4.Релация, композиция, функция, редици. Индукция - резюме

Релацията е множество от n-торки, които са подмножество на декартовото произведение.

Бинарна релация- Ако X и Y са 2 множества, то **X ↔ Y** е множество на всички бинарни релации между X и Y: X ↔ Y = P (X x Y)

Нотация за двойка (“maps to”): “↦”

Видове релации: Бинарни, хомогенни и хетерогенни, рефлексивни, симетрични, транзитивни.

Рефлексивна релация: за всяко $x \in X$ е изпълнено $x\ \alpha$ x ∈R - **всеки елемент на множеството X е свързан със себе** си чрез релацията R.

Симетрична релация: R={(1,2),(2,1)} - всяка връзка (x, y) има своята обратна (y, x) т.е. връзките между елементите са двупосочни.

Ако R е релация от типа X↔Y, то:

-**Област (domain), Source** на дадената релация R е множество от елементи от X, които са свързани с елементи от Y

-**Обхват (range)**, **Target** на дадената релация R е множество от елементи Y, с които елемент от X е свързан

Част от областта (domain): Ако A ⊆ X, то A◁ R

Част от обхвата (range): Ако B ⊆ Y, то R ▷ B

Изваждане от областта (**X \ A**) или A⩤ R - всички елементи от областта X, които не принадлежат на множеството A

Изваждане от обхвата (**Y \ B**) или B ⩥ R - всички елементи от обхвата Y, които не принадлежат на множеството B - Това означава, че от релацията R се премахват всички двойки (x,y), при които y∈B. Резултатът е нова релация, в която елементите от обхвата, принадлежащи на B, са изключени.

Образ на A в релацията R **| A | = ran(A◁ R)** всички елементи, с които елементите на A са свързани чрез *R*:

X={1,2,3,4}, Y={a,b,c,d} R={(1,a),(2,b),(3,c),(4,d)}

A={1,3} A⊲R={(1,a),(3,c)} **Образ на A в R**: |A|=ran(A⊲R)={a,c}

Обратна релация R~

Композиция (Composition) - Ако **източникът (source) на релация R2 е цел (target) на друга релация R1**, то двете релации могат да формират нов обект, наречен **композиция на две релации (R1°R2)**. Знакът ○, както и ⨾, се използват за означаване на композиция.

x ↦ z∈R1°R2 ⇔∃y :Y •x↦ y∈R1∧ y ↦ z∈R2

Композицията на две релации свързва началния елемент на първата релация с крайния елемент на втората релация, ако съществува **междинен** елемент, който ги свързва последователно.

Функция- релация, при която елемент от едно множество е свързан с **най-много един** елемент от друго множество-**няма двойки, съдържащи еднакъв първи елемент**.

Видове функции: Тотални, частични, инекции, сюрекции, биекции.

→ **за всяко x ∈ A** съществува **едно y ∈ B** f е **тотална функция**: **A→B**

⇸ **всяко x ∈ A** съществува **най-много едно y** ∈ B - **частична функция**:**A ⇸ B**(elements of A that are not related to any element of B)

⇻ функция за **крайно множество** от стойности на **A**– **крайна**: **A**⇻ **B**

↣ **"one-to-one"** – **инекция** - X ↣Y (**покрива всички елементи в X**)

↠ "**onto**" - **сюрекция** и X ↠ Y (**покрива всички елементи на Y**)

⤖ **one-to-one и onto - биекция** - X⤖ Y

Ако **a** е елемент от **dom** на функцията f, то записът **f(a)** означава единственият **елемент, който е резултат от приложението на функцията върху a.** Две правила за извод: 1) Ако ∃ **единствена двойка a ↦ b ∈ f с първи елемет a и втори елемент b, то b = f(a).** **2**/Ако **b = f(a) и** ∃ **единствена двойка с първи елемет a, то a↦ b∈ f**

Ламбда нотация (λ декларация | ограничение ● резултат), където "резултат" е математически израз: **f = (λ x: T ● израз)**

f = (λ x: Z ● x^2) - Квадратична функция

Тъй като релациите са множества (от наредени двойки), то върху тях можем да приложим операторите за множества.

R1 = {(1,red), (2,blue)}

R2 = {(3,green), (2,blue)}

R1 U R2 = {(1,red), (2,blue), (3,green)}

R1 ∩ R2 = {(2,blue)}

\# (R1 U R2) = 3 - брой елементи в обединението R1 U R2

Отменяне (overriding) - често се налага да сменим стойността на функцията за една или повече стойности на областта

Новата релация **f⊕g** ще приема: стойностите на функцията f, където **f не е дефинирана в g** и ст-те на функцията g, където g е дефинирана:

$f == \{(1,red), (2,blue), (3,green)\};$ $g == \{(1,pink), (4,white)\}$

$\mathbf{f} \oplus \mathbf{g} = \{(1,\mathbf{pink}), (2,blue), (3,green), (4, \mathbf{white})\}$

Отменянето се отнася само до **стойностите на областта** на ф-те. Операторът ⊕ е приложим само към функции от **един и същи тип**.
.. "между" - дефиниране на крайни множества**2 .. 5 = {2,3,4,5}**

Редици: **Подредени** множества- **позволяват дублиране**, като всяка **позиция е важна. Празна редица:** ⟨ ⟩

Множествата не са редици. Те различават от **редиците**:

- Множествата **нямат определен ред на елементите**
- Множествата **не позволяват дублиране** на елементи, докато в редицата може да има повтарящи се елементи.

