Boolova algebra

Resničnostne tabele

Vsaka izjava ima **resničnostno vrednost**. Resničnostni vrednosti sta $\bot$ (resnica) in $\top$ (neresnica).

Primer: $\bot \vee (\top \Rightarrow \top)$ je resnična, njena resničnostna vrednost je $\top$.

Primer: $2 + 2 = 5$ je neresnična, njena resničnostna vrednost je $\bot$.

Kadar izjava vsebuje spremenljivke (pravimo jim tudi *parametri*), je njena resničnostna vrednost *odvisna* od parametrov.

Primer: Naj bosta $x, y \in \mathbb{N}$. Resničnostna vrednost izjave $x + y < 3$ je odvisna od x in y, kar lahko prikažemo z **resničnostno tabelo**:

x	y	$x + 2 * y < 3$
0	0	$\top$
0	1	$\top$
1	0	$\top$
2	0	$\top$
1	1	$\bot$
0	2	$\bot$
...		

Kot vidimo, je lahko takšna tabela neskončna, kar ni praktično.

V izjavi lahko nastopajo tudi **izjavne spremenljivke** ali **izjavni simboli**, to se spremenljivke, ki zavzamejo vrednosti $\bot$ in $\top$.

Primer: Naj bosta $p, q \in 2$. Tedaj je $\neg p \vee q$ izjava, katere resničnostna tabela je

p	q	$\neg p \vee q$
$\bot$	$\bot$	$\top$
$\bot$	$\top$	$\top$
$\top$	$\bot$	$\bot$
$\top$	$\top$	$\top$

Izjava $\varphi(p_1, \ldots, p_n)$, v kateri nastopajo izjavne spremenljivke $p_1, \ldots, p_n$ (in nobeni drugi parametri) določa preslikavo

$$2 \times \cdots \times 2 \to 2$$

s predpisom

$$(p_1, \ldots, p_n) \mapsto \varphi(p_1, \ldots, p_n)$$

Preslikavi, ki slika iz produkta $2 \times \cdots \times 2$ v 2 pravimo **Boolova preslikava**. Prikažemo jo lahko z resničnostno tabelo. Če ima preslikava n argumentov, ima tabela 2^n vrstic.

Tavtologije

Izjava je **tavtologija**, če je njena resničnostna vrednost $\top$ ne glede na vrednosti parametrov. Premisli: kako iz resničnostne tabele razberemo, ali je izjava tavtologija?

> **Izrek:** Naj bo φ izjava, v kateri nastopajo le izjavni simboli $p_1,\ldots,p_n$. Tedaj velja:
>
> 1. Če je φ tavtologija, potem ima dokaz.
> 2. Če ima φ dokaz, je tavtologija.

Dokaz. Dokaz najdete v N. Prijatelj: *Osnove matematične logike* (1. del).

Izrek je pomemben, ker nam pove, da lahko dokazovanje izjav v nekaterih primerih nadomestimo s preverjanjem resničnostnih tabel.

*Opomba:** Izrek velja samo za izjave, ki jih sestavimo iz izjavnih simbolov, $\bot$, $\top$ in logičnih veznikov $\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\Leftrightarrow$. Za splošne izjave, ki vsebujejo tudi $\forall$ in $\exists$ izrek *ne* velja.

Polni nabori

Vsaka formula v izjavnem računu ima resničnostno tabelo. Ali lahko vsako tabelo dobimo kot resničnostno tabelo neke formule? Na primer, ali obstaja formula, katere resničnostna tabela se glasi

p	q	?
$\bot$	$\bot$	$\bot$
$\bot$	$\top$	$\top$
$\top$	$\bot$	$\top$
$\bot$	$\bot$	$\bot$

Odgovor je pritrdilen. Na kratko povejmo, kako dobimo tako izjavo. Imamo dve množnosti.

Disjunktivna oblika: za vsako vrstico v tabeli, ki ima vrednost $\top$ zapišemo konjunkcijo simbolov in njihovih negacij, pri čemer negiramo tiste simbole, ki imajo v dani vrstici vrednost $\bot$. Na primer, v zgornji tabeli imata druga in tretja vrstica vrednost $\top$, zanju zapišemo konjukciji:

- 1. vrstica: $\neg\, p \wedge q$
- 1. vrstica: $p \wedge \neg\, q$

Nato tvorimo disjunkcijo tako dobljenih konjukcij:

$$(\neg\, p \wedge q) \wedge (p \wedge \neg\, q)$$

Dobljena formula ima želeno resničnostno tabelo.

