1. Bevezető

Bevezetés

- Halmazelmélet
- Formális rendszerek
- Logika
- λ kalkulus
- Típuselmélet
- Absztrakt algebra
- Kategóriaelmélet
- Funkcionális programozás

Bevezetés

- Halmazelmélet
- Formális rendszerek
- Logika
- λ kalkulus
- Típuselmélet
- Absztrakt algebra
- Kategóriaelmélet
- Funkcionális programozás

Kicsit lazán kötődő dolgok először, de ezen fogalmak egy része állandóan elő fog kerülni, a másik felét pedig implementálni fogjuk☺

Halmazelmélet

- Alapja a tartalmazási reláció: $x \in H$
- Műveletek:
  - Unió: $A \cup B$
  - Metszet: $A \cap B$
  - Különbség: $A \setminus B$
  - Szimmetrikus különbség: $A \triangle B = (A \setminus B) \cup (B \setminus A)$
  - Descartes szorzat: $A \times B$
  - Hatványhalmaz: $\mathcal{P}(A)$ (A minden részhalmazának halmaza)

Halmazelmélet

- Spec halmaz: az üres halmaz: $\emptyset$

- Definiálás: Set-builder notation: $\{x \mid \Phi(x)\}$
  Ahol x változó, Φ pedig egy igaz/hamis értéket adó kifejezés. Az igaz értéket adó x-ek az elemei az eredmény halmaznak.
- Megjegyzés a Descartes-szorzathoz:

$$A \times B = \{ (a, b) \mid a \in A \ \wedge \ b \in B \}$$

Azaz $A \times A$ elemei (a, a) párok: meg tudjuk duplázni az elemeket…

Halmazelmélet

- Halmazok halmazai…

Csomó későbbi paradoxon származik abból, ha feltesszük, hogy a halmazok halmazai is automatikusan halmazok.

Ennek elkerülésére:

- Class (osztály): halmazok „gyűjteménye”, de maga nem halmaz! Van egy tulajdonság, ami minden elemére igaz.
  - Néha szigorúbban: az elemei semmilyen más halmaznak nem elemei (proper class).

Snitt…

Formális rendszerek

- Formális rendszer: kicsit nehezen definiálható fogalom:
  az absztrakt gondolkodásnak, matematikának egy rendszere.
- Tartozik hozzá egy formális nyelv, ami ezt reprezentálja, aminek elemei:
  - Szimbólumok véges sokasága
  - Nyelvtan: ami megmondja, hogy a szimbólumokból hogyan lehet értelmes kifejezéseket (well-formed formulas) építeni,
    illetve egy adott kifejezés értelmes-e az adott rendszerben
  - Axiómák egy véges halmaza
    mindegyik axiómának értelmes formulának kell lennie.
  - Inferencia szabályok, átalakítási szabályok:
    olyan „függvények”, amik egy értelmes kifejezésekből
    másik értelmes kifejezést hoz létre.

Logika

- Propositional Logic (Ítéletlogika):
  - Adottak propozíciók, ítéletek (állítások), amik tovább nem bonthatóak.
  - Ezeknek két „értéke” lehet: igaz vagy hamis.
  - Ezekből építünk bonyolultabb kifejezéseket, a logikai műveletekkel (connectives, junktorok):
    - Negáció ($\neg$)
    - És ($\wedge$)
    - Vagy ($\vee$)
    - Implikáció ($\rightarrow$)
    - Ekvivalencia ($\equiv, \leftrightarrow$)

Logika

- Propositional Logic (Ítéletlogika):
  - Zárt a fenti műveletekre:
    bármely művelet alkalmazása propozíciókra újra egy propozíció.
  - Következtetés (argument): propozíciók egy listája, ahol az utolsó következik az előzőekből.

- A propozíciós logika egy formális rendszer, ahol:
  - A szimbólumok az elemi propozíciókat jelölik
  - A nyelvtan a logikai junktorokból áll
  - Az axiómákat igaznak fogadjuk el
  - Az inferencia szabályok a következtetések,
    a következtetéssel kapott propozíciók a tételek

Logika

- A következtetési szabályoknál használt jelölés a turnstile: $\vdash$

  $P \vdash Q$: P maga után vonja Q-t, azaz ha P minden eleme egy tétel, akkor Q is az.

- A fő alkalmazása az egész konstrukciónak, hogy belássa két formuláról a következtetési szabályok sorozatos alkalmazásával, hogy ekvivalensek-e. Ez a bizonyítás.

- Egy ilyen logikai rendszerről (következtetési szabály rendszerről) két dolgot célszerű bizonyítani:
  - Soundness (értelmesség): a szabályok nem mondanak ellent
  - Teljesség: nincs szükség más szabályra, hogy bármilyen bizonyítást végigvihessünk

Logika

Kiterjesztések:

- Elsőrendű logika:
  logikai függvények: pl.: P(x), Q(x, y)
  kvantifikációk: hivatkozás elemi objektumok egy csoportjára, amelyek alanyai a propozícióknak, de maguk nem lehetnek függvények, vagy propozíciók
  - Egzisztenciális: $\exists x(P(x))$: létezik olyan x, hogy P(x) igaz,
  - Univerzális: $\forall x(P(x))$: minden x-re P(x) igaz.

- Magasabb rendű logika:
  A kvantifikáció propozíciók illetve függvények felett is történhet, illetve ezek halmazai felett is!

Snitt…

λ-kalkulus

- Formális rendszer, amely csak a függvény absztrakciót és alkalmazását valósítja meg.
- Úttörői: Alonso Church
  és tanítványai: Stephen Kleene és John Barkley Rosser

- A nyelvtan elemei:
  - Változó szimbólum, pl.: x
  - Lambda absztrakció, pl.: $\lambda x.x^2$
  - Lambda alkalmazás, pl.: $f\ x$, tehát pl.: $(\lambda x.x^2)\ 4 \rightarrow 16$

λ-kalkulus

Tulajdonságok:

- α – ekvivalencia: a paraméter változó (bound variable) szimbóluma nem számít, ezek azonosak: $\lambda x.x^2 \equiv \lambda z.z^2$

- β – redukció: a függvény alkalmazás nem más, mint változó csere a lambda hasában: $\left(\lambda x.\frac{x^2+x}{2^x-x}\right)4 \equiv \frac{4^2+4}{2^4-4}$

  feltéve, hogy a behelyettesítendő változó neve nem ütközik egy a lamdbában már használatban levővel.

- η – ekvivalencia: $\lambda x.f\ x \equiv f$

λ-kalkulus

Egy „jól viselkedő” formális rendszerben vannak normál formák, amik tovább nem egyszerűsíthetőek az átírási szabályokkal.

A λ –kalkulus nem ilyen, van olyan alakzat, amire végtelen sokszor lehet alkalmazni a β – redukciót:

$$(\lambda x.\, x\ x)\ (\lambda x.\, x\ x) \rightarrow (\lambda x.\, x\ x)\ (\lambda x.\, x\ x)$$

λ-kalkulus

- A természetes számok és az alapműveletek reprezentálhatóak:

$$0 \coloneqq \lambda f.\lambda x\,.x$$

$$1 \coloneqq \lambda f.\lambda x\,.f\ x$$

$$2 \coloneqq \lambda f.\lambda x\,.f\ (f\ x)$$

$$3 \coloneqq \lambda f.\lambda x\,.f\ \big(f\ (f\ x)\big)$$

$$n \coloneqq \lambda f.\lambda x\,.f^n\ x$$

- Ezek magasabb rendű függvények; egy tetszőleges f-et n-szer alkalmaznak x-re: $3\ f\ x = f\ (\ f\ (\ f\ x)\ )$

λ-kalkulus

- 1-el növelés (succ):

  $\lambda n.\, \lambda f.\, \lambda x.\, f\ (n\ f\ x)$

- Összeadás $f^{m+n}(x) = f^m(f^n(x))$:

  $\lambda m.\, \lambda n.\, \lambda f.\, \lambda x.\, m\ f\ (n\ f\ x)$

- Szorzás: $f^{m*n}(x) = (f^n)^m\ (x)$

  $\lambda m.\, \lambda n.\, \lambda f.\, m\ (n\ f)$

- Exponencializálás:
  a def szerint: $n\ f\ x = f^n\ x$, és ebben $f \to m, x \to f$ cserével kapjuk:
  $\lambda m.\, \lambda n.\, n\ m$

- 1-el csökkentés (pred):
  $\lambda n.\, \lambda f.\, \lambda x.\, n\ (\lambda g.\, \lambda h.\, h\ (g\ f)\ )\ (\lambda u.\, x)\ (\lambda u.\, u)$

- Kivonás: $\lambda m.\, \lambda n.\, (n\ pred)\ m$

Hasonlóan a bool logika, lista műveletek is kódolhatóak

λ-kalkulus

A legérdekesebb konstrukció talán a rekurzív függvények kódolása

- Y-kombinátor:
  $\mathrm{Y} := \lambda g.\left(\lambda x.\, g\ (x\ x)\right)\left(\lambda x.\, g\ (x\ x)\right)$
- Két β – redukció után ugyanis (l. tavaly): $Y\ f = f\ (Y\ f)$
  Folytatva: $Y\ f = f\ (Y\ f) = f\ (\ f\ (Y\ f)\ )$
- Az Y-nal kompatibilis rekurzív függvényt úgy kell definiálni, hogy az első argumentuma egy függvény, ami saját maga lesz:

$$\mathrm{Factorial\_proto} := \lambda f.\lambda n.\,(n == 1\ ?\ 1 : n \times (f\ f\ (n-1))\ )$$

