Množice in preslikave

Pri predmetu Logika in množice se bomo učili, kako matematiki komuniciramo in razmišljamo. Spoznali bomo osnove logike in teorije množic, tako iz povsem praktičnega vidika kot tudi matematičnega. Pri tem predmetu cenimo ne le matematično razmišljanje, ampak tudi razmišljanje o matematiki.

Za uvod povejmo nekaj osnovnega o množicah in spoznajmo nekatere osnovne konstrukcije.

Osnovno o množicah

Množice kot skupki elementov, relacija ∈

Naivno bi rekli, da je množica kakeršnakoli zbirka ali skupek matematičnih objektov. Le-ti so lahko števila, funkcije, množice, množice števil ipd., skratka karkoli.

Najbolj preprosti primeri množic so končne množice, katerih elemente naštejemo. Zapišemo jih na primer takole:

```
{1, 2, 3}
{sin, cos, tan}
{{1}, {2}, {3}}
```

Objektom, ki tvorijo množico, pravimo *elementi*. Na primer, elementi množice `{1, {4}, 7/3}` so število `1`, množica `{4}`, in število `7/4`.

Kadar je `a` element množice `M`, to zapišemo `a ∈ M` in beremo "`a` je element `M`".

Ali sta množici `{1, 4, 10}` in `{4, 10, 1, 10}` enaki? Da, saj množice obravnavamo kot *neurejene* skupke, v katerih ni pomembno, kolikokrat se pojavi kak element. Da vrstni red in število pojavitev nista pomembna, sledi iz *aksioma ekstenzionalnosti*:

> **Ekstenzionalnost množic:** *Množici sta enaki, če imate iste elemente*.

Povedano drugače: če je vsak element množice `A` tudi element množice `B` in je vsak element množice `B` tudi element množice `A`, potem velja `A = B`.

Z uporabo ekstenzionalnosti, lahko *dokažemo*, da sta `{1, 4, 10}` in `{4, 10, 1, 10}` enaki:

1. Vsak element `{1, 4, 10}` je tudi element `{4, 10, 1, 10}`:
   - velja `1 ∈ {4, 10, 1, 10}`
   - velja `4 ∈ {4, 10, 1, 10}`
   - velja `10 ∈ {4, 10, 1, 10}`
2. Vsak element `{4, 10, 1, 10}` je tudi element `{1, 4, 10}`:
   - velja `4 ∈ {1, 4, 10}`
   - velja `10 ∈ {1, 4, 10}`
   - velja `1 ∈ {1, 4, 10}`
   - velja `10 ∈ {1, 4, 10}`

Iz 1. in 2. z uporabo ekstenzionalnosti sledi, da `{1, 4, 10} = {4, 10, 1, 10}`.

Naloga: Zapiši podroben dokaz, da sta množici `{x, y}` in `{y, x}` enaki.

Opomba: Poznamo tudi skupke, pri katerih je pomembno, kolikokrat se pojavi vsak element. Imenujejo se *multimnožice*.

Opozorimo takoj, da v praksi pogosto uporabljamo zapise, ki niso povsem natančni. Takrat se zanašamo, da bodo ostali pravilni uganili, kaj imamo v mislih. Na primer, katere elemente vsebuje množica:

`` `{1, 2, 3, ..., 2021}` ``

Verjetno bi vsi "uganili" da so mišljena vsa naravna števila med `1` in `2021`, ali ne? Zavedati se je treba, da zgornji zapis tega ne določa! Morda smo imeli v mislih vsa števila med `1` in `2021`, ki pri deljenju s `5` ne dajo ostanka `4`.