  Queue = ⟨ Rob, Peter, Mark, Mark, Matt ⟩

  seq[Queue] = {(1,Rob), (2, Peter) ...} = = {1↦ Rob, 2 ↦ Peter, ...}

  AskedQns == ⟨ Rob, Peter, Mark, Mark, Matt ⟩

  head (AskedQns) = Rob

  tail (AskedQns) = ⟨ Peter, Mark, Mark, Matt ⟩

  front (AskedQns) = ⟨ Rob, Peter, Mark, Mark ⟩

  last (AskedQns) = Matt

  # AskedQns = 5

Достъп до отделни елементи: AskedQns(1) = Rob; AskedQns 1 = Rob

Свързване (concatenation) ⟨ 1,3,1 ⟩ ^ ⟨ 3,4 ⟩ = ⟨ 1,3,1,3,4 ⟩

Обобщение - distributed concatenation (flattening) - обединяваме вложени структури в една подредена редица, премахвайки вложеността. **∧/⟨⟨a,b,c⟩,⟨d,e⟩,⟨f,g,h⟩⟩**

Достъпът до отделните елементи позволява проверка на стойността на елемент на дадена позиция.

Филтърът премахва нежеланите елементи от структурата, като запазва реда и мултипликативността на др.елементи:

⟨a, b, c, d, e, d, c, b, a ⟩ ↾ {a, b} = ⟨a, d, d, a⟩

инективни редици – редици, в които няма повтарящи се елементи

композиция на редица с функция: **S е редица** върху тип X- **s: seq[X]**. **f е функция върху елементите на X**, т.е. **f: X → Y** - f приема елементи от тип X и връща елементи от тип Y. За да приложим f към всеки елемент от s можем да запишем композицията **s ○ f** или **s ⨾ f**

Композицията s ○ f - f се прилага към всеки елемент от редицата s. **Резултатът** е нова редица, в която на всяка позиция е поставен резултатът от прилагането на f към съответния елемент от s.

Дистрибутивност - Ако имаме редица **s** и функция **f** и ако приложим **f** към редицата, можем да постигнем същия резултат, ако приложим **f** към всеки елемент от редицата поотделно и след това комбинираме тези резултати: **s∘(f∘g)=(s∘f)∘(s∘g)**

s⁀t – **конкатенация** /свързване/ на редици

Дистрибутивност - **f(s⁀t)=f(s)⁀f(t)**

Трасета (Traces) - **поредици от събития**, които се случват в система и записват действия в определен ред; **Event-** съвкупност от **събития**

Ограничения на трасетата: trace ↾ event - извличане на специфични събития като записи или четения

Bags (Чанти, multiset) е съвкупност от елементи, в която **същият елемент може да се появява повече от веднъж**

Free Types (Свободни типове) - набор от възможни стойности, които могат да бъдат използвани за създаване на типове:

colors::=red|orange|yellow|green|blue|indigo|violet

Индукция - Използва се за **доказателства върху числа, структури като бинарни дървета и редици**. Доказване **истинността на дадено твърдение за цяла категория обекти**, на база на 2 осн. принципа:

1. **База на индукцията** -Основен случай (**Base Case**):Доказва се, че твърдението е вярно за най-малкия обект от категорията.

   Пр: За естествени числа, осн. случай обикновено е за 0 или 1.

2. **Индуктивна стъпка (Inductive Step)**: - ако твърдението е вярно за някакъв обект n, то е вярно и за следващия обект $n + 1$

Видове индукция:**Натурална индукция** (Mathematical Induction) - за естествени числа; **Структурна индукция** (Structural Induction) - обекти с **рекурсивна** структура, като **дървета, списъци или редици**.

Техники/ методи за доказателство:

1. Natural Deduction (Естествен извод) -използва правила за извод за да извежда нови твърдения от вече доказани (конюнкция, импликация, дизюнкция). Базира се на формална логика и позволява да извеждаме нови твърдения от осн. предпоставки.

2. **Equational Reasoning** (Еквивалентно разсъждение) – основава се на принципа на еквивалентност. Доказват се твърдения чрез преработка на изрази, като се използват логически еквивалентни формули. Използват алгебрични правила за манипулиране на

уравнения/ изрази, за да се покаже, че два израза са идентични или че едно твърдение води до друго.

4. **Special Forms (Специални форми на доказателства)**:

-Case analysis (Анализ на случаи): - разделяме доказателството на разл.случаи и доказваме твърдението поотделно за всеки от тях. След това обединяваме резултатите, за да направим обобщение.

-One-point rule (Правило за една точка) - доказва се твърдението чрез **анализ на един конкретен случай**, който е достатъчен за доказване на общото твърдение.

-Proof by contradiction (Доказателство чрез противоречие): приемането на отрицанието на твърдението и доказването, че това води до противоречие с известни факти или аксиоми. Ако достигнем противоречие, то твърдението трябва да е вярно, тъй като неговото отрицание води до нелогичност.