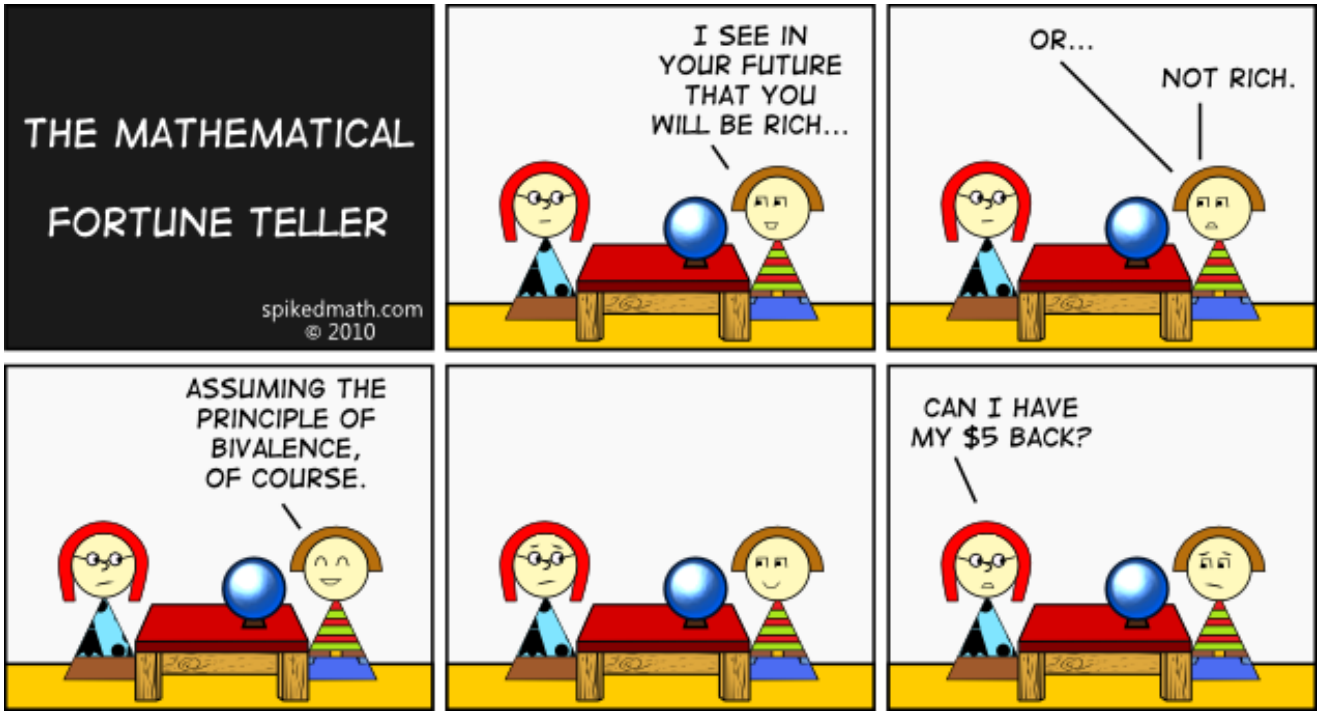


Konstruktive Mathematik, die Doppelnegationsübersetzung und Continuations

Ingo Blechschmidt
Universität Augsburg

Haskell in Leipzig
4. Dezember 2015

Gliederung

Ein Märchen über klassische Logik

Erzähler. Vor langer, langer Zeit begab sich im fernen, fernen Curryland folgende Geschichte. Eines Tages holte die Königin des Landes und aller Haskellistas und Lambdroiden ihren Haus- und Hof-Philosophen zu sich.

Königin. Philosoph! Ich habe folgenden Auftrag an dich: Beschaffe mir den Stein der Weisen, oder alternativ finde heraus, wie man mithilfe des Steins unbegrenzt Gold herstellen kann!

Philosoph. Aber meine Königin! Ich habe nichts Brauchbares studiert! Wie soll ich diese Aufgabe erfüllen?

Königin. Das ist mir egal! Wir sehen uns morgen wieder. Erfüllst du deine Aufgabe nicht, sollst du gehängt werden. Oder wir hacken deinen Kopf ab und verwenden ihn als Cricket-Ball.

Erzähler. Nach einer schlaflosen Nacht voller Sorgen wurde der Philosoph erneut zur Königin berufen.

Königin. Nun! Was hast du mir zu berichten?