Konjunktivna oblika: za vsako vrstico v tabeli, ki ima vrednost $\bot$ zapišemo disjunkcijo simbolov in njihovih negacij, pri čemer negiramo tiste simbole, ki imajo v dani vrstici vrednost $\top$. Na primer, v zgornji tabeli imata prva in čertrta vrstica vednost $\bot$, zanju zapišemo disjukciji:

- 1. vrstica: $p \vee q$
- 1. vrstica: $\neg\, p \vee \neg\, q$

Nato tvorimo konjunkcijo tako dobljenih disjunkcij:

`(p ∨ q) ∧ (¬p ∨ ¬q)`

Zgornjo tabelo bi lahko dobili tudi kot resničnostno tabelo formule

`p ⇔ q`

Vidimo, da lahko vsako resničnostno tabelo dobimo z uporabo veznikov ¬, ∨ in ∧. **Polni nabor** je tak izbor veznikov, k katerim lahko dobimo vsako resničnostno tabelo.

Torej je ¬, ∨, ∧ poln nabor. Lahko bi ga še zmanjšali na ¬, ∧, saj lahko

`p ∨ q`

izrazimo kot

`¬p ∧ ¬q.`

Boolova algebra

Ekvivalentni izjavi imata enake resničnostne vrednosti, torej lahko ekvivalenco ⇔ obravnavamo kar kot enakost, saj to tudi je, kar se tiče resničnostnih vrednosti. Zato lahko namesto `p ⇔ q` pišemo tudi `p = q`, če imamo v mislih le resničnostne vrednosti.

(*Opomba:* izjavi sta lahko ekvivalentni, a nimata enakega *pomena*. Na primer, izjavi `∀ x, y ∈ R . x + y = y + x` in `∀ α ∈ R . sin(2 α) = 2 · cos α · sin α` sta ekvivalentni, saj sta obe resnični, a ne moremo reči, da je njun pomen enak.)

Za logične veznike veljajo *algebrajska pravila*. Ta pravila lahko uporabljamo kot računska pravila, s katerimi lahko izjavo poenostavmi v ekvivalentno obliko. Pogosto je tako računanje bolj prikladno kot dokazovanje. Spodaj našteta pravila lahko preverimo tako, da zapišemo resničnostne tabele izjav in jih primerjamo.

Pravilom, ki veljajo za logične veznike, pravimo **Boolova algebra**.

Pravila za ⊤ in ⊥

- `⊤ ∨ p = ⊤` (⊤ absorbira ∨)
- `⊤ ∧ p = p` (⊤ je nevtralni element za ∧)
- `¬ ⊤ = ⊥`
- `⊥ ∧ p = ⊥` (⊥ absorbira ∧)
- `⊥ ∨ p = p` (⊥ je nevtralni element za ∧)
- `¬⊥ = ⊤`

Pravila za negacijo ¬

- `¬¬p = p` (negacija je involucija)
- de Morganovi pravili:
  - `¬(p ∧ q) = ¬p ∨ ¬q`
  - `¬(p ∨ q) = ¬p ∧ ¬q`

Pravila za konjukcijo in disjunkcijo

- `p ∧ q = q ∧ p` (konjunkcija je komutativna)
- `p ∧ p = p` (konjunkcija je idempotentna)
- `(p ∧ q) ∧ r = p ∧ (q ∧ r)` (konjunkcija je asociativna)
- `p ∨ q = q ∨ p` (disjunkcija je komutativna)
- `p ∨ p = p` (disjunkcija je idempotentna)
- `(p ∨ q) ∨ r = p ∨ (q ∨ r)` (disjunkcija je asociativna)

Absorbcijski pravili:

- `p ∧ (p ∨ q) = p`
- `p ∨ (p ∧ q) = p`

Distributivnostni pravili:

- `p ∧ (q ∨ r) = (p ∧ q) ∨ (p ∧ r)`
- `p ∨ (q ∧ r) = (p ∨ q) ∧ (p ∨ r)`

Ostala pravila

- `p ∨ ¬ p = ⊤` (izključena tretja možnost)
- `p ∧ ¬ p = ⊥`
- `(p ⇒ q) = (¬q ⇒ ¬p)`
- `(p ⇒ q) = ¬q ∨ p`

Ekvivalence za kvantifikatorje

Zapišimo še uporabna logična pravila za kvantifikatorje. Tokrat uporabimo ⇔ namesto =, ker je to bolj običajno.

