BME TMIT
2022

14/6a

Németh Gábor
Janky Ferenc Nándor vázlatai alapján

Funkcionális programozás C++-ban

Kategóriaelmélet-
bevezető

Kategóriaelmélet - bevezető

- a programozásban – főleg az imperatív nyelvek esetében - nem ugyanazt értjük függvény alatt, mint a matematikában
  - matematikai függvény: leképzés az értelmezési tartomány elemeiről az értékkészlet elemeire
- ha egy függvény mellékhatásokkal rendelkezik, nehezen modellezhető matematikailag. Miért szükséges modellezni?
  - helyesség bebizonyítása
  - determinisztikus viselkedés
- **<u>Tiszta függvény (pure function):</u>** olyan függvény, amelynek nincsenek mellékhatásai és a kimenete csak a bemeneti paraméterétől függ, ugyanarra a bemenetre, mindig ugyanazt a kimenetet adja (referenciálisan transzparens kifejezés)

Kategóriaelmélet - bevezető

- Kategória: egyszerű koncepció amely a matematikai struktúrákat formalizálja
- **<u>Definició</u>**: ***C*** egy kategória, amely áll:
  - <u>Objektumok gyűjteményéből:</u>
    - $ob(C) = \{a, b, c, d, \dots\}$
  - <u>Nyilakból (morfizmusok):</u>
    - $mor(C) = \{f\colon a \to b, g\colon b \to c, h\colon c \to d, \dots\}$
    - $f\colon a \to b$; értsd: f egy nyíl a-ból b-be
    - $\mathrm{hom}(a, b) = \mathrm{hom}_C(a, b) = mor(a, b) = \mathrm{C}(\mathrm{a}, \mathrm{b})$: *hom-halmaz,* az összes morfizmus a-ból b-be

Kategóriaelmélet - bevezető

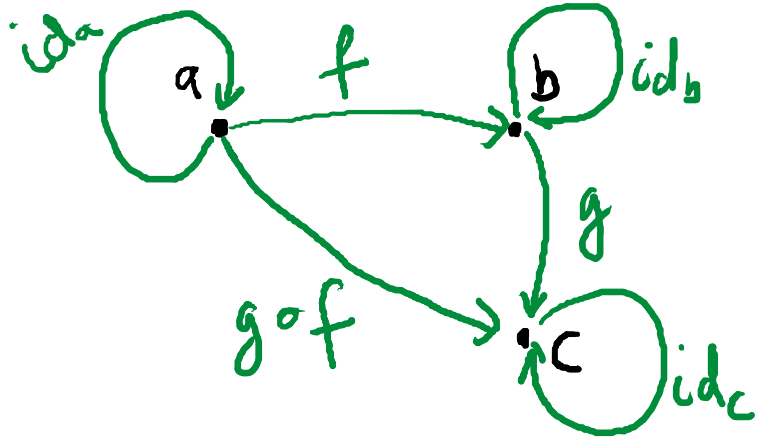
- Két műveletből: ***dom,cod:*** amely hozzárendeli minden nyílhoz a forrás és a cél objektumát, pl: ***dom***_(f)=a,_ ***cod***_(f)=b_

- Identitás nyilakból:
  - $\forall \mathrm{o} \in ob(C) \exists id_o \in mor(C), hogy\ dom(id_o) = o\ \wedge cod(id_o) = o$

- Kompozíció bináris operátorból:
  - $\circ \coloneqq\ \forall\ f, g \in mor(C), ahol\ cod(f) = dom(g)$
    $\exists\ g \circ f, hogy\ dom(g \circ f) = dom(f) \wedge cod(g \circ f) = cod(g)$

