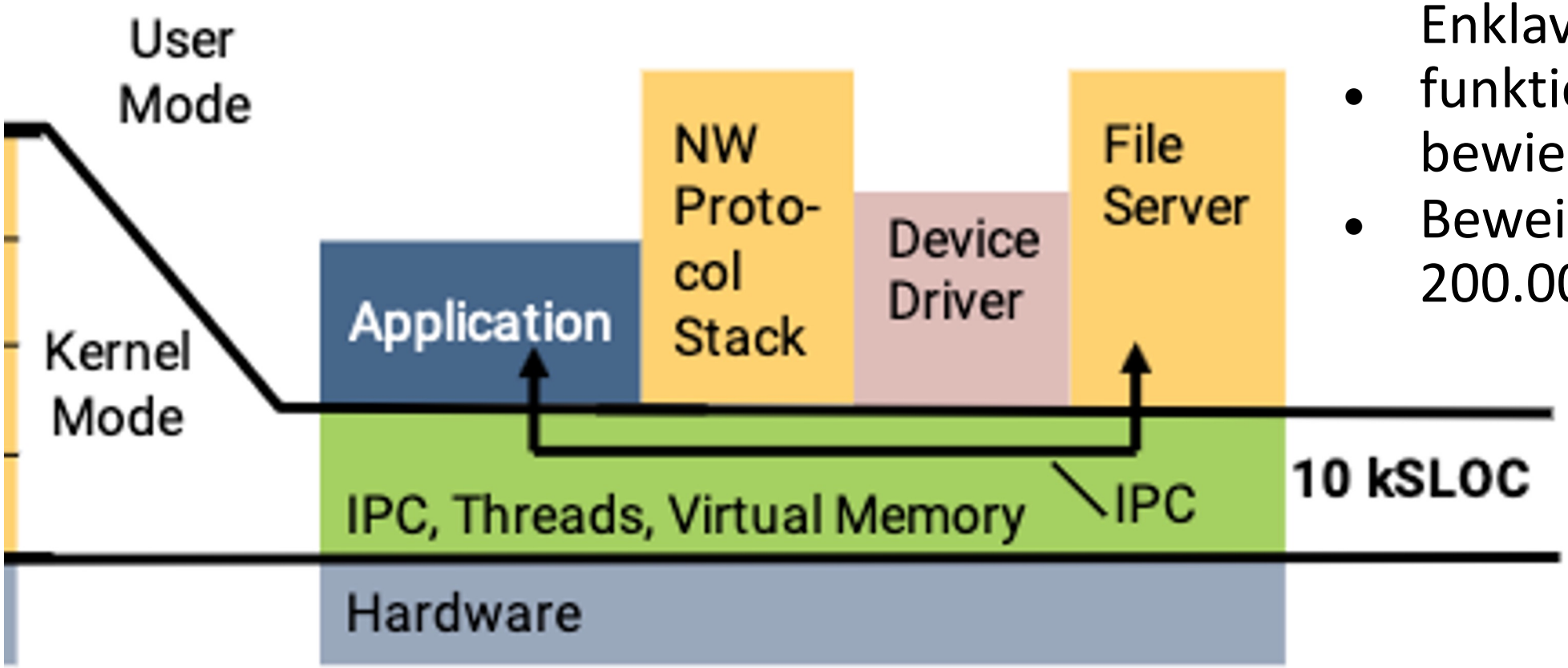
Software System Award Goes to Fourteen for the Development of Groundbreaking High-Performance Operating System

Gernot Heiser, University of New South Wales; **Gerwin Klein**, Proofcraft; **Harvey Tuch**, Google; **Kevin Elphinstone**, University of New South Wales; **June Andronick**, Proofcraft; **David Cock**, ETH Zurich; **Philip Derrin**, Qualcomm; **Dhammika Elkaduwe**, University of Peradeniya; **Kai Engelhardt**; **Toby Murray**, University of Melbourne; **Rafal Kolanski**, Proofcraft; **Michael Norrish**, Australian National University; **Thomas Sewell**, University of Cambridge; and **Simon Winwood**, Galois, receive the **ACM Software System Award** for the development of the first industrial-strength, high-performance operating system to have been the subject of a complete, mechanically-checked proof of full functional correctness.

View the full list of ACM Awards

seL4

- Mikrokernel
- Varianten in Qualcomm-Modems und iPhone-Security-Enklave
- funktionale Korrektheit bewiesen (2009)
- Beweis 200.000loc



Gernot Heiser: The seL4 Microkernel - an Introduction

Abstrakte Spezifikation

[Klein et al: seL4: Formal Verification of an OS Kernel]

```
schedule ≡ do
    threads ← all_active_tcbs;
    thread ← select threads;
    switch_to_thread thread
od OR switch_to_idle_thread
```

Isabelle/HOL

Ausführbare Spezifikation

```
schedule = do
  action <- getSchedulerAction
  case action of
    ChooseNewThread -> do
      chooseThread
      setSchedulerAction ResumeCurrentThread
      ...
chooseThread = do
    r <- findM ... [minBound .. maxBound]
    when (r == Nothing) $ switchToIdleThread
chooseThread' prio = do
     q <- getQueue prio
     liftM isJust $ findM chooseThread'' q
chooseThread'' thread = do
     runnable <- isRunnable thread
     if not runnable then do
             tcbSchedDequeue thread
             return False
     else do
             switchToThread thread
             return True
```