Philosoph. Ich habe es tatsächlich geschafft, herauszufinden, wie man den Stein verwenden könnte, um unbegrenzt Gold herzustellen. Aber nur ich kann dieses Verfahren durchführen, Eure Hoheit.

Königin. Nun gut, dann sei es so!

Erzähler. Und so vergingen die Jahre, in denen sich der Philosoph in Sicherheit wähnte und die Angst vor Cricket-Schlägern langsam verlor. Die Königin suchte nun selbst nach dem Stein, aber solange sie ihn nicht fand, hatte der Philosoph nichts zu befürchten. Doch eines Tages passierte das Unfassbare: Die Königin hatte den Stein gefunden! Und lies prompt den Philosophen zu sich rufen.

Königin. Philosoph, sieh! Ich habe den Stein der Weisen gefunden, hier! Nun erfülle du deinen Teil der Abmachung! *[übergibt den Stein]*

Philosoph. Danke. Ihr hattet von mir verlangt, Euch den Stein der Weisen zu beschaffen oder herauszufinden, wie man mit ihm unbegrenzt Gold herstellen kann. Hier habt Ihr den Stein der Weisen. *[übergibt den Stein zurück]*

Nichtkonstruktive Beweise

Eine Zahl heißt genau dann **rational**, wenn sie sich als Bruch zweier ganzer Zahlen schreiben lässt.

- $\frac{21}{13}$ und 37 sind rational.
- $\sqrt{2}$ und π sind irrational.

Nichtkonstruktive Beweise

Eine Zahl heißt genau dann **rational**, wenn sie sich als Bruch zweier ganzer Zahlen schreiben lässt.

- $\frac{21}{13}$ und 37 sind rational.
- $\sqrt{2}$ und π sind irrational.

Satz. Es gibt **irrationale** Zahlen x und y sodass x^y rational ist.

Nichtkonstruktive Beweise

Eine Zahl heißt genau dann **rational**, wenn sie sich als Bruch zweier ganzer Zahlen schreiben lässt.

- $\frac{21}{13}$ und 37 sind rational.
- $\sqrt{2}$ und π sind irrational.

Satz. Es gibt **irrationale** Zahlen x und y sodass x^y rational ist.

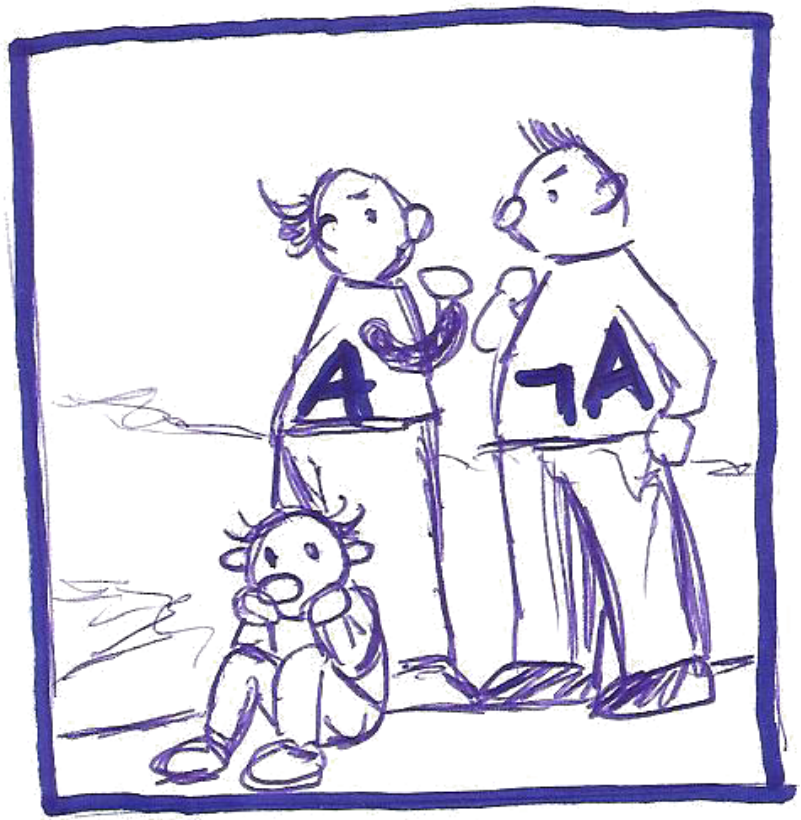
Beweis. Entweder ist $\sqrt{2}^{\sqrt{2}}$ rational oder nicht.