Kategóriaelmélet - bevezető

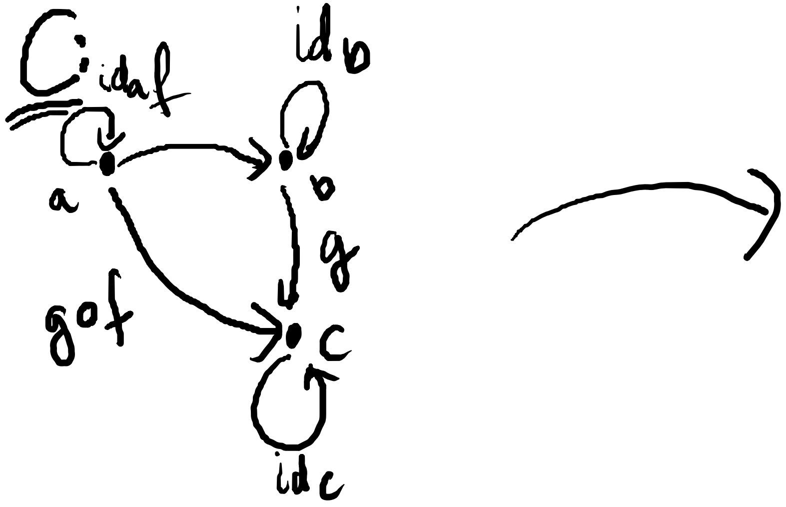
- Vizualizáció általában irányított, címkézett gráfokkal:

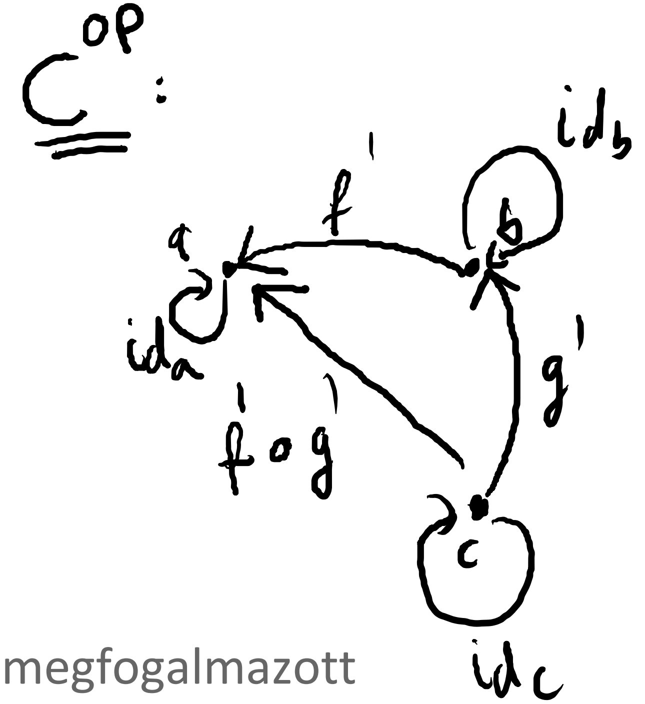


- Megfordítása: minden irányított gráf kiegészíthető élekkel, hogy érvényes kategóriát kapjunk – szabad kategória

Kategóriaelmélet - bevezető

- <u>Duális kategória:</u> szabatosan: minden nyílnak megfordul az iránya ,pl:





- Következmény: Általánosan minden megfogalmazott állításnak létezik egy duális állítása, amely állítás igaz a C kategóriában, annak az állításnak a duálisa igaz a C^{OP} kategóriában

Kategóriaelmélet - bevezető

- Az identitásnak és a kompozíciónak az alábbi „törvényeknek” kell engedelmeskedniük:
  - ***Identitás axióma:***
    - $\forall f, g \in Mor_C, hogy\ cod(f) = a = dom(g)$
      - $id_a \circ f = f$
      - $g \circ id_a = g$
  - ***Asszociativitás törvénye:***
    - $\forall f, g, h \in Mor_C, hogy\ cod(f) = dom(g) \wedge cod(g) = dom(h)$
      - $(h \circ g) \circ f = h \circ (g \circ f)$