Pri predmetu Logika in množice bomo pogosto opozarjali na razne nejasnosti in nenatančne zapise, ki jih uporabljajo matematiki v praksi. Ni mišljeno, da bi se pretvarjali, da je kaj narobe s "človeško matematiko". Želimo se predvsem zavedati, kje se nejasnosti v praksi pojavljajo in kako bi jih lahko odpravili (tudi če jih v praksi dejansko ne odpravimo). Ko bo torej asistent pri analizi na tablo napisal

```
1, 2, 4, 8, ...
```

imate tri možnosti:

1. Ste zmedeni.
2. Uganete, da ima v mislih potence števila 2.
3. Vprašate, ali je `n`-ti člen število regij, na katerega lahko razdelimo prostor z `(n-1)` ravninami?

Sami se odločite, kakšen odnos želite vzpostaviti z asistentom.

Prazna množica `∅`

Verjetno ni treba izgubljati besed o prazni množici. To je množica, ki nima nobenega elementa. Zapišemo jo `∅` ali `{}`.

Vprašanje: Ali je kakšna razlika med `{}` in `{∅}`?

Standardni enojec `1`

Množici, ki ima natanko en element, pravimo **enojec**.

Ali znamo pojasniti, kaj pomeni, da ima množica natanko en element, ne da bi pri tem omenili število `1` ali katerokoli drugo število? Takole: množica `A` ima natanko en element če velja:

1. Obstaja `x ∈ A`.
2. Če je `x ∈ A` in `y ∈ A`, potem `x = y`.

Naloga: Kako bi opredelili "množica ima natanko dva elementa" brez uporabe števil?

Pogosto bomo potrebovali enojec (že na naslednjih predavanjih). Seveda se ni težko domisliti enojca, na primer `{42}` ali `{sin}`. Da pa ne bomo vedno znova izgubljali časa s tem, se dogovorimo da je **standardni enojec `1`** množica `{()}`. To je zelo čudno ker:

1. Smo označili množico s številko `1` (opomba: ali ločite med "števka", "številka" in "število"?)
2. Ker je element standardnega enojca `()`, česar še nikoli nismo videli.

Glede 1. povejmo, da imamo kot matematiki *načelno svobodo* pri izbiri zapisa, a je smiselno in vljudno, da se ne zafrkavamo. Ali se torej predavatelj zafrkava, ko standardni enojec označi s številko `1`? Ne, saj gresta "enka" in "enojec" lepo skupaj, poleg tega pa bomo na naslednjih predavanjih spoznali tudi matematične razloge za tak zapis.

Glede 2. se bo kmalu izkazalo, da je zapis smiselen, ker je `()` pravzaprav "urejena ničterica".

Številske in ostale množice

Seveda si bomo privoščili uporabo raznih množic, ki jih že poznate, kot so na primer številske množice $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$ itd.

Opozorimo pa na naslednji dilemo:

- V osnovni in srednji šoli z $\mathbb{N}$ označimo množico celih števil, ki so večja ali enaka `1`.
- Pogosto v matematiki, še posebe pa v logiki, tudi število `0` obravnavamo kot naravno število. V takih primerih $\mathbb{N}$ iznačuje množico celih števil, ki so večja ali enaka `0`.

Kaj je torej prav? To je napačno vprašanje! Lahko vprašamo le "kako se bomo torej mi dogovorili". Pri tem predmetu se dogovorimo, da je `0` naravno število, ker:

1. Vadimo "matematično svobodo".
2. Imamo dobre matematične razloge, da `0` uvrstimo med naravna števila.
3. Ker je predavatelj tako zapovedal.

(Če bo kdo med odmorom predavatelja vprašate, kakšni so ti dobri matematični razlogi, zaradi katerih je zapovedal, da je `0` naravno število, bo sledila filozofska razprava, ki vam bo pokvarila odmor.)

Konstrukcije množic

Ena od osnovnih matematičnih aktivnosti so **konstrukcije**. Poznamo na primer geometrijske konstrukcije z ravnilom in šestilom. Ko računamo rešite enačbe, bi lahko rekli, da konstruiramo število, ki zadošča enačbi. Ko pišemo dokaz, konstruiramo objekt, iz katerega je razvidna resničnost neke izjave. Tudi računalniški programi so le matematični konstrukti.