Haskell

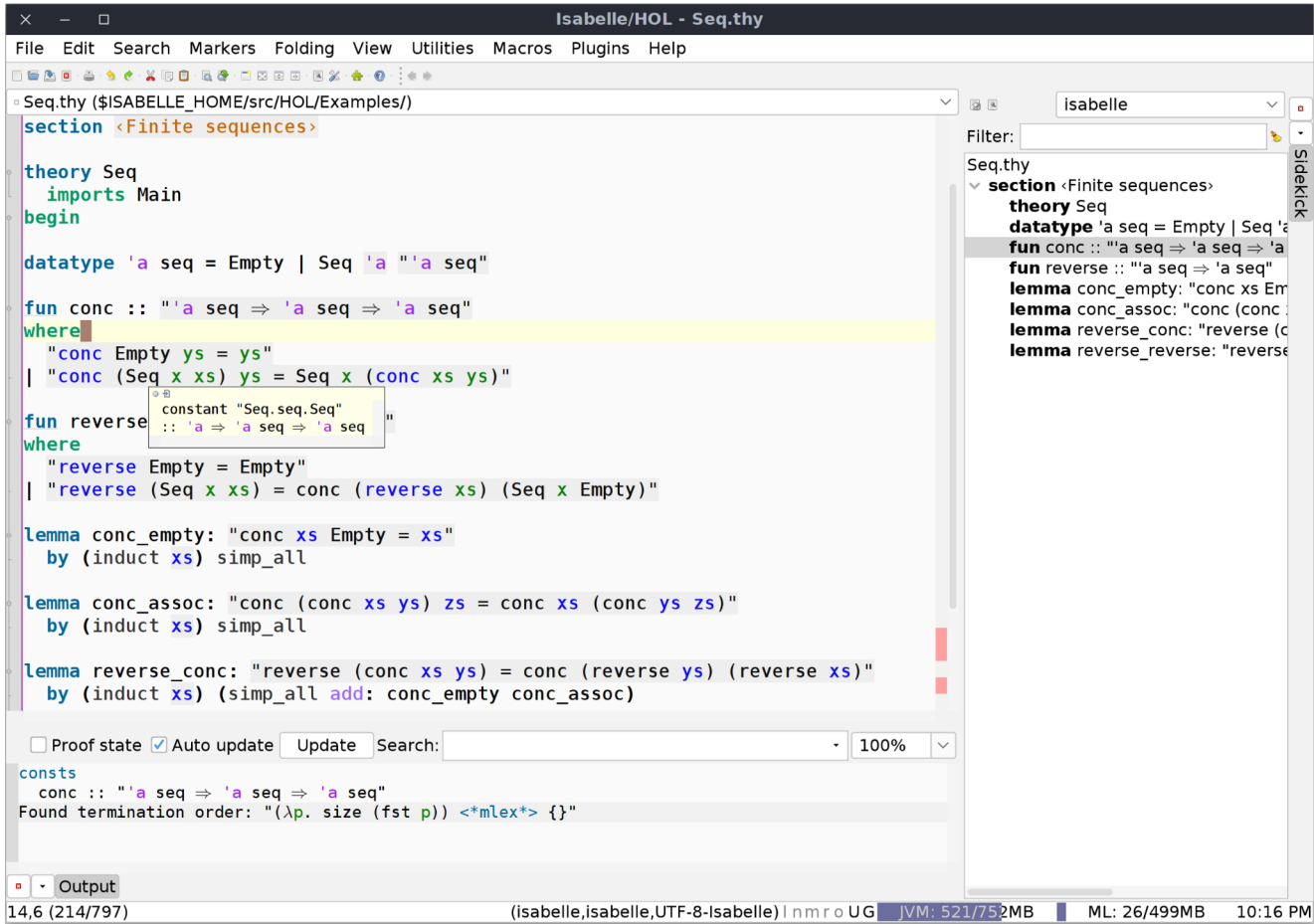
Implementierung

```
void setPriority(tcb_t *tptr, prio_t prio) {
  prio_t oldprio;
  if(thread_state_get_tcbQueued(tptr->tcbState)) {
    oldprio = tptr->tcbPriority;
    ksReadyQueues[oldpri...
      tcbSchedDequeue(t...r, ksReadyQueues[oldprio]);
    if(isRunnable(tptr)...{
      ksReadyQueues[prio] =
        tcbSchedEnqueue(tptr, ksReadyQueues[prio]);
    }
    else {
      thread_state_ptr_set_tcbQueued(&tptr->tcbState,
                                     false);
    }
  }
  tptr->tcbPriority = prio;
}
```

C

Isabelle

- “funktionale Programmiersprache + IDE”
- mathematische Syntax für Mengen, Algebra, Analysis, …
- Sprache für Beweise
- mathematisches Modulsystem
- unendlich erweiterbar



@ctive group

Mathe für Programmierer:innen

- nützlich
- nicht kompliziert
- aber abstrakt
- macht **sehr** glücklich
- funktionale Programmierung FTW

Quellen

- Michael Sperber, Herbert Klaeren: Schreibe Dein Programm!
- Brent Yorgey: Monoids: Theme and Variations
- Sandy Maguire: Algebra-Driven Design
- Conal Elliott: Denotational Design
- Peter Henderson: Functional Geometry