- A valódi faktoriálist ezek után az Y alkalmazásával kapjuk:

$$\mathrm{Factorial} := Y\ \mathrm{Factorial\_proto}$$

Pl.: $\mathrm{Factorial}\ 4\ \rightarrow 24$

λ-kalkulus

$$Y = \lambda f.\left(\lambda x.\ f\ \ (\lambda y.\ x\ x\ y)\right)\ \left(\lambda x.\ f\ \ \ (\lambda y.\ x\ x\ y)\right)$$

ha alkalmazzuk egy g függvényre, ami rekurzív: $\boldsymbol{\lambda g.\lambda a.}\,[g\ (h\ a)]$

$\lambda f.\left(\lambda x.\ f\ (\lambda y.\ x\ x\ y)\right)\left(\lambda x.\ f\ (\lambda y.\ x\ x\ y)\right)\ \boldsymbol{(\lambda g.\lambda a.}\,[g\ (h\ a)]\boldsymbol{)}\ \boldsymbol{n}$

$\left(\lambda x.\ \boldsymbol{(\lambda g.\lambda a.}\,[g\ (h\ a)]\boldsymbol{)}(\lambda y.\ x\ x\ y)\right)\left(\lambda x.\ \boldsymbol{(\lambda g.\lambda a.}\,[g\ (h\ a)]\boldsymbol{)}(\lambda y.\ x\ x\ y)\right)\ \boldsymbol{n}$

$\boldsymbol{(\lambda g.\lambda a.}\,[g\ (h\ a)]\boldsymbol{)}\left(\lambda y.\left(\lambda x'.\ \boldsymbol{(\lambda g'.\lambda a'.}\,[\ldots]\boldsymbol{)}(\lambda y'.\ x'\ x'\ y')\right)\left(\lambda x'.\ \boldsymbol{(\lambda g'.\lambda a'.}\,[\ldots]\boldsymbol{)}(\lambda y'.\ x'\ x'\ y')\right)\ y\right)\ \boldsymbol{n}$

$\boldsymbol{\lambda a.}\,[\left(\lambda y.\left(\lambda x'.\ \boldsymbol{(\lambda g'.\lambda a'.}\,[\ldots]\boldsymbol{)}(\lambda y'.\ x'x'y')\right)\left(\lambda x'.\ \boldsymbol{(\lambda g'.\lambda a'.}\,[\ldots]\boldsymbol{)}(\lambda y'.\ x'x'y')\right)\ y\right)\ (\ h\ \boldsymbol{a}\ )]\ \boldsymbol{n}$

$[\left(\lambda y.\left(\lambda x'.\ \boldsymbol{(\lambda g'.\lambda a'.}\,[\ldots]\boldsymbol{)}(\lambda y'.\ x'x'y')\right)\left(\lambda x'.\ \boldsymbol{(\lambda g'.\lambda a'.}\,[\ldots]\boldsymbol{)}(\lambda y'.\ x'x'y')\right)\ y\right)\ (h\ \boldsymbol{n})]$

$\left(\lambda x'.\ \boldsymbol{(\lambda g'.\lambda a'.}\,[\ldots]\boldsymbol{)}(\lambda y'.\ x'x'y')\right)\left(\lambda x'.\ \boldsymbol{(\lambda g'.\lambda a'.}\,[\ldots]\boldsymbol{)}(\lambda y'.\ x'x'y')\right)\ (h\ \boldsymbol{n})\ \ldots$

λ-kalkulus

A rekurziók és a normálformára redukálhatatlan kifejezések összefüggésben állnak azzal, hogy a λ – kalkulus, mint logikai rendszer inkonzisztens.

Snitt…

Történelem

Úttörők a Logika korai szakaszából: Arisztotelész, Ockham, Leibniz

A formális logika kidolgozói: Boole, De Morgan, Frege, Peirce, Peano

A 20. század elején Whitehead és Russell Principia Mathematica műve nyilvánvalóvá tette, hogy a matematika jó része kifejezhető a formális logika eszközeivel.

Ez vezette David Hilbertet, hogy kitűzze egy olyan teljes és konzisztens formális rendszer létrehozását, amelyben minden állításról „hatékonyan eldönthető”, hogy igaz, vagy hamis. (*Entscheidungsproblem)*

Történelem

Az ördög természetesen a részletekben rejlik:
Mi minősül hatékonynak? Mi az eldöntés folyamata?

Továbbá: a kitűzés feltételezi a teljességet, hogy minden állításról bizonyítható vagy az hogy igaz, vagy az hogy hamis

2000 évig nem sikerült a „hatékonyan eldönthető” definiálni, most egyszerre három definíció is született rá…

Történelem

- Church és tanítványai megalkották a λ – kalkulust. Azt állították, hogy ami ebben kifejezhető és normál formára redukálható, az minősülhet kiszámolhatónak (eldönthetőnek).
- Gödelnek ez nem tetszett, előállt a ma „általános rekurzív függvények” csoportjával. Church tanítványai megmutatták, hogy a két megfogalmazás ekvivalens.
- Alan Turing megalkotta az általános számítás modelljét a Turing gépet, amely szimbólumokat manipulál egy végtelen szalagon, megadott szabályok szerint. Kiderült, hogy az a kérdés, hogy a gép végére ér-e a feladatnak (Halting Problem) ekvivalens a fenti megfogalmazásokkal.

Történelem

A fentieknek két fontos következménye volt:

- David Hilbert programja az eldönthető logikai rendszerrel nem megvalósítható. A legvégső ítéletet Gödel nem-teljességi tételei mondták ki:
  1. Minden formális rendszerben, ami elég gazdag az aritmetika kódolásához kifejezhető olyan állítás, amit se magát, se az ellenkezőjét nem lehet bizonyítani, ezért ezen rendszerek nem teljesek.
  2. A fentihez hasonló rendszer nem tudja bizonyítani a saját konzisztenciáját.

- Valamit kezdeni kellett a paradoxonokkal, amik a logikában ellentmondásokat, a λ – kalkulusban normál formára hozhatatlan kifejezéseket eredményeztek.

Történelem

A logikában Russel paradoxona vált híressé:

- Legyen H egy halmaz, aminek elemei azok a halmazok, amik nem tartalmazzák saját magukat.
- Ha H nem eleme saját magának, akkor a definíció szerint tartalmaznia kell saját magát, ami ellentmond a definíciónak…

A probléma itt az, hogy a predikátum saját magára vonatkozik.

Az analóg eset a λ – kalkulusban az önreplikáló, önhivatkozó kifejezés...

Történelem

További fontos észrevételek derültek ki:

- Ha egy logikai rendszer elég összetett ahhoz, hogy reprezentálni tudja a szokásos számolási műveleteket, akkor nem tudja megoldani a saját Halting Problem-jét (pl. Normál formára hozást).
- Ha megszorítjuk a rendszert (pl. kivesszük a rekurziót), akkor a Halting Probléma triviális lesz (minden Normál formára hozható), de bizonyos számolásokat nem tudunk kifejezni…

Történelem

A logikai paradoxonok feloldására két törekvés vezetett sikerre:

- A halmazelméleti axiómák és a kapcsolódó formális rendszer megszorítása: ez a Zermelo–Fraenkel-féle halmazelmélet (a kiválasztási axiómával együtt ZFC), ez a modern halmazelmélet alapja.

- Típusok bevezetése, hogy elkülönítsék, hogy milyen propozíció milyen fajta elemek halmazán hathat.

Történelem

A paradoxonoknak több oka lehet:

- A nyelvi és a metanyelvi konstrukciók összekeverése: Richard paradoxon
- Korlátozatlan hivatkozás a set-builder jelölésben: Russel paradoxon
- Önmagára hivatkozás / nem létező kifejezésre hivatkozás a definícióban, amiből ellentmondás következik:
  Hazugság paradoxon: “amit mondok hazugság”, és hasonlóak.

Történelem

A paradoxonok feloldása az, hogy:

- megkülönböztetjük, hogy a predikátum milyen kifejezésekre hat, és azoknak milyen tulajdonságai vannak, ezt fejezi ki a típusuk,

- Illetve számon tartjuk a bevezetett szimbólumoknak az átírások során történő esetleges, a definíciójuk elé kerülését (scoping).

A Russel paradoxon pl. így írható:

$$R = \{x \mid x \notin x\}.$$

A $\notin$ jel két oldalán különböző típusú dolgoknak kell állniuk: az egyik a másiknak a kollekciója, halmaza, de a két oldal „rangja” különböző kell legyen. A halmazoknak a halmaza ismét egy más típus kell legyen. Ezért ugyan az a szimbólum nem értelmes a $\in, \notin$ jelek két oldalán.

Snitt

Típuselmélet

Alapvető tulajdonságok:

- A formális nyelvet a korábbiakon kívül kiegészítjük:
  minden kifejezéshez tartozni fog pontosan egy típus:

  $x : T$ például: $2 : \text{integer}$

- Egy ilyen megállapítást típus ítéletnek (typing judgement) nevezünk.
  Ha : bal oldalán összetett kifejezés áll, akkor a teljes kifejezés csak akkor értelmes, ha az alkifejezések is értelmesek és típussal elláthatóak!

- pl.: $2 + \text{true}$ nem látható el típussal…

Típuselmélet

Ahhoz, hogy a típusokat az operátorokra meg tudjuk határozni, a formális rendszert ki kell egészíteni típusítéletekkel.

Ezeket a következő formában írják: $\frac{\text{Kiindulási típusítéletek}}{\text{Eredmény típusítélet}}$

Például: $\frac{}{42 : \text{integer}}$ azaz 42 típusa egész szám. (itt nincs kiindulási feltevés)

Típuselmélet

A függvények típusának leírásához a → operátort használjuk:

Például: $\dfrac{}{+ : \text{integer} \to \text{integer} \to \text{integer}}$ (itt nincs kiindulási feltevés)

Típuselmélet

A → operátort Curry-zetten kell érteni:

A több argumentumú függvények tekinthetőek úgy, mintha minden egyes argumentum megadásakor egy egy argumentumú függvényt értékelnénk ki és egy új egy argumentumú függvényt kapnánk vissza.