Spoznajmo nekatere osnovne konstrukcije množic, se pravi, načine, kako iz množic naredimo nove množice.

Zmnožek ali kartezični produkt

Urejeni par `(x,y)` je matematični objekt, ki da dobimo tako, da združimo dva matematična objekta `x` in `y`. V srednji šoli ste večinoma pisali urejene pare števil (ki ste jih imenovali "koordinate"). V urejenem paru je vrstni red *pomemben*: urejena para `(1, 3)` in `(3, 1)` *nista* enaka. (Množici `{1, 3}` in `{3, 1}` sta enaki.)

Urejeni par `(x, y)` ima **prvo komponento** `x` in **drugo komponetno** `y`. Če imamo neki urejeni par `u`, njegovi pišemo tudi $\mathtt{pr}_1\mathtt{(u)}$ in $\mathtt{pr}_2\mathtt{(u)}$. Velja torej:

$$\mathtt{pr}_1\mathtt{(x,y) = x}$$
$$\mathtt{pr}_2\mathtt{(x,y) = y}$$

Simboloma $\mathtt{pr}_1$ in $\mathtt{pr}_2$ pravimo **prva in druga projekcija**. Običajne oznake za projekciji so tudi π_1 in π_2, v programiranju `fst` in `snd`, lahko pa tudi π_0, in π_1.

Spoznajmo sedaj **zmnožek** ali **kartezični produkt** množic `A` in `B`. Opis nove konstrukcijo množic mora navesti zapis za konstruirano množico, katere elemente ima, in kdaj sta elementa konstruirane množice enaka:

1. Zapis: zmnožek množic A in B označimo z $A \times B$
2. Elementi množice $A \times B$ so urejeni pari (x, y), pri čemer je $x \in A$ in $y \in B$
3. Enakost elemantov (princip ekstenzionalnisti za pare): $u \in A \times B$ in $v \in A \times B$ sta enaka, če velja $pr_1(u) = pr_1(v)$ in $pr_2(u) = pr_2(v)$.

Primer:

$\{1, 2, 3\} \times \{\square, \blacklozenge\} = \{(1, \square), (2, \square), (3, \square), (1, \blacklozenge), (2, \blacklozenge), (3, \blacklozenge)\}$

Iz principa ekstenzionalnisti za pare sledi, da je vrstni red v urejenem paru pomemben, saj $(1, 3) \neq (3, 1)$, ker $pr_1(1,3) = 1 \neq 3 = pr_1(3,1)$.

Zmnožek večih množic

Tvorimo lahko tudi zmnožek večih množic. Na primer, zmnožek množic $A \times B \times C$ je množica, katerih elementi so **urejene trojke** (x, y, z), kjer je $x \in A$, $y \in B$ in $z \in C$. V tem primeru imamo *tri* projekcije pr_1, pr_2, pr_3. Podobno lahko tvorimo zmnožek štirih, petih, šestih, ... množic.

Vprašanje: Ali lahko tvorimo zmnožek ene množice? Kaj pa zmnožek nič množic? Kaj bi to bilo?

Vsota ali koprodukt

Naslednja osnovna konstrukcija je **vsota** ali **koprodukt** množic A in B

- vsoto množic A in B označimo z $A + B$
- elementi množice $A + B$ so $in_1(x)$ za $x \in A$ in $in_2(y)$ za y in B.
- elementa $u \in A + B$ in $v \in A + B$ sta enaka, kadar velja
  1. bodisi za neki $a \in A$ velja $u = in_1(a) = v$
  2. bodisi za neki $b \in B$ velja $u = in_2(b) = v$

Primeri:

1. $\{1, 2, 3\} + \{\square, \blacklozenge\}\} = \{in_1(1), in_1(2), in_1(3), in_2(\square), in_2(\blacklozenge)\}$

2. $\{a, b\} + \{b, c\} = in_1(a), in_1(b), in_2(b), in_2(c)\}$

3 $in_1(b) \neq in_2(b)$.