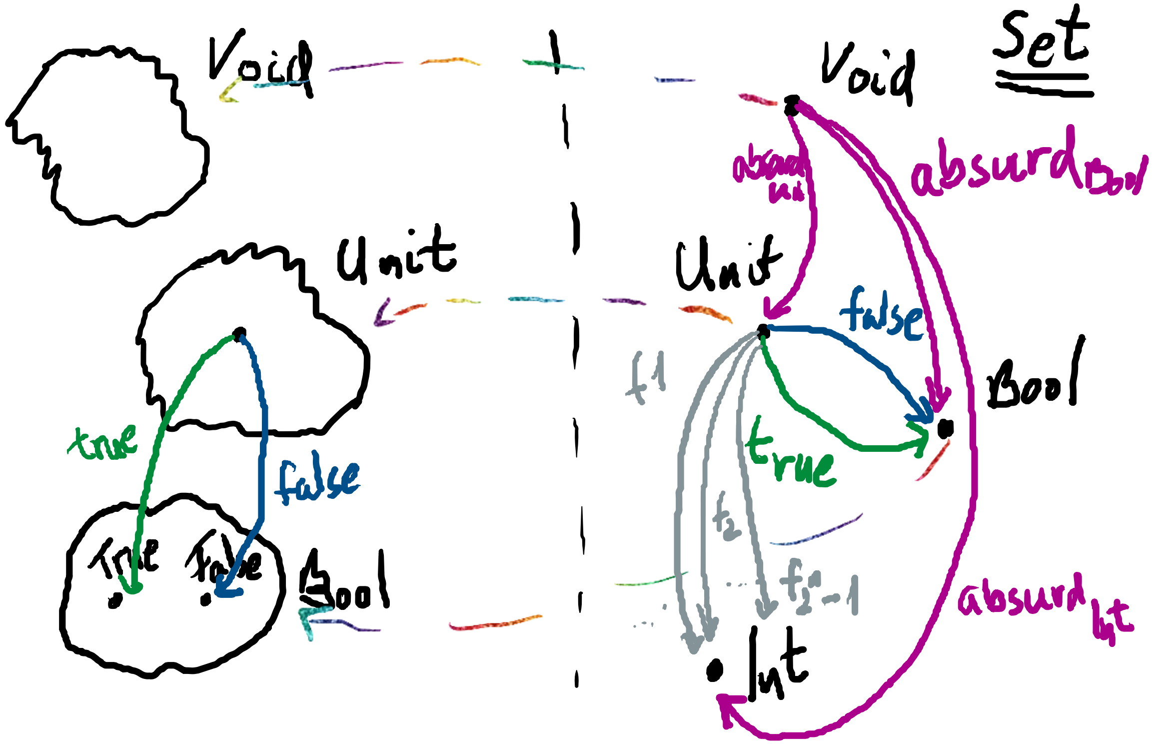
Kategóriaelmélet - bevezető

- Példák kategóriákra:

Kategória	Objektumok	Morfizmusok
Set	Halmazok	Függvények
Top	Topológiai terek	Folytonos függvények
Vect	Vektorterek	Lineáris transzformációk
Cat	Kategóriák	Funktorok

Mik is a típusok?

- <u>**Intuitíven:**</u> egy típus értékek halmaza
  - Pl.: Bool = {True,False} , jelentsen a True és False bármit is…
- **Set** kategória szuper alany, mivel itt bele tudunk nézni az objektumok belsejébe
  - Üres halmaz
  - Egy elemű halmaz
  - Kételemű halmaz
  - Stb.