Példa:

$$+ : \text{integer} \rightarrow \text{integer} \rightarrow \text{integer}$$

3+4 első lépésben:

$$(3\ +) : \text{integer} \rightarrow \text{integer}$$

majd:

$$(3+)4 : \text{integer}$$

A → operátor tehát jobbra asszociatív, azaz:

$$\text{A} \rightarrow B \rightarrow C = A \rightarrow (B \rightarrow C)$$

Típuselmélet

A függvényalkalmazás típusítélete:

$$\frac{f : A \to B \quad x : A}{f\ x : B}$$

Ez alapján az előző példa (3+4) típusának levezetése

(ahol komolyan vesszük, hogy a függvény alkalmazás csak simán az utánaírás):

$$\dfrac{\dfrac{\dfrac{}{+ : \text{integer} \to \text{integer} \to \text{integer}} \quad \dfrac{}{3 : \text{integer}}}{+\ 3 : \text{integer} \to \text{integer}} \quad \dfrac{}{4 : \text{integer}}}{+\ 3\ 4 : \text{integer}}$$

Típuselmélet

A változóknak is meg kell mondani a típusát. Ez különböző környezetben más és más lehet, ezért bevezetjük a kontextus fogalmát, ami szimbólumok típusítéleteinek sorozata:

$$\Gamma = \{x : \text{integer}, f : \text{integer} \rightarrow \text{integer}\}$$

Ekkor a típushozzárendelési jelölésünket ki kell egészíteni:

$$\Gamma \vdash e : T$$

amit úgy kell érteni, hogy e típusa T, ha az e-ben előforduló szimbólumoknak mind van típusa Γ-ban.

Az ehhez tartozó típusítélet: $\frac{}{\Gamma \vdash x : T}$, ahol x szerepel Γ-ban és típusa T.

Most minden típusítéletet át kell írnunk, pl.: $\frac{}{\Gamma \vdash 42 : \text{integer}}$

Típuselmélet

λ függvények bevezetése ($\lambda \mathrm{x}.\,\mathrm{e}$ típusa, ha x típusa T és e típusa U):

$$\frac{\Gamma, \mathrm{x} : \mathrm{T} \vdash \mathrm{e} : \mathrm{U}}{\Gamma \vdash (\lambda \mathrm{x} : \mathrm{T}\,.\,\mathrm{e}) : \mathrm{T} \to \mathrm{U}}$$

Típuselmélet

- Egyediség:
  minden kifejezésnek legfeljebb egy típusa lehet. Elképzelhető, hogy egy kifejezést nem lehet ellátni típussal, ez a típushiba.

- Értelmesség (soundness):
  ha egy kifejezés típusa meghatározható, akkor az a kifejezés kiértékelhető, és az eredmény típusa az, amit meghatároztunk.

- Teljesség (completeness):
  Minden ellátható típussal, ami nem eredményez hibát kiértékeléskor? (kifejezőképesség) Rövid olvasnivaló

Típuselmélet

Ha a korábban tárgyalt λ – kalkulust ellátjuk a bemutatott típusrendszerrel, akkor egy logikailag konzisztens rendszert (simply typed λ –calculus) kapunk, ami mindig normál formára hozható (strongly normalizing),

azonban ekkor az Y kombinátor és más önhivatkozó kifejezések nem láthatóak el típussal…

Típuselmélet

Alapvető típuskombinátorok:

- Összegtípus: $A + B$ (tagged union, variant, …)
  Vagy egy A vagy egy B típusú elemet tartalmaz
  (vagy egyikből, vagy másikból hozható létre)

- Szorzattípus: $A \times B$ (tuple, record, …)
  Egy A és egy B típusú elemet tartalmaz
  (1-1 ilyen értékből hozható létre)

A typing rule-ok megtalálhatóak pl. Benjamin C. Pierce „Types and Programming Languages” könyvében

Típuselmélet

Ha bevezetjük a típusbeli elemszám jelölésére $|A|$-t, akkor:

$$|A + B| = |A| + |B|$$

és

$$|A \times B| = |A| \cdot |B|$$

Típuselmélet

A függvény típus elemszáma az elemszámok exponenciálisával függ össze:

$$|A \to B| = |B^A| = |\mathrm{B}|^{|\mathrm{A}|}$$

amit úgy kell érteni, hogy összesen ennyi különböző függvény létezhet az adott típusokra.

Típusok deriválása: link, link

Történelem

1935: Gerhard Gentzen létrehozott egy rendszert, aminek Natural Deduction lett a neve. Az új ötlet, hogy az átírási szabályoknak Bevezetési – Eliminációs párokban kell lenniük:

- Bevezetési szabály:
  milyen körülmények (premisszák) esetén tételezhető fel egy adott logikai operátor (pl.: ha A és B is igaz, akkor írhatunk $A \wedge B$-t).

- Eliminációs szabály:
  Milyen következtetésekben lehet felhasználni az adott operátort
  (pl.: ha A és $A \wedge B$ igaz, akkor B is igaz kell legyen).

Példa: λ – absztrakció és alkalmazás pár

Ezekkel elkerülhetőek a logikai hurkok.

Történelem

1934 Haskell Brooks Curry vette észre, hogy

- A függvények típus szignatúrája implikációként is értelmezhető és ekkor minden függvénytípus egy bizonyítható állítást jelent.
- Fordítva: minden bizonyítható állításhoz létezik egy függvénytípus.
- Továbbá hasonló megfeleltetés van a kifejezések és a bizonyítások között is.

1969 William Alvin Howard tovább tudta vinni az elképzelést:

- ugyanilyen megfeleltetés van Gentzen természetes dedukciós rendszere és az egyszerű típusos λ – kalkulus között,
- valamint a kifejezések kiértékelése megfelel logikai bizonyítások egyszerűsítésének.

Történelem

Howard vizsgálta meg, hogy mi felel meg az alapvető logikai junktoroknak:

- $A \vee B$ felel meg az összegtípusnak
- $A \wedge B$ felel meg a szorzattípusnak (Descartes-szorzat)
- $A \rightarrow B$ felel meg a függvénytípusnak

Maga a tény, hogy az albizonyítások hogyan kapcsolhatóak össze komplexebbekké, már ismert volt, ez a Brouwer-Heyting-Kolmogorov interpretáció, de a konkrét kapcsolat a programozáselmélettel Curry és Howard érdeme ezért az ő nevüket viseli a:

- Curry-Howard correspondencia

Történelem

Howard az eredeti cikkében még tovább megy:

A logikai kvantoroknak is kell, hogy legyenek megfelelői a típusok terén

- Egzisztenciális és Univerzális típusok

Ma ezeket együtt dependens típusoknak nevezzük:

- Ha $\mathrm{A} : \mathcal{U}$, ahol A egy típus, $\mathcal{U}$ a típusok halmaza, akkor
- Tekinthetünk olyan hozzárendeléseket, amik: $\mathrm{B} : \mathrm{A} \to \mathcal{U}$ típusúak, azaz
- Egy $\mathrm{a} : \mathrm{A}$ értékhez egy $\mathrm{B}(\mathrm{a}) : \mathcal{U}$ típust rendelnek.

Történelem

A Curry-Howard megfeleltetés vonalán alkotta meg Jean-Yves Girard és John Reynolds a System F (vagy polimorfikus λ – kalkulus) rendszerét, majd ennek alapozását folytatta Per Martin-Löf amiből Intuitionistic Type Theory-nőtt ki.

- A Π – típusok felelnek meg az univerzális kvantifikációnak:
  Olyan dependens típus, ami egy értékhez típust rendel:
  Például a T típust tároló, n elemű tömbök halmaza

$$\prod_{\mathrm{n}:\mathbb{N}} \mathrm{Vector}(\mathrm{T},\mathrm{n}) \leftrightarrow \forall n \in \mathbb{N}\,.\,T\colon \mathcal{U} \to \mathrm{Vector}(\mathrm{T},\mathrm{n}) : \mathcal{U}$$

Ha nincs függőség, akkor a függvénytípusra egyszerűsödik.

- A Σ – típusok felelnek meg az egzisztenciális kvantifikációnak:

$$\sum_{\mathrm{n}:\mathbb{N}} \mathrm{List}(\mathrm{T},\mathrm{n}) \leftrightarrow \exists n \in \mathbb{N}\,.\,\mathrm{List}(\mathrm{T},\mathrm{n}) : \mathcal{U}$$

Ha nincs függőség, akkor szorzattípusra egyszerűsödik

Történelem

A System F baja, hogy az átírási szabályok erősen normalizálnak, de a típusítéletek nem eldönthetőek…

Általánosan: a dependens típusú nyelveknek egyensúlyozni kell a típusrendszer kifejezőképessége és eldönthetősége között.

Az egész gondolatsor következménye az volt, hogy a különböző logikai rendszerek különböző λ – kalkulusoknak felelnek meg, ez további típusrendszereket inspirált, és létrehozta a bizonyítás-segédek (proof-assistants) területét a programozáson belül.

- A System F módosításaira (Hindley-Milner) épül ma a Haskell és az ML nyelvcsalád, F#, az ötletek tovább szivárogtak a Java-ba és a C#-ba,
- A Martin-Löf elméletre a Coq, Agda, Idris és más bizonyítássegédek.