1. Im ersten Fall sind wir fertig.
2. Im zweiten Fall können wir $x := \sqrt{2}^{\sqrt{2}}$ und $y := \sqrt{2}$ nehmen. Dann ist $x^y = \sqrt{2}^{\sqrt{2}\cdot\sqrt{2}} = \sqrt{2}^2 = 2$ rational.

Das Axiom vom ausgeschlossenen Dritten

„Für jede Aussage A dürfen wir $A \vee \neg A$ voraussetzen.“

Klassische Logik =
konstruktive Logik + das Axiom vom ausgeschlossenen Dritten.



Das Axiom vom ausgeschlossenen Dritten

„Für jede Aussage A dürfen wir $A \vee \neg A$ voraussetzen.“

Klassische Logik =
konstruktive Logik + das Axiom vom ausgeschlossenen Dritten.

Klassische Interpretation

$\bot$	Widerspruch.
$A \wedge B$	A und B sind wahr.
$A \vee B$	A ist wahr oder B ist wahr.
$A \Rightarrow B$	Sollte A wahr sein, so auch B.
$\forall x{:}X.\, A(x)$	Für alle $x : X$ gilt $A(x)$.
$\exists x{:}X.\, A(x)$	Es gibt ein $x : X$ mit $A(x)$.

Klassische vs. konstruktive Logik

Symbol	klassisch	konstruktiv
$\bot$	Widerspruch.	Widerspruch.
$A \wedge B$	A und B sind wahr.	Wir haben Beleg für A und für B.
$A \vee B$	A ist wahr oder B ist wahr.	Wir haben Beleg für A oder für B.
$A \Rightarrow B$	Aus A folgt B.	Wir können Beleg für A in Beleg für B transformieren.
$\forall x{:}X.\, A(x)$	Für alle $x : X$ gilt $A(x)$.	Zu jedem $x : X$ können wir Beleg für $A(x)$ konstruieren.
$\exists x{:}X.\, A(x)$	Es gibt ein $x : X$ mit $A(x)$.	Wir haben ein $x : X$ zusammen mit Beleg für $A(x)$.

Klassische vs. konstruktive Logik

Symbol	klassisch	konstruktiv
$\bot$	Widerspruch.	Widerspruch.
$A \wedge B$	A und B sind wahr.	Wir haben Beleg für A und für B.
$A \vee B$	A ist wahr oder B ist wahr.	Wir haben Beleg für A oder für B.
$A \Rightarrow B$	Aus A folgt B.	Wir können Beleg für A in Beleg für B transformieren.
$\forall x{:}X.\, A(x)$	Für alle $x : X$ gilt $A(x)$.	Zu jedem $x : X$ können wir Beleg für $A(x)$ konstruieren.
$\exists x{:}X.\, A(x)$	Es gibt ein $x : X$ mit $A(x)$.	Wir haben ein $x : X$ zusammen mit Beleg für $A(x)$.
$\neg A$	A ist falsch.	Es gibt keinen Beleg für A.
$\neg\neg A$	A ist nicht nicht wahr.	Es gibt keinen Beleg für $\neg A$.

Die Doppelnegationsübersetzung

$$
\begin{aligned}
(x = y)^{\square} &:\equiv \neg\neg(x = y) \\
(A \wedge B)^{\square} &:\equiv \neg\neg(A^{\square} \wedge B^{\square}) \\
(A \vee B)^{\square} &:\equiv \neg\neg(A^{\square} \vee B^{\square}) \\
(A \Rightarrow B)^{\square} &:\equiv \neg\neg(A^{\square} \Rightarrow B^{\square}) \\
(\forall x{:}X.\, A(x))^{\square} &:\equiv \neg\neg(\forall x{:}X.\, A^{\square}(x)) \\
(\exists x{:}X.\, A(x))^{\square} &:\equiv \neg\neg(\exists x{:}X.\, A^{\square}(x))
\end{aligned}
$$