- `(∀ x ∈ ∅ . φ(x)) ⇔ ⊤`
- `(∃ x ∈ ∅ . φ(x)) ⇔ ⊥`
- `(∀ x ∈ {a} . φ(x)) ⇔ φ(a)`
- `(∃ x ∈ {a} . φ(x)) ⇔ φ(a)`
- `(¬ ∀ x ∈ A . φ(x)) ⇔ ∃ x ∈ A . ¬ φ(x)`
- `(¬ ∃ x ∈ A . φ(x)) ⇔ ∀ x ∈ A . ¬ φ(x)`
- `(ψ ⇒ ∀ x ∈ A . φ(x)) ⇔ ∀ x ∈ A . ψ ⇒ φ(x)`
- `(ψ ∨ ∀ x ∈ A . φ(x)) ⇔ ∀ x ∈ A . ψ ∨ φ(x)`
- `(ψ ∧ ∃ x ∈ A . φ(x)) ⇔ ∃ x ∈ A . ψ ∧ φ(x)`
- `(∀ u ∈ A × B . φ(u)) ⇔ ∀ x ∈ A . ∀ y ∈ B . φ(x, y)`
- `(∃ u ∈ A × B . φ(u)) ⇔ ∃ x ∈ A . ∃ y ∈ B . φ(x, y)`
- `(∀ u ∈ A + B . φ(u)) ⇔ (∀ x ∈ A . φ(in₁(x))) ∧ (∀ y ∈ B . φ(in₂(y)))`
- `(∀ u ∈ A ∪ B . φ(u)) ⇔ (∀ x ∈ A . φ(x)) ∧ (∀ y ∈ B . φ(y))`
- `(∃ u ∈ A + B . φ(u)) ⇔ (∃ x ∈ A . φ(in₁(x))) ∨ (∃ y ∈ B . φ(in₂(y)))`
- `(∃ u ∈ A ∪ B . φ(u)) ⇔ (∃ x ∈ A . φ(x)) ∨ (∃ y ∈ B . φ(y))`
- `(∀ u ∈ {x ∈ A | ψ(x)} . φ(u)) ⇔ ∀ x ∈ A . ψ(x) ⇒ φ(x)`
- `(∃ u ∈ {x ∈ A | ψ(x)} . φ(u)) ⇔ ∃ x ∈ A . ψ(x) ∧ φ(x)`

Te ekvivalence je treba preveriti tako, da jih dokažemo.

Podmnožice in potenčne množice

Definicija relacije ⊆

Pravimo, da je množica `S` **podmnožica** množice `T`, pišemo `S ⊆ T`, ko velja `∀ x ∈ S . x ∈ T`. Pravimo tudi, da je `S` **vsebovana** v `T` in da je `T` **nadmnožica** `S`.

Vedno velja `∅ ⊆ S` in `S ⊆ S`.

Princip ekstenzionalnosti za množice pravi:

```
S = T ⇔ (∀ x ∈ S . S ∈ T) ∧ (∀ y ∈ T . y ∈ S)
```

kar lahko zapišemo s podmnožicami:

```
S = T ⇔ S ⊆ T ∧ T ⊆ S
```

Vsaka podmnožica `S ⊆ A` opredeljuje neko lastnost elementov iz `A`: tisti elementi, ki imajo opredeljeno lasnost, so v `S`, ostali pa ne.

Primer: naj bo `P` množica vseh praštevil, torej je `P ⊆ N`. Podmnožica `P` opredeljuje lasnost "je praštevilo".

Kako tvorimo podmnožice

Če je `φ(x)` logična formula, v kateri nastopa spremenljivka `x ∈ A`, lahko tvorimo množico

```
{ x ∈ A | φ(x) }
```

Pri tem je `x` vezana spremenljivka. Za to množico velja:

```
a ∈ { x ∈ A | φ(x) } ⇔ a ∈ A ∧ φ(a)
```

Povedano z besedami: elementi množice `{ x ∈ A | φ(x) }` so tisti elementi iz `A`, ki zadoščajo pogoju `φ`.

Velja `{ x ∈ A | φ(x) } ⊆ A`.