Történelem

A logika és a programozás elmélet kéz a kézben fejlődött innentől kezdve. Különböző felépítésű logikák megfeleltetését sikerült belátni különböző programozási konstrukciókkal. Egy példa:

- Modális logika:
  Megkülönbözteti a propozíciókat aszerint, hogy
  - Általában, valószínűleg igazak, vagy
  - Szükségszerűen, mindig igazak.

- Monádok a funkcionálos programozásban:
  Szekvenciális, állapottal működő és esetlegesen side-effektusokkal (exception, I/O) járó számítások reprezentálása, komponálása
  Ezekkel működik a Haskell és más funkc. Nyelvek I/O modellezése

Kiderül, hogy a kettő között hasonló megfeleltetés tehető, mint a korábbiak

Snitt

Absztrakt algebra

Az algebra vizsgálata elég messziről indult, olyan problémák motiválták, mint:

- Lineáris egyenletrendszerek megoldása → Lineáris Algebra
- Polinomiális (magasabbfokú) egyenletek megoldása → Csoportelmélet

Pl.: Moduláris aritmetika, Permutációk, Komplex számok, Mátrixok műveleti tulajdonságai.

A XX. század elején ide is betört az axiomatizáció és a „nagy kép” összerakásának igénye.

Absztrakt algebra

Axiomatizálás: mi kell egy algebrai struktúra leírásához?

- Van egy (vagy esetleg több) alaphalmaz (underlying set)
- Adott egy vagy több művelet az alaphalmaz(ok)on
- Adott néhány axióma, hogy milyen tulajdonságokat kell, hogy teljesítsenek a műveletek.

Absztrakt algebra

Minimális példa:

- Egy S halmaz, műveletek és axiómák nélkül☺

Kevésbé minimális példa:

Magma:

- Egy S halmaz felett egyetlen bináris művelet, amire zárt a halmaz.

Absztrakt algebra

A fizikában az algebrai struktúrák közül legalapvetőbb szerepet a csoportok játsszák.

Csoport:

- Olyan algebrai struktúra, amelyen egy darab kétváltozós művelet van (amit általában szorzás jellel jelölünk).

Csoport axiómák:

- Zártság: a szorzás nem visz ki a struktúrából, azaz $\forall a, b \in G \,.\, \exists a \cdot b \in G$
- Asszociativitás: $a \cdot (b \cdot c) = (a \cdot b) \cdot c$
- Egységelem: $\exists 1 \in G \;\; \forall a \in G \,.\, 1 \cdot a = a \cdot 1 = a$
- Inverz: $\forall a \in G \; \exists a^{-1} \in G \,.\, a \cdot a^{-1} = a^{-1} \cdot a = 1$

Absztrakt algebra

A csoportelmélet alkalmazási területei a fizikában sokrétűek, ami abból ered, hogy:

- A fizika alapvető struktúrái a megmaradási tételek, amelyek bizonyos mennyiségek bizonyos transzformációkkal szembeni változatlanságát fejezik ki
- A transzformációk általában éppen a csoportaxiómákat teljesítik
- Ha ismert a csoportstruktúra, akkor számos ekvivalencia relációt lehet levezetni, amikkel a modellek komplexitása jelentősen csökkenthető

Absztrakt algebra

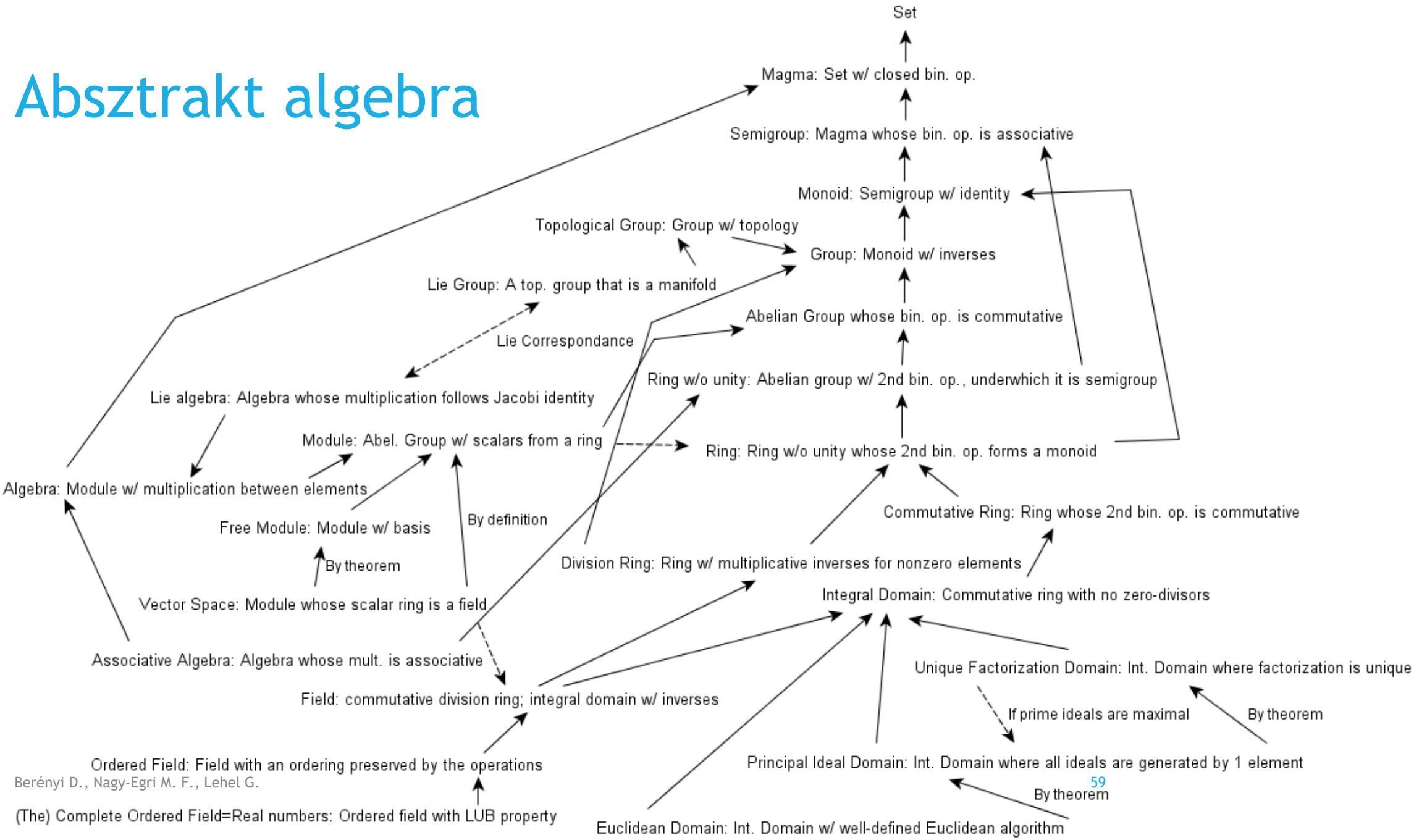
Az absztrakt algebra számos struktúra tulajdonságait feltérképezte:

Group-like structures

	Totality[α]	Associativity	Identity	Divisibility	Commutativity
Semicategory	Unneeded	Required	Unneeded	Unneeded	Unneeded
Category	Unneeded	Required	Required	Unneeded	Unneeded
Groupoid	Unneeded	Required	Required	Required	Unneeded
Magma	Required	Unneeded	Unneeded	Unneeded	Unneeded
Quasigroup	Required	Unneeded	Unneeded	Required	Unneeded
Loop	Required	Unneeded	Required	Required	Unneeded
Semigroup	Required	Required	Unneeded	Unneeded	Unneeded
Monoid	Required	Required	Required	Unneeded	Unneeded
Group	Required	Required	Required	Required	Unneeded
Abelian Group	Required	Required	Required	Required	Required

^α Closure, which is used in many sources, is an equivalent axiom to totality, though defined differently.

Absztrakt algebra



Absztrakt algebra

További fontos struktúrák még:

- A gyűrű típusúak:
  - Az S halmaz felett két bináris operátor van, $+$ és $\cdot$ jellegű, ahol az axiómák:
    - S az $+$-ra nézve ábel csoport, tehát kommutatív, asszociatív, egységelemes és inverzes
    - S a $\cdot$-ra nézve monoid: tehát asszociatív és egységelemes
    - $+$ és $\cdot$ disztributív: $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$
- Ha osztani is lehet: division ring (ferdetest)
- Ha a szorzás kommutatív és osztani is lehet: field (test)

Absztrakt algebra

Vektorterek:

- Ha F egy test akkor V vektortér F felett, ha:
  - V egy ábel csoport az összeadásra nézve
  - V-n van egy szorzásművelet $\cdot$, ami egy F és egy V-beli elemet szoroz össze és eredménye V -beli. Ez a szorzás:
    - Egységelemes V-ben
    - Kompatibilis az F-beli szorzással $*$: $a \cdot (b \cdot \vec{v}) = (a * b) \cdot \vec{v}$
    - Disztributív mind az F, mind a V-beli összeadásra nézve.

- Ha F nem test, csak gyűrű, akkor ezt module-nak (modulusnak) hívják.

Absztrakt algebra

Ha a vektortéren skaláris szorzat is van:

- $V \times V \to F$, akkor V egy normált vektortér.

Ha van értelme arról is beszélni, hogy két elem közel van egymáshoz:

- Topologikus vektortér

Ha még minden Cauchy-sorozatnak létezik határértéke:

- Teljes Topologikus Vektortér
  - Példák: Banach-terek és Hilbert-terek.
    Utóbbi igen fontos a Kvantummechanikában.

Ha a vektortéren van szorzás a vektorok között (pl. vektoriális szorzat 3D-ben), akkor algebrát kapunk. Példák: komplex számok, kvaterniók.