1. Vsota *ni* unija!
   - $\{3, 5\} \cup \{3, 5\} = \{3, 5\}$
   - $\{3, 5\} + \{3, 5\} = \{in_1(3), in_1(5), in_2(3), in_2(5)\}$

Vsoti pravimo tudi "disjunktna unija", a se bo bomo temu izrazu izogibali, ker obravnavamo vsoto kot osnovno operacije in ne kot poseben primer unije.

Oznakama in_1 in in_2 pravimo **prva in druga injekcija**. Uporabljajo se tudi iznake ι_1 in ι_2, v funkcijskem programiranju `inl` in `inr`, pa tudi ι_0 in ι_1. Pravzaprav ni pomembno, kakšne oznake uporabimo, poskrbeti moralo le, da sta to različna simbola, s katerima razločimo elemente prvega in drugega sumanda.

Tvorimo lahko vsoto večih množic, na primer $A + B + C$. V tem primeru imamo tri injekcije in_1, in_2 in in_3.

Preslikave ali funkcije

Poleg množic so preslikave še en osnovni matematični pojem, ki se mu bomo posvetili (kasneje bomo spoznali še relacije).

Preslikava ali **funkcija** sestoji iz treh sestavin:

- množice, ki ji pravimo **domena**
- množice, ki ji pravimo **kodomena**
- **prirejanja**, ki vsakemu elementu domene priredi natanko en element kodomene

Če je `f` funkcija z domeno `A` in kodomeno `B`, to zapišemo

```
f : A → B
```

ali

```
  f
A → B
```

Rišemo lahko tudi diagrame, ki prikazujejo več funkcij hkrati, na primer

```
  f   g
A → B → C
        ↓h
        D
```

prikazuje tri preslikave: `f : A → B`, `g : B → C` in `h : C → D`.

V srednji šoli ste spoznavali posmične zvrsti funkcij, na primer linearne funkcije, trigonometrične funkcije, eksponentno funkcijo itd. Le-te so običajno slikale števila v števila, bile so *številske funkcije*. Mi se bomo ukvarjali s preslikavami na splošno, se pravi s poljubnimi preslikavami med poljubnimi množicami.

Princip ekstenzionalnosti preslikav

> **Ekstenzionalnost preslikav:** Če za preslikavi `f : A → B` in `g : C → D` velja `A = C`, `B = D` in `f(x) = g(x)` za vse `x ∈ A`, potem velja `f = g`.

Prirejanje in funkcijski predpisi

Prirejanje mora biti *celovito* in *enolično*. Za prirejanje z domeno `A` in kodomeno `B` to pomeni, da mora biti:

- **celovito:** vsakemu `x ∈ A` je prirejen vsaj en `y ∈ B` (priredimo vsaj en element)
- **enolično:** če sta `x ∈ A` prirejena `y ∈ B` in `z ∈ B`, potem velja `y = z` (priredimo največ en element)

(Pozor, celovitost *ni* surjektivnost in enoličnost *ni* injektivnost!)

Kako pravzaprav podamo prirejanje? Kaj to je, natančno? Čez kak mesec bomo znali odgovoriti na to vprašanje natančno, zaenkrat pa le povejmo, da je prirejanje kakeršnakoli metoda, tabela, postopek, prikaz, ali konstrukcija, ki zagotavlja celovitost in enoličnost prirejanja elementov kodomene elementom domene.

Običajni način za podajanje prirejanja je **funkcijski predpis**, ki ga pišemo

$x \mapsto \cdots$

Pri čemer je $\cdots$ neki smiselen izraz, ki določa enolično vrednost za vsak x iz domene. Spremenljivki x na levi pravimo **argument**, izrazu $\cdots$ na desni pa **prirejena vrednost**.