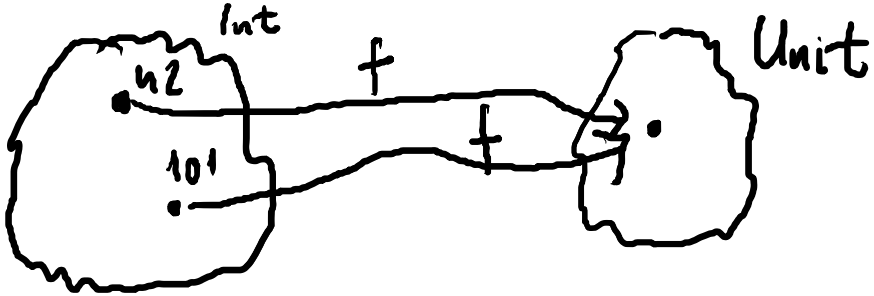


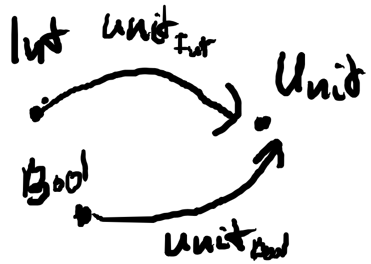
Mik is a típusok?

- <u>Speciális esetek:</u>
  - ***<u>Void</u>*****:** abszurd morfizmus, minden objektumba pontosan egy morfizmus vezet: ha adsz egy példányt az üres halmazból, bármit előállítok neked… „ex falso sequitur quodlibet”, azaz a hamis állításból minden következik
  - ***<u>Unit:</u>*** egyelemű halmaz, C++ *void* típusával analóg, illetve C++17-től az *std::monostate-tel* (a két típus egyező izomorfizmus erejéig)*, unit* morfizmus

  - ***<u>Pl.:</u>***

```
void unit(int){} using /*vagy */ Unit = std::monostate; Unit unit(int){ return {};}
```





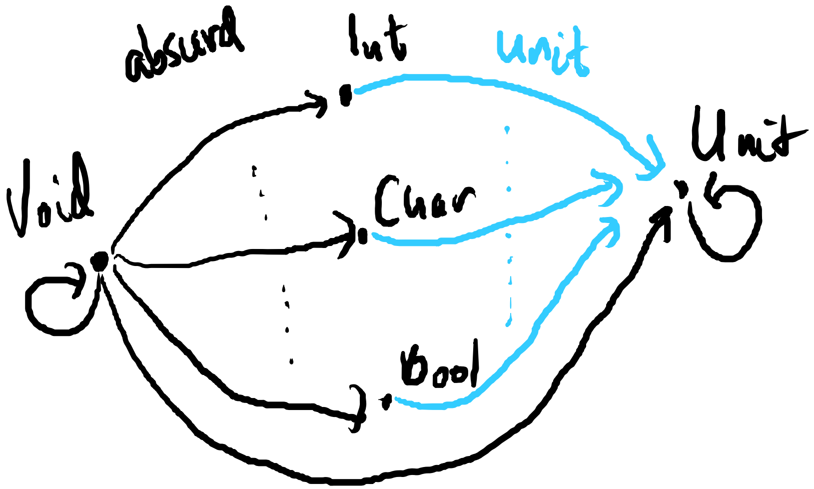
Izomorfizmus

- Matematikailag: (szabatosan) létezik leképzés a-ból b-be, és b-ből a-ba, és ez a két leképzés egymás inverze.

- Kategóriaelméletben: Invertálható morfizmus, azaz egy morfizmus pár, amelynek kompozíciója az identitást adja
  - $g \circ f = id_x$
  - $f \circ g = id_y$
    - Megj.: x és y nem feltétlenül ugyanaz az objektum!

Kezdeti és terminális objektum

- <u>**Definíció:**</u> ha létezik a kategóriában kezdeti objektum, akkor az egy olyan objektum, amelyből minden objektumba egy és csakis egy morfizmus vezet.
  - **Set** kategória esetén: ***Void*** és az ***abszurd*** morfizmus.
- <u>**Definíció:**</u> ha létezik a kategóriában terminális objektum, akkor az egy olyan objektum, amelybe minden objektumból egy és csakis egy morfizmus vezet.
  - **Set** kategória esetén: ***Unit*** és az ***unit*** morfizmus.