Absztrakt algebra

Egyetlen hátulütője van az Absztrakt Algebrai megközelítésnek:

- A struktúrákat általában önmagukban, minden mástól izoláltan tárgyalja

Absztrakt algebra

Azért ennél árnyaltabb a kép, például:

- Csoporthomomorfizmusok:
  Olyan transzformációk (függvények), amik egyik csoportból a másikba képeznek, miközben a struktúrát megtartják
  - Legyen $G(S, \cdot)$ és $H(P, *)$ két csoport
  - $\phi: G \to H$ homomorfizmus, ha teljesül hogy:
    $$\phi(g_1 \cdot g_2) = \phi(g_1) * \phi(g_2)$$
  - Ebből következik, hogy az egységelemet egységelembe, az inverzet inverzbe képzi: $\phi(g^{-1}) = \phi(g)^{-1}$

A kulcs szóösszetétel: „struktúra tartó leképezések”

Kategóriaelmélet

Az absztrakt algebrai struktúrák vizsgálatának másik megközelítése a struktúrák közötti megfeleltetések absztrakt vizsgálata.

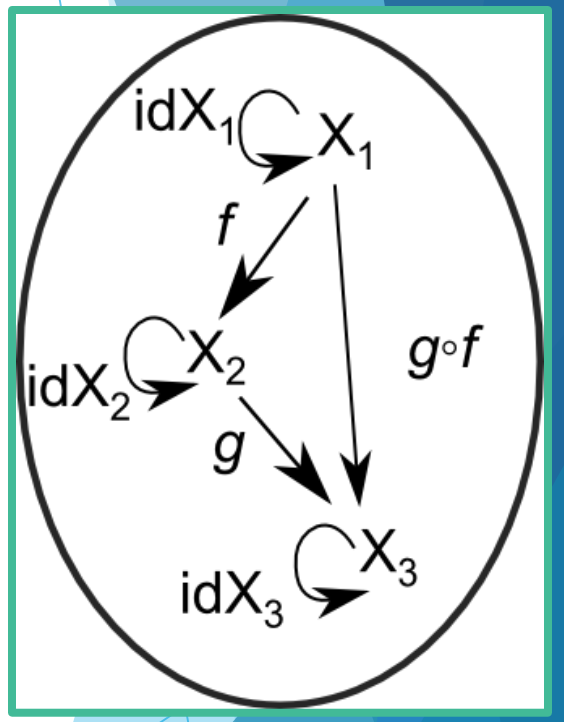
Azaz:

- hogyan képezhetünk át egyik struktúrából a másikba úgy, hogy a szerkezetet (axiómák, műveleti tulajdonságok) továbbra is érvényben maradnak?

Kategóriaelmélet

Minimális definíció: C egy kategória, ha:

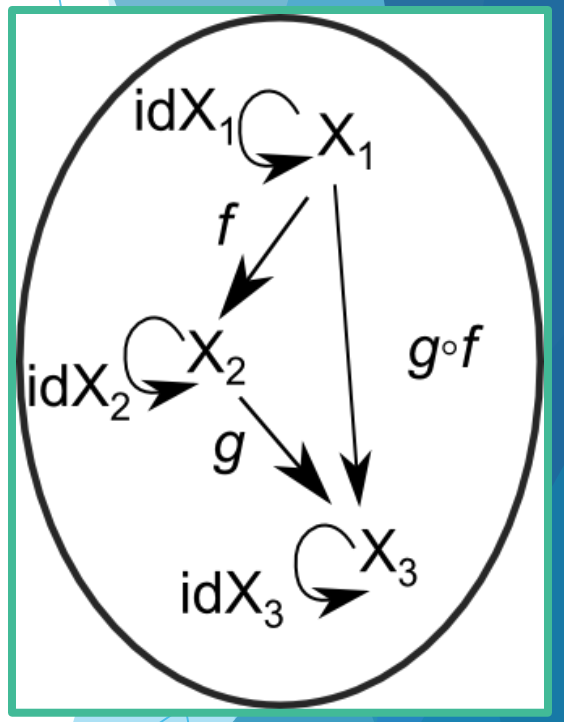
- Tartozik hozzá egy ob(C) osztály, az objektumok osztálya,
- Tartozik hozzá egy hom(C) osztály, a morfizmusok osztálya, amelyek két objektum között teremtenek irányított kapcsolatot
  - A morfizmusokon létezik egy bináris, asszociatív, egységelemes művelet, amelyet komponálásnak nevezünk.



Kategóriaelmélet

Jelölések:

- Absztrakt kategóriák maguk: $C, D, \ldots$
- Az objektumok: $X, Y, \ldots$
- A morfizmusok: $f, g, \ldots$
- Az objektumokhoz tartozó identitás morfizmus (az egységelem): $id_X, id_Y, \ldots$
- A komponálás művelete: $\circ$

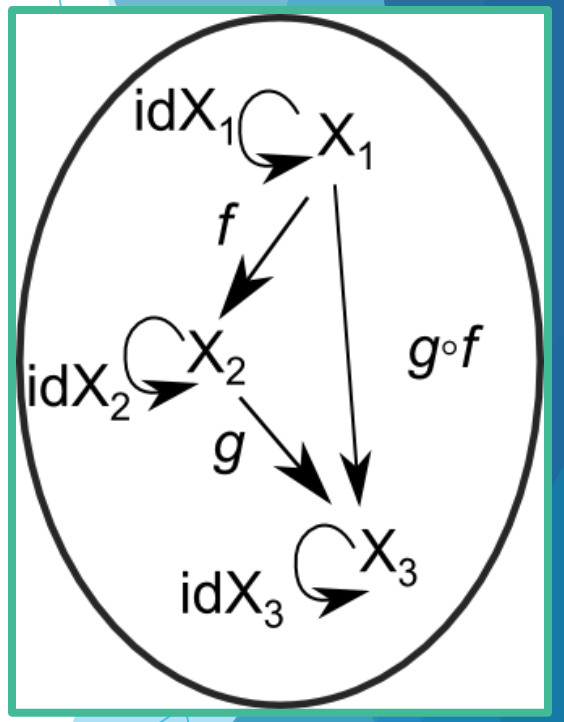


Kategóriaelmélet

Morfizmusok:

- Izomorfizmus: ha egy morfizmusnak létezik inverze, amivel komponálva az identitás morfizmust kapjuk.
- Endomorfizmus: ha a morfizmus kezdő és vég objektuma ugyan az
- Automorfizmus: a fenti kettő egyszerre

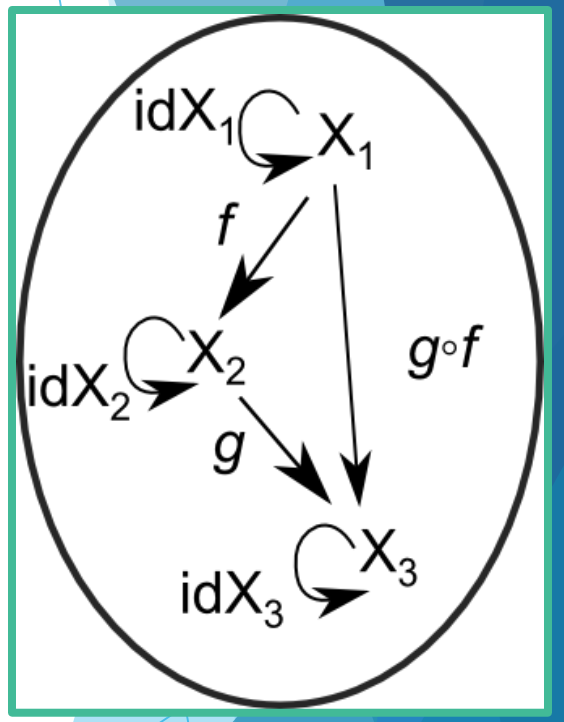
- Az összes létező morfizmus X és Y között: hom(X, Y)



Kategóriaelmélet

A konkrét kategóriákat rövid (3-4 betűs) nevekkel jelölik:

- Set:
  - A halmazok osztálya, a halmazok közötti függvényekkel, mint morfizmusokkal, ahol a függvény kompozíció a bináris művelet.



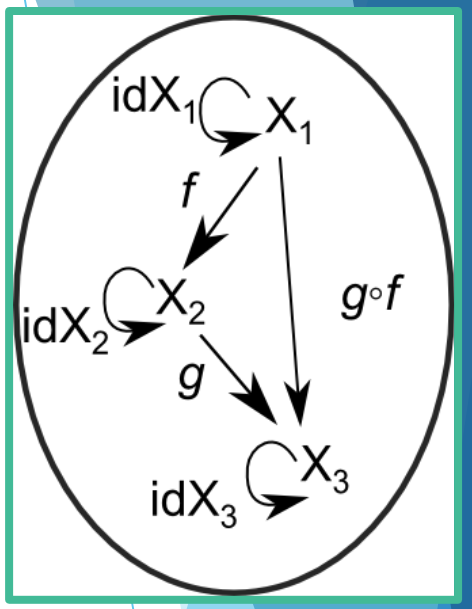
Kategóriaelmélet

További konkrét kategóriák:

Category	Objects	Morphisms
Mag	magmas	magma homomorphisms
Man^p	smooth manifolds	p-times continuously differentiable maps
Met	metric spaces	short maps
R-Mod	R-modules, where R is a ring	R-module homomorphisms
Ring	rings	ring homomorphisms
Set	sets	functions
Grp	groups	group homomorphisms
Top	topological spaces	continuous functions
Uni	uniform spaces	uniformly continuous functions
Vect$_K$	vector spaces over the field K	K-linear maps

Kategóriaelmélet

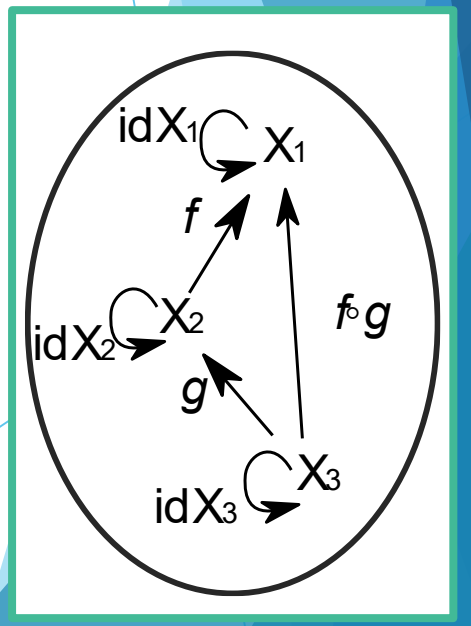
Minden kategóriához, C-hez, definiálható a duális (ellentétes) kategória, C^{op}, amiben a morfizmusok meg vannak fordítva.