Primeri:

- prirejanje "prištej 7 in kvadriraj" zapišemo s funkcijskim predpisom $x \mapsto (x + 7)^2$
- prirejanje "kvadriraj in prištej 7" zapišemo s funkcijskim predpisom $x \mapsto x^2 + 7$
- prirejanje "prištej kvadrat 7" zapišemo s funkcijskim predpisom $x \mapsto x + 7^2$

Pozor: če podamo *samo* funkcijski predpis brez domene in kodomene, še nismo podali preslikave! Preslikava sestoji iz *treh* delov: domena, kodomena in prirejanje. Torej zgornji trije primeri *ne* podajajo preslikav.

Takole podamo funkcijo f iz celih v naravna števila, ki številu k priredi število $k^2 + 7$ zapišemo

$$f : \mathbb{Z} \to \mathbb{N}$$
$$f : x \mapsto x^2 + 7$$

ali

$$f : \mathbb{Z} \to \mathbb{N}$$
$$f(x) = x^2 + 7$$

ali

$$f : \mathbb{Z} \to \mathbb{N}$$
$$f = (x \mapsto x^2 + 7)$$

(Opomba: od tu naprej so zapiski še pomanjkljivi, dopolnil jih bom naknadno.)

Simbol x je vezana spremenljivka, če jo preimenujemo, se predpis ne spremeni. Naslednji funkcijski predpisi so *enaki*:

- $x \mapsto x^2 + 7$
- $y \mapsto y^2 + 7$
- $\mathrm{banana} \mapsto \mathrm{banana}^2 + 7$

Funkcijski predpis $x \mapsto 7 + x^2$ pa *ni enak* zgornjim trem (čeprav vrača enak vrednosti)!

Aplikacija in substitucija

V $x \mapsto f(x + 3) + 4$ vstavi za f vrednost $y \mapsto y^2 + 7$

Odgovor: $x \mapsto (y \mapsto y^2 + 7)(x+3) + 4$

Preslikave in prazna množica

Preslikave $\emptyset \to X$: obstaja nantanko ena taka preslikava, ki ji pravimo *prazna* preslikava. Zakaj natanko ena?

- vsaj ena: preslikava, katere predpis je $x \mapsto 42$
- ne več kot ena: za poljubni $f : \emptyset \to X$ in $g : \emptyset \to X$ velja "za vsak $x \in \emptyset$ je $f(x) = g(x)$, torej $f = g$.

Kdaj imamo preslikavo `X → ∅`?

- če je `X = ∅`, tedaj imamo natanko eno preslikavo, namreč prazno preslikavo.
- če je `X ≠ ∅`, tedaj ni nobene preslikave `X → ∅` (ker ne more biti celovita).

Identiteta in kompozicija

Identiteta na `A` je preslikava `id_A : A → A`, podana s predpisom `x ↦ x`

Kompozitum preslikav

```
  f   g
A → B → C
```

je preslikava `g ∘ f : A → C`, podana s predpisom `x ↦ g(f(x))`.

Kompozitum je asociativen, identiteta je nevtralni element.

Eksponentna množica

Tretja konstrukcija množic, ki jo bomo spoznali danes, je **eksponent** ali **eksponentna množica**:

- eksponent množic `A` in `B` označimo `B^A` (`B` na potenco `A`)
- elementi `B^A` so preslikave z domeno `A` in kodomeno `B`.

Eksponent `B^A` pišemo tudi `A → B`. To pomeni, da bi lahko namesto `f : A → B` pisali tudi `f ∈ B^A` ali celo `f ∈ A → B`, vendar je ta zadnji zapis neobičajen.

Primer: `{a, b}^{1, 2}` ima 4 elemente, namreč

1. `a ↦ 1, b ↦ 1`
2. `a ↦ 1, b ↦ 2`
3. `a ↦ 2, b ↦ 1`
4. `a ↦ 2, b ↦ 2`