Kezdeti és terminális objektum

- A kezdeti (és a terminális) objektumok egyediek izomorfizmus erejéig:
  - Tfh.: 2 db kezdeti objektum létezik: $i_1$és i_2

$\underline{C}$: kategória

$$\begin{tikzcd} i_1 \arrow[loop left, "f''=id_1"] \arrow[r, bend left, "f'"] & i_2 \arrow[l, bend left, "g'"] \arrow[loop right, "g''=id_2"] \end{tikzcd}$$

$\exists\, id_o \;\forall o \in Ob(C)$

$g' \circ f' = id_1$

$f' \circ g' = id_2$ ∎

Gyakorlati feladatok

- Implementáljuk C++-ban az identitás függvényt a “legjobban”
- Implementáljuk a kompozíciót C++-ban
- Teszteljük, hogy az identitás és asszociáció törvényei fennálnak-e
- Mutassuk be egy példán, hogy az invertálható függvények kompozíciója valóban az identitás-e
- Implementáljuk a **Void** típust és az abszurd függvényt C++-ban

Kategóriaelmélet-
univerzális struktúrák

Szorzat univerzális struktúra

- **<u>Definició:</u>** c objektum a és b objektum szorzata, amelyhez tartozik két projekció, úgy, hogy bármely más c’ objektumból, amelyhez hasonlóan tartozik két projekció, létezik egy egyedi m morfizmus, amely faktorizálja ezt a két projekciót

$$
\begin{tikzcd}
 & c' \arrow[d, dashed, "m"] \arrow[ddl, bend right, "p'"'] \arrow[ddr, bend left, "q'"] & \\
 & c \arrow[dl, "p"'] \arrow[dr, "q"] & \\
a & & b
\end{tikzcd}
$$

$$
\left.\begin{aligned} p' &= p \circ m \\ q' &= q \circ m \end{aligned}\right\} \Rightarrow \exists m \Leftrightarrow c = a \times b
$$

Koproduktum (összeg) univerzális struktúra

- **Definició:** c objektum a és b objektum koproduktuma(összege), amelyhez tartozik két injekció, úgy, hogy bármely más c’ objektumba, amelyhez hasonlóan tartozik két injekció, létezik egy egyedi m morfizmus, amely faktorizálja ezt a két injekciót

$$\begin{tikzcd}
a \arrow[r, "p"] \arrow[dr, bend right, "p'"'] & c \arrow[d, dashed, "m"] & b \arrow[l, "q"'] \arrow[dl, bend left, "q'"] \\
 & c' &
\end{tikzcd}$$

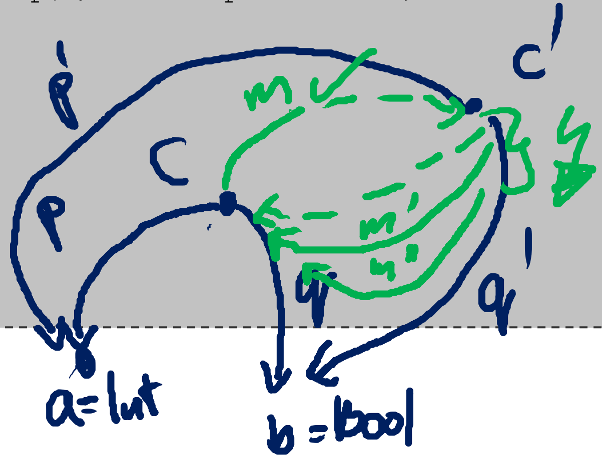
$$\left.\begin{array}{l} p' = m \circ p \\ q' = m \circ q \end{array}\right\} \exists m \text{ egyedi} \Leftrightarrow c = a + b$$

Algebrai típusok C++-ban

- Szorzatnak az *std::pair* felel meg

```
using c = std::pair<int,bool>; // c tipus a peldaban,a = Int, b = Bool
template<typename F,typename S> F p(std::pair<F,S> p){return p.first;}
template<typename F,typename S> S q(std::pair<F,S> p){return p.second;}

struct c_prime
{
 int v1;bool v2; char v3;
};
int p_prime(prime bp){ return bp.v1; }
int q_prime(prime bp){ return bp.v2; }
```



- Miért nem a *c'* a szorzat típus?