Továbbá, a dualizáláskor a morfizmusok kompozíciója is sorrendet vált:

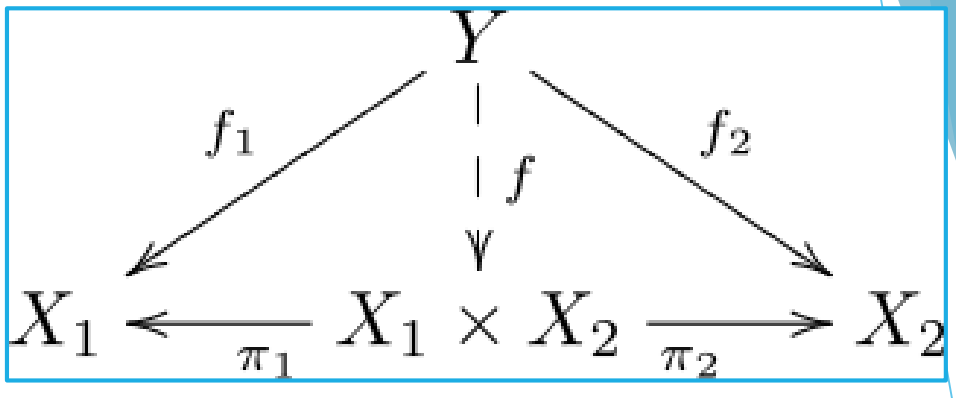
$$(f \circ g)^{op} = g \circ f$$

$$(f \circ g \circ h)^{op} = h \circ g \circ f$$



Sokszor egy fogalom duálisát a *co-* előtaggal képezzük.

Kategóriaelmélet



Szorzás:

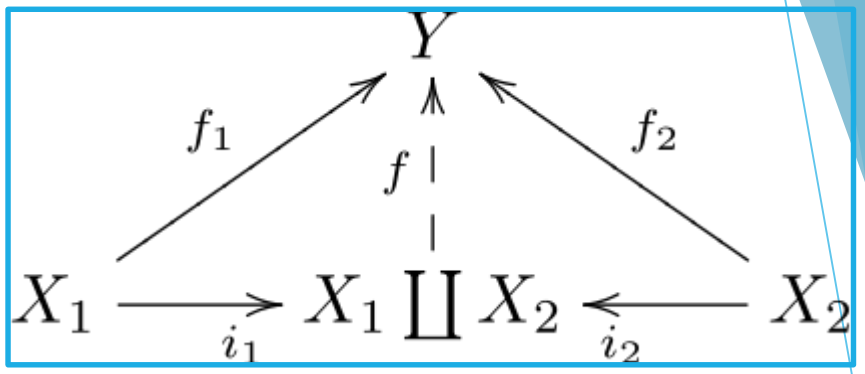
- $X = X_1 \times X_2$ ha a jelölt morfizmusok (π_1, π_2) léteznek
- és minden Y, f_1, f_2-hármasra létezik egyetlen $f : Y \to X$ morfizmus, amire a diagramm kommutál.

- f-et ekkor az f_1 f_2 morfizmusok szorzatának nevezzük.

- Ez a művelet asszociatív.

Kategóriaelmélet

Ko-Szorzás (összeadás):



- $X = X_1 + X_2$ ha a jelölt morfizmusok (i_1, i_2) léteznek
- és minden Y, f_1, f_2-hármasra létezik egyetlen $f : X \to Y$ morfizmus, amire a diagramm kommutál.

- f-et ekkor az f_1 f_2 morfizmusok ko-szorzatának, összegének nevezzük.

- Ez a művelet asszociatív.

Kategóriaelmélet

A legalapvetőbb struktúra a kategóriák között a Funktor:

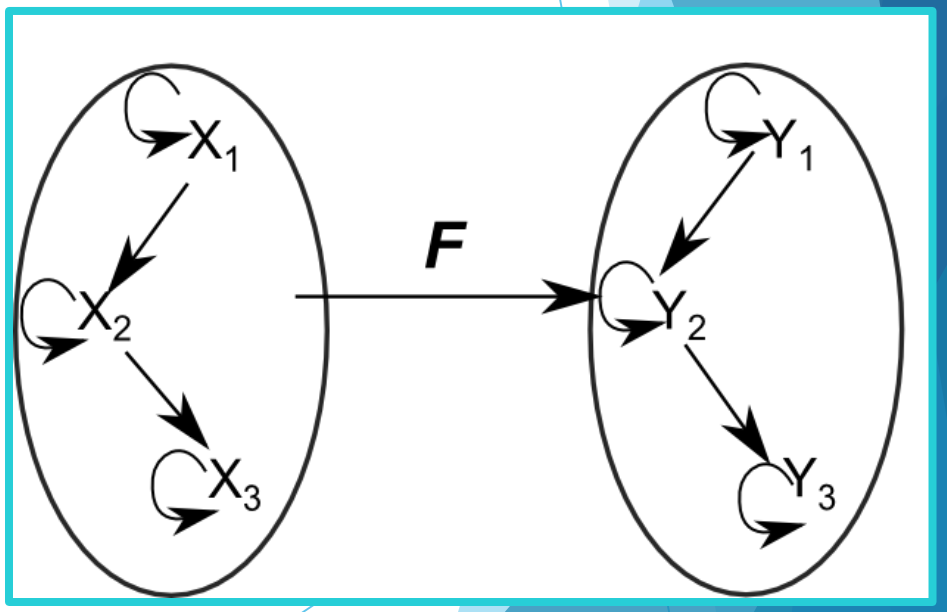
- A homomorfizmus általánosítása a kategóriákra

F egy Funktor $C \to D$ kategóriák között, ha:

- $X \in C \Rightarrow F(X) \in D$
- $f: X \to Y \in C \Rightarrow F(f): F(X) \to F(Y) \in D$

úgy, hogy:

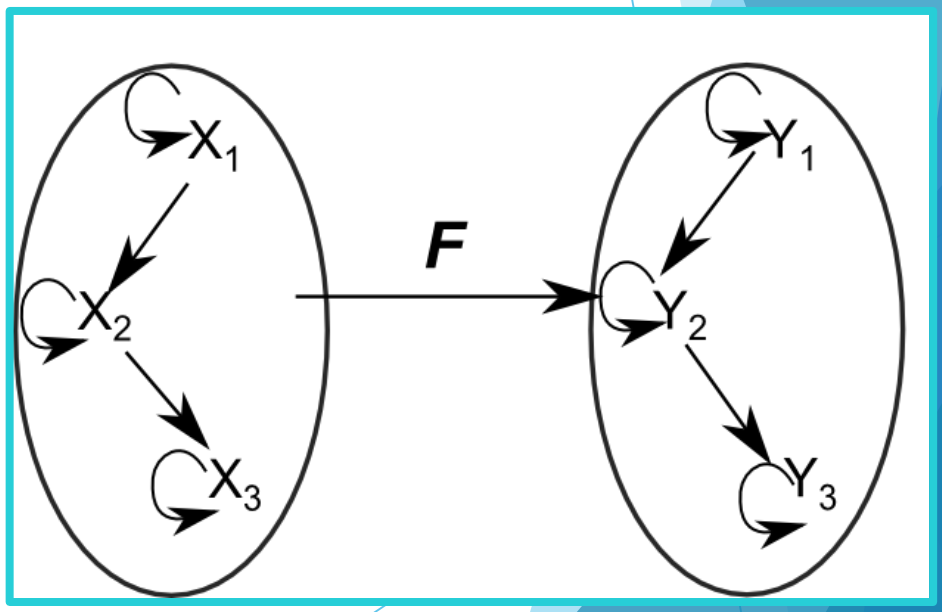
- $F(id_X) \Rightarrow id_{F(X)} \; \forall X \in C$
- $F(g \circ f) = F(g) \circ F(f) \; \forall f, g \in C$



Kategóriaelmélet

A funktor tehát úgy képzi egyik kategória objektumait és morfizmusait a másikéba, hogy:

- az egységelemhez egységelemet rendel,
- és megtartja a komponálás műveleti struktúráját



Kategóriaelmélet

A funkcionális programozásnál végtelen sok példa lesz funktorokra. Most egy fizikai példa:

- Legyen C egy olyan kategória, aminek
  - az elemei absztrakt rögzített vektortér bázisok,
  - a morfizmusok bázistranszformációk (komponálás: egymás utáni trafó hatása)
- Legyen D egy olyan kategória, ahol a fentiek koordináta reprezentációi vannak
- $F: C \rightarrow D$ Funktor, ha az absztrakt bázisokhoz a koordináta reprezentációkat rendeli úgy, hogy:
  - A bázisokhoz a bázisvektorok halmazát,
  - A transzformációkhoz a bázisvektorokat transzformáló mátrixokat rendeli, ahol a komponálás a mátrixszorzás.