Beispiel: Die Doppelnegationsübersetzung von

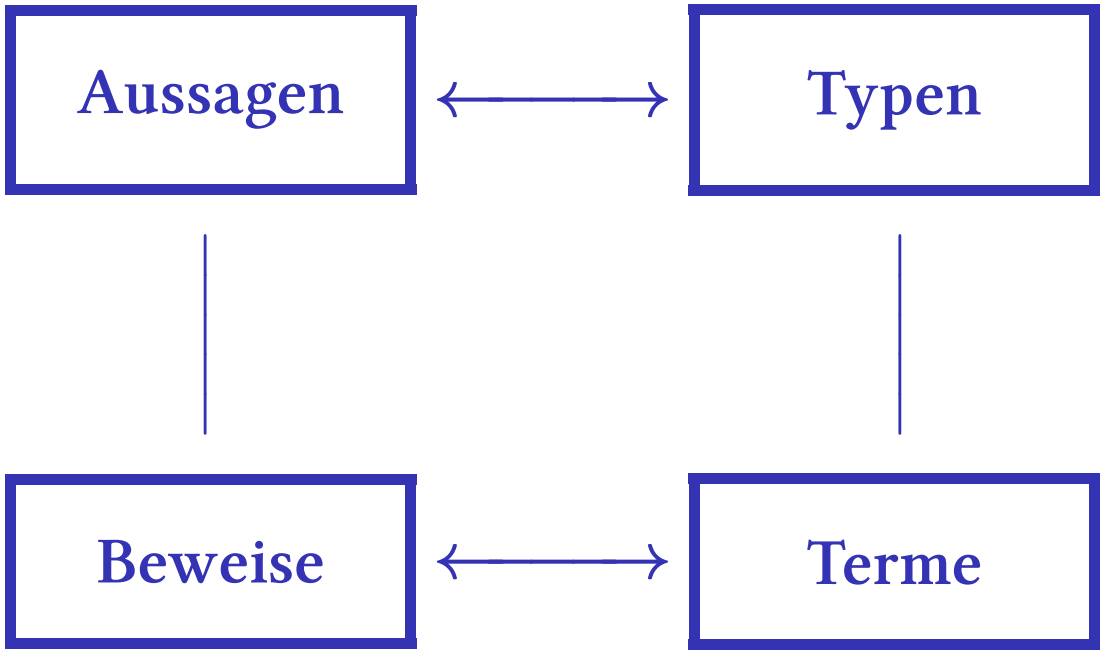
Es gibt eine Stelle, an der der Schlüssel liegt.

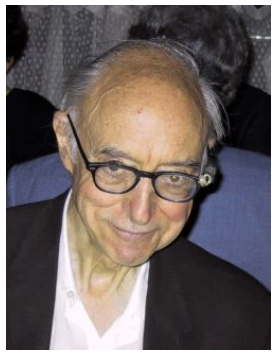
ist

*Es gibt **nicht nicht** eine Stelle, an der der Schlüssel liegt.*

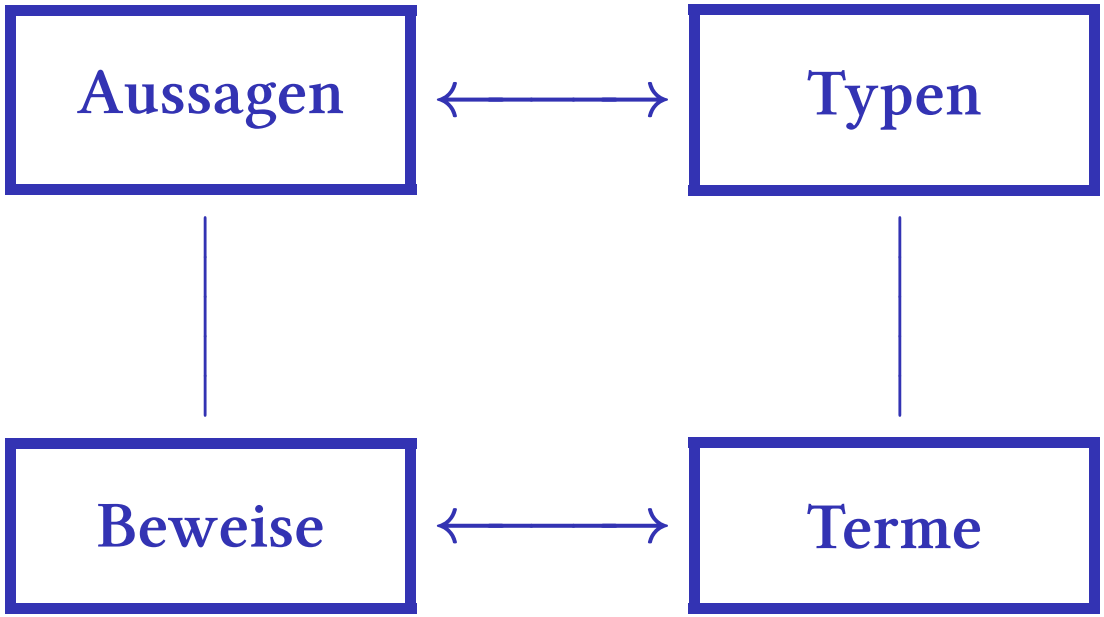
Satz. Es gilt A genau dann klassisch, wenn $A^{\square}$ konstruktiv gilt.