Algebrai típusok C++-ban

- Szorzatnak az *std::pair* felel meg

```
using c = std::pair<int,bool>; // c tipus a peldaban,a = Int, b = Bool
template<typename F,typename S> F p(std::pair<F,S> p){return p.first;}
template<typename F,typename S> S q(std::pair<F,S> p){return p.second;}

struct c_prime
{
 int v1;bool v2; char v3;
};
int p_prime(prime bp){ return bp.v1; }
int q_prime(prime bp){ return bp.v2; }
```

- Miért nem a *c’* a szorzat típus?
  - ha m és m’ is létezik és egyediek, akkor a két típus megegyezik izomorfizmus erejéig
  - ha csak m léteik és egyedi, c az univerzális konstrukció
  - ha csak m’ létezik és egyedi, c’ az univerzális konstrukció

Algebrai típusok C++-ban

- m:

```
c m(c_prime bp){return {bp.v1,bp.v2};}
```

Algebrai típusok C++-ban

- m:

```
c m(c_prime bp){return {bp.v1,bp.v2};}
```

- m':

```
c_prime m_prime(c p){return {p.first,p.second,'a'};}
c_prime m_prime_2(c p){return {p.first,p.second,'b'};}
c_prime m_prime_3(c p){return {p.first,p.second,'c'};}
//…
```

Algebrai típusok C++-ban

- m:

```
c m(c_prime bp){return {bp.v1,bp.v2};}
```

- m’:

```
c_prime m_prime(c p){return {p.first,p.second,’a’};}
c_prime m_prime_2(c p){return {p.first,p.second,’b’};}
c_prime m_prime_3(c p){return {p.first,p.second,’c’};}
//…
```

- m’ nem egyedi, ezért az std::pair a szorzat típus!

Típus algebra

- Szorzat és összeg művelet monoidális struktúrákat alkotnak a típusrendszerben, tulajdonságok:
  - léteznek egységelemek (***Unit*** és ***Void)***
  - asszociatív (izomorfizmus erejéig)
  - kommutatív is(izomorfizmus erejéig)
  - a szorzás disztributív

- A két művelettel a típusrendszer additív inverz nélküli gyűrűt alkot

- pl.: a szorzás művelte disztributív (a betűk akár típusokat is jelölhetnek…)
  - $a \times (b + c) \cong a \times b + a \times c$

Gyakorlati feladat

- Implementáljuk az összeg típust C++-ban
- Mutassuk meg, hogy a Void és a Unit az identitáselem
  - $a + 0 = a = 0 + a$
  - $a \times 1 = a = 1 \times a$
- Mutassuk meg, hogy a szorzat disztributív :
  - $a \times (b + c) \cong a \times b + a \times c$

Típus algebra

- Tetszőleges algebrai kifejezés megfogalmazható, például

Szám	Típus
0	Void
1	Unit
A + B	Either<A,~~A>~~B>
A x B	Pair<A,B>
1 + A	Either<Unit,A> = (Maybe<A>)

- $L = \sum_{n=0}^{\infty} q^n$; mi is lehet ez ?