Kategóriaelmélet

Ha most a választunk egy vektort, v-t, és F_v Funktor a bázisokhoz a vektor koordinátáit rendeli, a bázistranszformációkhoz a v-t transzformáló mátrixokat, akkor mást látunk:

- A bázistranszformációs mátrixoknak igazából az inverzeire van szükségünk, és fordított sorrendben kell őket szorozni:
  egy B_1 és egy B_2 transzformáció egymásutánja úgy hat a vektorra, hogy:

$$\underline{B_2}^{-1} \cdot \underline{B_1}^{-1}$$

Ha most a választunk egy lineáris funkcionált (ami tulajdonképpen a skaláris szorzat fele: egy v-t skalárba visz), ϕ-t, akkor a reprezentáció: $\underline{B_1} \cdot \underline{B_2}$

Kategóriaelmélet

A példa segítségével bevezethetünk két fogalmat:

- Egy $F: C \to D$ Funktor kovariáns, ha:
  - $f: X \to Y \in C \to F(f): F(\boldsymbol{X}) \to F(\boldsymbol{Y}) \in D$
  - $F(g \circ f) = F(\boldsymbol{g}) \circ F(\boldsymbol{f})\ \forall f, g \in C$

- Egy $G: C \to D$ Funktor kontravariáns, ha:
  - $f: X \to Y \in C \to G(f): G(\boldsymbol{Y}) \to G(\boldsymbol{X}) \in D$
  - $G(g \circ f) = G(\boldsymbol{f}) \circ G(\boldsymbol{g})\ \forall f, g \in C$

- Azaz, a kontravariáns funktoroknál megfordul a morfizmusok iránya és a komponálás sorrendje.

Kategóriaelmélet

A funktoroknak sokféle jelzőjük lehet még, egy fontos:

- Endofunktor: a kiindulási és a cél kategória ugyanaz

Kategóriaelmélet

Exponencializálás a kategóriákban:

- Legyen C egy kategória, amin létezik a korábban bevezetett szorzat az objektumok között, és $X, Y \in \mathrm{obj}(C)$ két objektum C –ben
- Az exponenciális objektum X^Y, ha létezik egy morfizmus,

$$\mathrm{ev}: X^Y \times Y \to X$$

- Ami univerzális abban az értelemben, hogy:

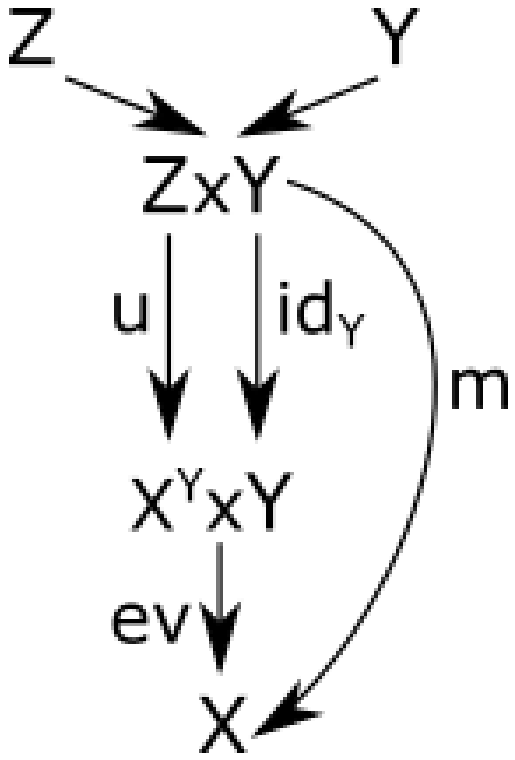
  $\forall Z \in C$-re és $\mathrm{m}: Z \times Y \to X$ morfizmusra, létezik egyetlen $\mathrm{u}: Z \to X^Y$ úgy, hogy

$$Z \times Y \xrightarrow{u \times id_Y} X^Y \times Y \xrightarrow{\mathrm{ev}} X = \mathrm{m} = Z \times Y \to X$$

Kategóriaelmélet

$$Z \times Y \xrightarrow{u \times id_Y} X^Y \times Y \xrightarrow{\text{ev}} X = \text{m} = Z \times Y \to X$$

- Ha ez $\forall X \in C$-re teljesül, akkor: $\hom(Z \times Y, X)$ és $\hom(Z, X^Y)$ között bijektív megfeleltetés van



Kategóriaelmélet

$$Z \times Y \xrightarrow{u \times id_Y} X^Y \times Y \xrightarrow{\text{ev}} X = \text{m} = Z \times Y \to X$$

- Ha ez $\forall X \in C$-re teljesül, akkor:
  $\hom(Z \times Y, X)$ és $\hom(Z, X^Y)$ között bijektív megfeleltetés van

- Funkcionális programozásban:
  - $X^Y \times Y \to X$ a $X \to Y$ függvény kiértékelését jelenti az Y argumentumra (eval, apply)
  - $Z \to X^Y$ a lambda absztrakciót, azaz $\lambda Z : Z \to X^Y$ morfizmust jelenti

Kategóriaelmélet

A Funktor struktúra egy szinttel feljebb:

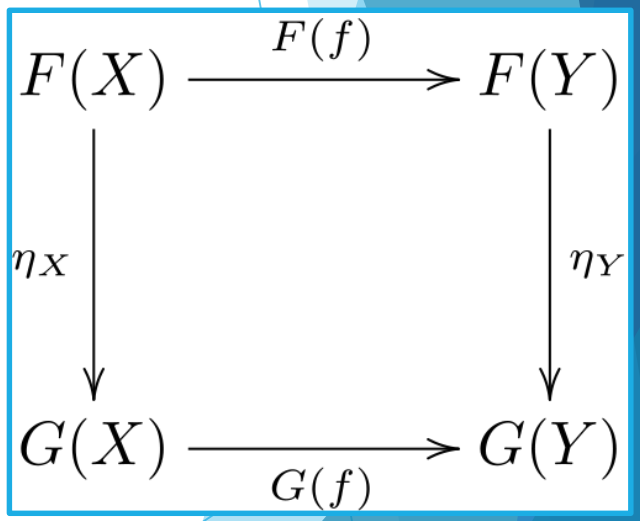
- Cat: a kategóriák kategóriája, aminek
  - Az objektumai kategóriák
  - A morfizmusai a kategóriák közti Funktorok

- A Cat morfizmusait (Funktorait) a Natural Transformation-ök viszik egymásba

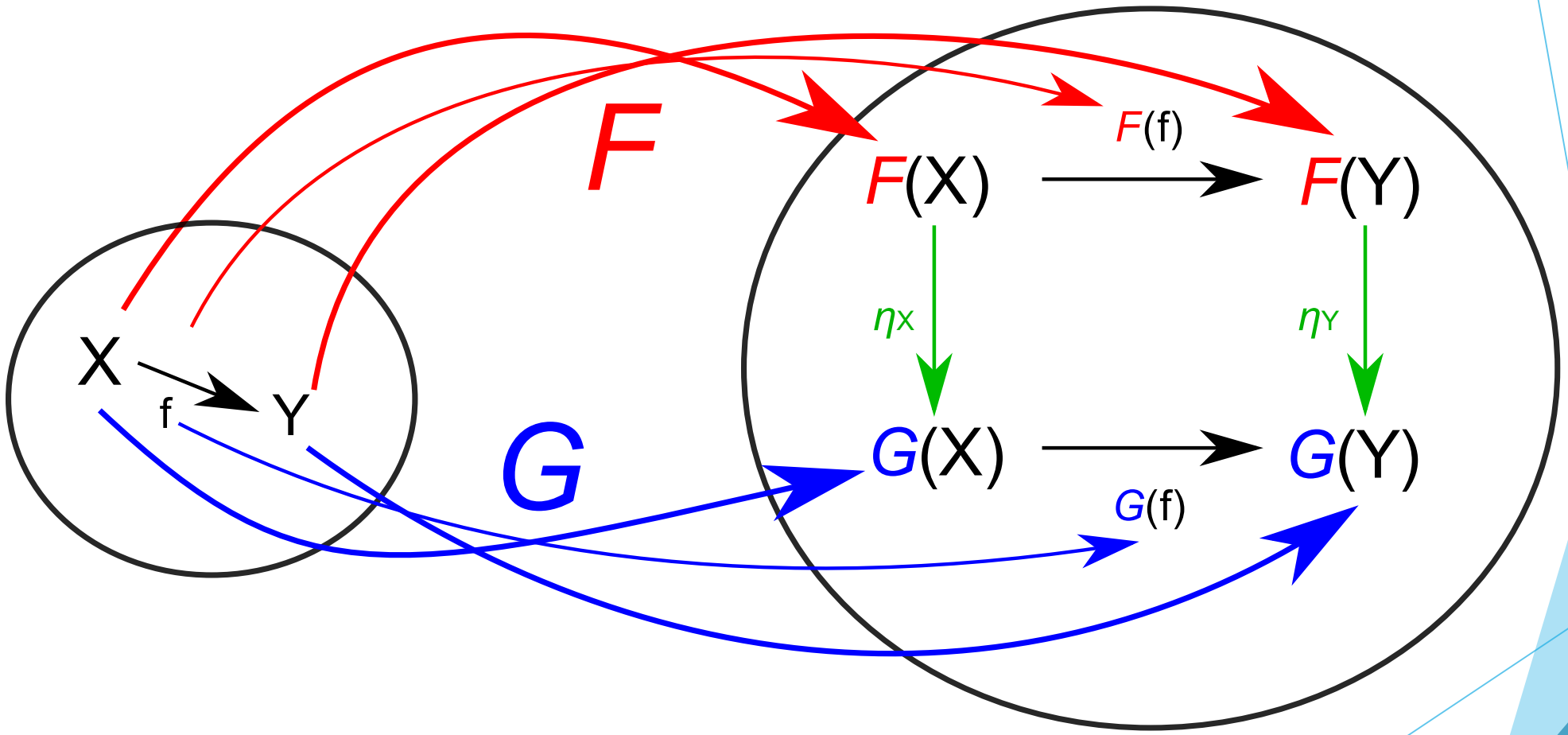
Kategóriaelmélet

Natural Transformation (NT):