Die Curry–Howard-Korrespondenz





Die Curry–Howard-Korrespondenz



Logik	Programmierung
$A \Rightarrow A$	`A → A`
$(A \wedge B) \Rightarrow A$	`(A,B) → A`
$A \Rightarrow (A \vee B)$	`A → Either A B`

Die Curry–Howard-Korrespondenz

Logik	Programmierung
Aussage A	Typ `A` der Belege für A
Beweis von A	Programm vom Typ `A`
$A \Rightarrow A$	`A → A`
$(A \wedge B) \Rightarrow A$	`(A,B) → A`
$A \Rightarrow (A \vee B)$	`A → Either A B`
$A \Rightarrow B$	`A → B`
Es gibt eine natürliche Zahl.	`Nat`

Aus jedem konstruktiven Beweis kann man ein Programm extrahieren. Jedes Programm beweist eine Behauptung.

Die Curry–Howard-Korrespondenz

Logik	Programmierung
Aussage A	Typ `A` der Belege für A
Beweis von A	Programm vom Typ `A`
$A \Rightarrow A$	`A → A`
$(A \wedge B) \Rightarrow A$	`(A,B) → A`
$A \Rightarrow (A \vee B)$	`A → Either A B`
$A \Rightarrow B$	`A → B`
Es gibt eine natürliche Zahl.	`Nat`
$\neg A$	??

Aus jedem konstruktiven Beweis kann man ein Programm extrahieren. Jedes Programm beweist eine Behauptung.

Die Curry–Howard-Korrespondenz

Logik	Programmierung
Aussage A	Typ `A` der Belege für A
Beweis von A	Programm vom Typ `A`
$A \Rightarrow A$	`A → A`
$(A \wedge B) \Rightarrow A$	`(A,B) → A`
$A \Rightarrow (A \vee B)$	`A → Either A B`
$A \Rightarrow B$	`A → B`
Es gibt eine natürliche Zahl.	`Nat`
$\neg A$, d. h. $A \Rightarrow \bot$	??

Aus jedem konstruktiven Beweis kann man ein Programm extrahieren. Jedes Programm beweist eine Behauptung.

Die Curry–Howard-Korrespondenz

Logik	Programmierung
Aussage A	Typ `A` der Belege für A
Beweis von A	Programm vom Typ `A`
$A \Rightarrow A$	`A → A`
$(A \wedge B) \Rightarrow A$	`(A,B) → A`
$A \Rightarrow (A \vee B)$	`A → Either A B`
$A \Rightarrow B$	`A → B`
Es gibt eine natürliche Zahl.	`Nat`
$\neg A$, d. h. $A \Rightarrow \bot$	`A → r`

Aus jedem konstruktiven Beweis kann man ein Programm extrahieren. Jedes Programm beweist eine Behauptung.

Die Curry–Howard-Korrespondenz

Logik	Programmierung
Aussage A	Typ `A` der Belege für A
Beweis von A	Programm vom Typ `A`
$A \Rightarrow A$	`A → A`
$(A \wedge B) \Rightarrow A$	`(A,B) → A`
$A \Rightarrow (A \vee B)$	`A → Either A B`
$A \Rightarrow B$	`A → B`
Es gibt eine natürliche Zahl.	`Nat`
$\neg A$, d. h. $A \Rightarrow \bot$	`A → r`
$\neg\neg A$, d. h. $(A \Rightarrow \bot) \Rightarrow \bot$	??

Aus jedem konstruktiven Beweis kann man ein Programm extrahieren. Jedes Programm beweist eine Behauptung.

Die Curry–Howard-Korrespondenz

Logik	Programmierung
Aussage A	Typ `A` der Belege für A
Beweis von A	Programm vom Typ `A`
$A \Rightarrow A$	`A → A`
$(A \wedge B) \Rightarrow A$	`(A,B) → A`
$A \Rightarrow (A \vee B)$	`A → Either A B`
$A \Rightarrow B$	`A → B`
Es gibt eine natürliche Zahl.	`Nat`
$\neg A$, d. h. $A \Rightarrow \bot$	`A → r`
$\neg\neg A$, d. h. $(A \Rightarrow \bot) \Rightarrow \bot$	`(A → r) → r`

Aus jedem konstruktiven Beweis kann man ein Programm extrahieren. Jedes Programm beweist eine Behauptung.