Típus algebra

- $L = \sum q^n = \frac{1}{1-q}$ $/(1-q)$

$L \cdot (1-q) = 1$

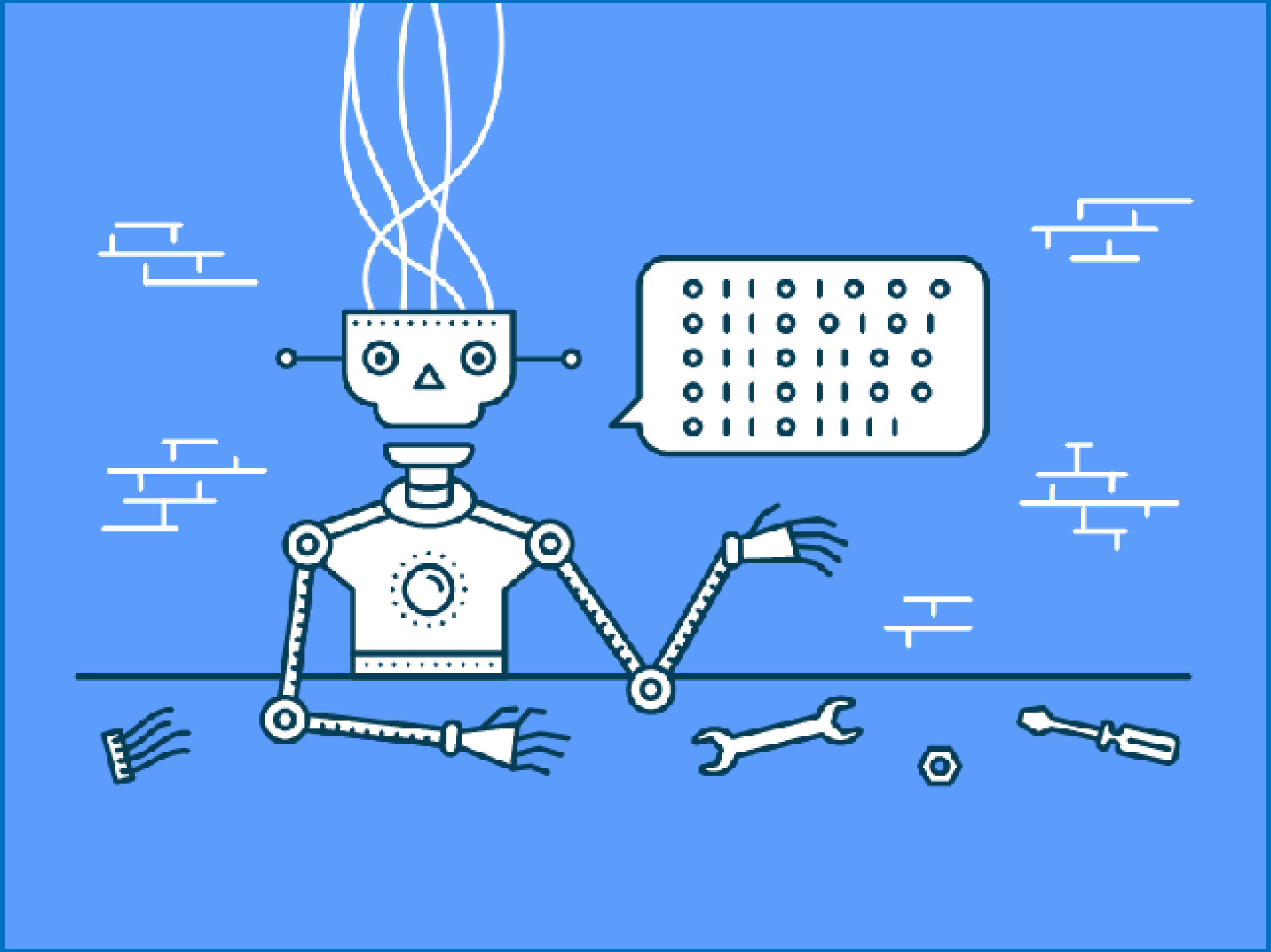
$L - q \cdot L = 1$

$L = 1 + q \cdot L = 1 + q(1 + q \cdot L) = 1 + q + q \cdot q \cdot L$

$L = 1 + q + q \cdot q(1 + q \cdot L) = 1 + q + q q + q \cdot q q L$

$L = \overset{\text{Unit}}{1} + q + qq + qqq + qqqq + \ldots\ldots$ Lista

ex_0: Maybe



A folytatásban…

- Funktorok
- Természetes transzformációk
- Monádok
- És ezen fogalmak bemutatás C++-ban …

Köszönöm a figyelmet!

Folytatjuk…