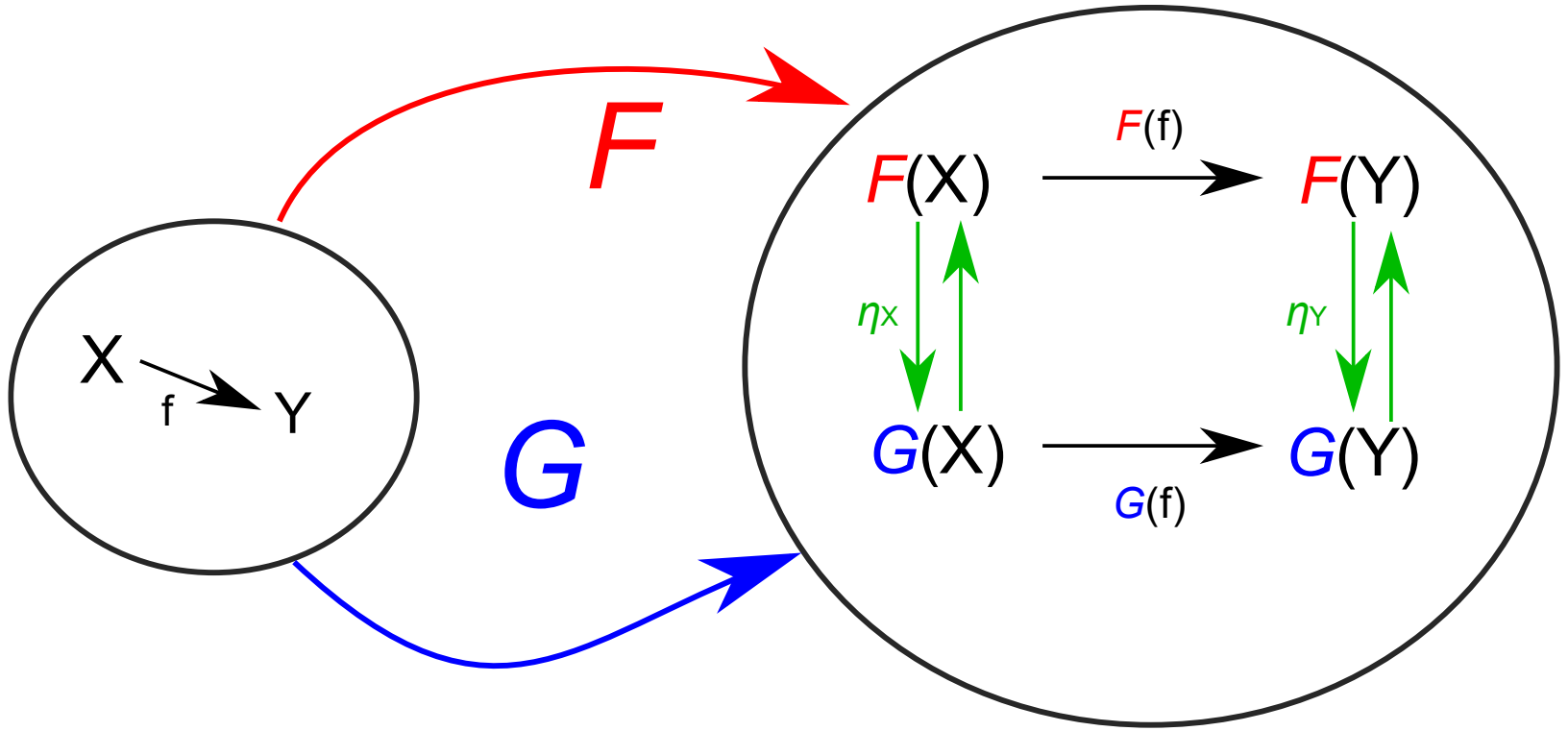
- Ha F és G két funktor C és D kategóriák között, akkor
- η egy NT, azaz morfizmusok egy osztálya, ami
  - $X \in C \Rightarrow \eta_X : F(X) \to G(X) \in D$
  - Úgy, hogy $\forall f : X \to Y \in C$-re $\eta_Y \circ F(f) = G(f) \circ \eta_X$
  - Másképp, ez a diagramm kommutál (mindegy melyik nyilakat követjük F(X)-ből):



Kategóriaelmélet



Kategóriaelmélet



Natural Isomorphism (NI):

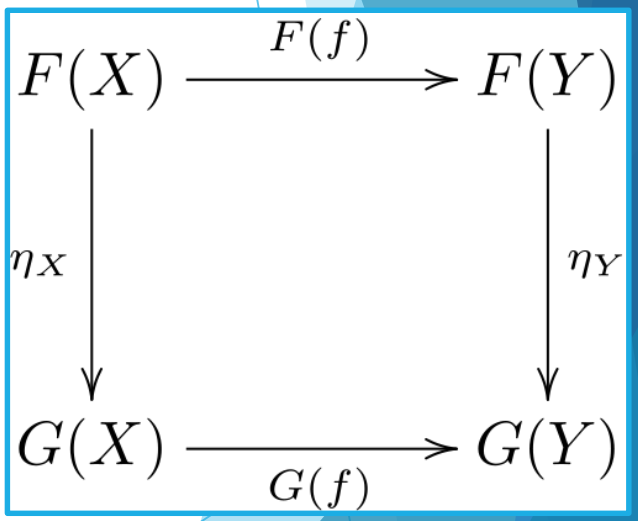
- Ha C minden elemére η képe D-ben egy izomorfizmus, akkor η egy természetes izomorfizmus

Kategóriaelmélet

Maguk a Funktorok is tekinthetőek kategóriaként (Funktor Kategória):

- Az objektumok a funktorok,
- A morfizmusok a NT-k

- Példák



Kategóriaelmélet

Cartesian Closed kategóriák (CCC):

- Ha egy kategóriának minden objektumpárjára létezik a szorzat és az exponenciális a korábban bevezetett módon, valamint
- Létezik terminális objektum, amibe minden másikból van morfizmus

Kategóriaelmélet

Monoidális kategóriák:

- Azt a struktúrát próbálják általánosítani, amit a monoidok tudnak

- Emlékeztetőül: monoid: egy S halmaz felett egyetlen bináris asszociatív művelet, amire a halmaz zárt, és létezik hozzá egységelem.

Kategóriaelmélet

Monoidális kategóriák:

- Legyen C egy kategória, amin
- $\otimes: \mathrm{C} \times \mathrm{C} \to \mathrm{C}$ szorzás egy bifunktor (2 argumentumú funktor)
- $1 \in C$ egységelem
- Továbbá létezik 3 természetes izomorfizmus, amikre két megfelelő diagramm kommutál (link),
  - ezek a szorzat asszociativitását és az egységelem viselkedését írják le.

Kategóriaelmélet

A Funktorok ereje (Functorial Strength):

- M legyen egy monoidális kategória és $F: M \to M$ endofunktor.
- F ereje az a natural transformation, hogy:
  - $v \otimes F(w) \to F(v \otimes w)$
  - úgy, hogy $u \otimes v \otimes F(w) = F(u \otimes v \otimes w)$
  - és $1 \otimes F(w) = F(w)$

Kategóriaelmélet

Ha

- C egy Cartesian Closed kategória, és még monoid struktúra is van rajta, akkor
- az, hogy F-nek létezik ereje ekvivalens azzal, hogy létezik egy olyan Natural Transformation, ami $f \in C$ morfizmusokat $F(f)$ morfizmusokba visz át.
- Másképp: $\forall f: X \to Y \in C \quad \exists F(X) \to F(Y) \in C$

Kategóriaelmélet

Ez azt jelenti, hogy CCC felett, ha az F funktornak van ereje, akkor:

- Egy $a \to b$ morfizmus szorzata a Funktoriális képével:
- $\hom(a,b) \otimes F\,(a) \Rightarrow F(\hom(a,b) \otimes a)$
- Amire viszont alkalmazva a Cartesian Closed Kategóriák exponens szabályait:
- $F(\hom(a,b) \otimes a) \Rightarrow F(b)$
- Szavakban: egy $a \to b$ függvény szorzata egy a-t transzformáló funktor képével természetesen izomorf egy b-t transzformáló funktor képével
- Funkcionális programozásban ez lesz a map (fmap) magasabb rendű függvény a következő szignatúrával:
  - $(a \to b) \to f\ a\ \to f\ b$

Olvasnivalók

- Philip Wadler
  - Propositions as Types
  - És más érdekes gyűjtemények
- Benjamin C. Pierce – Types and Programming Languages
  - Hogyan implementáljuk a simply typed λ-kalulust ÉS bizonyítsuk be a tulajdonságait Coq-ban
- Bob Coecke - Introducing categories to the practicing physicist
- Tai-Danae Bradley – What is Applied Category Theory?
- John C. Baez
  - Physics, Topology, Logic and Computation: A Rosetta Stone
  - A Prehistory of n-Categorical Physics