$\neg\neg$-Übersetzung $\widehat{=}$ CPS-Transformation

Logik	Programmierung
Aussage A	Typ `A` der Belege für A
Beweis von A	Programm vom Typ `A`
$\neg\neg A$, d. h. $(A \Rightarrow \bot) \Rightarrow \bot$	`(A → r) → r`, d. h. `Cont r A`
Doppelnegationsübersetzung	CPS-Transformation

```
type Cont r a = ((a → r) → r)

-- Axiom vom ausgeschlossenen Dritten ohne Übersetzung:
-- lem :: Either a (a → r)

lem :: Cont r (Either a (a → Cont r b))
lem = \k → k $ Right $ \x → (\k' → k (Left x))
```

Minima unendlicher Listen

Satz. In jeder unendlichen Liste `xs` natürlicher Zahlen gibt es ein kleinstes Element.

Aber nicht konstruktiv! Es gibt kein Programm vom Typ:

```
minimum :: [Nat] → Ix
-- Ix: Typ der Indizes (also Nat)
```

Minima unendlicher Listen

Satz. In jeder unendlichen Liste `xs` natürlicher Zahlen gibt es ein kleinstes Element.

Aber nicht konstruktiv! Es gibt kein Programm vom Typ:

```
minimum :: [Nat] → Ix
-- Ix: Typ der Indizes (also Nat)
```

Beweis. Durch noethersche Induktion über die Größe eines gegebenen Elements `xs!!i`. Entweder gibt es einen Index `j` mit `xs!!j < xs!!i` oder nicht.

1. Im ersten Fall gibt es ein Minimum nach Ind'voraussetzung.
2. Im zweiten Fall ist `xs!!i` minimal.

Minima unendlicher Listen

```
type Cont r a = ((a → r) → r)

lem :: Cont r (Either a (a → Cont r b))
lem = \k → k $ Right $ \x → (\k' → k (Left x))

minimum :: [Nat] → Cont r (Ix, Ix → Cont r ())
minimum xs = go 0 where
    go i = do
        oracle ← lem
        case oracle of
            Left  j → go j
            Right f → return (i, \j →
                if xs!!j ≥ xs!!i then return () else f j)
```

Minima unendlicher Listen

```
type Cont r a = ((a → r) → r)

lem :: Cont r (Either a (a → Cont r b))
lem = \k → k $ Right $ \x → (\k' → k (Left x))

minimum :: [Nat] → Cont r (Ix, Ix → Cont r ())
minimum xs = go 0 where
   go i = do
      oracle ← lem
      case oracle of
         Left  j → go j
         Right f → return (i, \j →
            if xs!!j ≥ xs!!i then return () else f j)

example = do
   (i, g) ← minimum [...]
   g 5 ≫ g 7 ≫ g 3
   ...
```

Fazit

- Konstruktive Beweise haben algorithmischen Inhalt. Dieser lässt sich mit der Curry–Howard-Korrespondenz maschinell extrahieren.
- Die Doppelnegationsübersetzung macht aus jedem klassischen Beweis einen konstruktiven. Auf diese Weise steckt auch in klassischen Beweisen noch algorithmischer Inhalt. Dieser spielt sich in der Continuation-Monade ab.
- Manchmal lösen die so erhaltenen Algorithmen konkrete Probleme.

Gründe einen funktionalen Stammtisch!

In Augsburg haben wir ein monatliches Treffen mit Vorträgen und gemeinsamen Restaurantbesuch für alle, die an funktionalen Sprachen interessiert sind, ins Leben gerufen. Überleg doch, so etwas auch in deiner Stadt zu gründen!

Gründe einen Matheschülerzirkel!

Organisiere ein Angebot, bei dem an Mathematik interessierte Schülerinnen und Schüler aus deiner Stadt regelmäßig an die Uni kommen können, um in kleinen Gruppen spannende Mathematik abseits des Schulunterrichts zu sehen. Mach das ganze per Post für weiter entfernt wohnende Kinder und Jugendliche. Und veranstalte immer in den Sommerferien ein einwöchiges Mathecamp.

In Augsburg machen wir das seit etwa drei Jahren mit durchschnittlich etwa 200 Teilnehmenden. Vielleicht würde auch in deiner Stadt ein solches Angebot gut aufgenommen werden! Wir können dich mit umfangreichen Materialien und Erfahrungsberichten versorgen.