Poleg tega velja

```
{x ∈ A | φ(x)} ⊆ {x ∈ A | ψ(x)} ⇔ ∀ x ∈ A . φ(x) ⇒ ψ(x)
```

Kanonična inkluzija

Za podmnožico `S ⊆ T` definiriamo **kanonično inkluzijo** ali **kanonično vključitev** `i_S : S → T`, s predpisom `i_S : x ↦ x` (to ni identiteta, razen v primeru `S = T`!). Oznaka `i_S` ni standardna, pravzaprav standardne oznake ni.

Če je `f : T → U` in `S ⊆ T`, pravimo kompozitumu `f ∘ i_S` **zožitev* preslikave `f` na `S`, pišemo `f|_S`.

Potenčna množica

Definicija potenčne množice

Za vsako množico `A` tvorimo množico `P(A)`, ki ji pravimo **potenčna množica**. Elementi potenčne množice `P(A)` so natanko podmnožice množice `A`:

```
S ∈ P(A) ⇔ S ⊆ A
```

Primer: `P(∅) = {∅}`

Primer: `P({a,b,c}) = {{}, {a}, {b}, {c}, {a,b}, {a,c}, {b,c}, {a,b,c}}`

Karakteristične funkcije

Karakteristična funkcija na množici `A` je fukcija z domeno `A` in kodomeno `2`. Tu je `2 = {⊥, ⊤}` množica resničnostnih vrednosti.

Eksponentna množica `2^A` je torej množica vseh karakterističnih funkcij na `A`.

Opomba: karakteristične funkcije se uporabljajo tudi v analizi, kjer jih običajno razumemo kot preslikave `A → {0,1}` namesto `A → {⊥, ⊤}`. Ker sta množici `{⊥,⊤}` in `{0,1}` izomorfni, to ni bistvena razlika.

Karakteristično funkcijo si lahko predstavljamo kot preslikavo, ki opredeljuje neko lastnost elementov `A`: tisti elementi, ki imajo opredeljeno lastnost, se slikajo v `⊤`, ostali pa v `⊥`.

Primer: preslikava `p : N → 2`, definirana s predpisom

```
p(n) = ⊤, če n je praštevilo
p(n) = ⊥, če n ni praštevilo
```

je karakteristična preslikava lastnosti "je praštevilo".

Izomorfizem `P(A) ≅ 2^A`

Videli smo, da lahko neko lastnost elementov množice `A` predstavimo bodisi s podmnožico bodisi s karakteristično preslikavo. To nam da idejo, da med podmnožicami `A` in karakterističnimi preslikavami na `A` obstaja neka zveza.

Izrek: `P(A) ≅ 2^A`

Dokaz. Definirajmo preslikavi

```
χ : P(A) → 2^A
ξ : 2^A → P(A)
```

s predpisoma

```
χ_S(x) := ⊥ če x ∉ S
χ_S(x) := ⊤ če x ∈ S
```

in

```
ξ_f := {x ∈ A | f(x) = ⊤}.
```

Ta predpisa bi lahko krajše zapisali tudi takole:

```
χ_S(x) := (x ∈ S)
ξ_f := {x ∈ A | f(x) }
```

Preslikavi `χ_S` pravimo **karakteristična funkcija podmnožice S**.

Trdimo, da sta `χ` in `ξ` inverza:

1. Dokažimo `χ ∘ ξ = id_{2^A}`. Uporabimo princip ekstenzionalnosti za preslikave. Naj bo `f ∈ 2^A`.

Dokažimo, da je $\chi_{\xi_f} = f$. Uporabimo princip ekstenzionalnosti za preslikave. Naj bo $x \in A$:

$\chi_{\xi_f}(x) = (x \in \xi_f) = f(x)$.

2. Dokažimo $\xi \circ \chi = id_{P(A)}$. Uporabimo princip ekstenzionalnosti za preslikave. Naj bo $S \in P(A)$. Dokažimo, da je $\xi_{\chi_S} = S$:

   $\xi_{\chi_S} = \{x \in A \mid \chi_S(x)\} = \{x \in A \mid x \in S\} = S$ □

Boolova algebra podmnožic

Podmnožice množice A tvorijo Boolovo algebro za operaciji presek $\cap$ in unija $\cup$.

Boolova algebra množic (unija, presek, komplement).

Operacija simetrična razlika $\oplus$. Potentčna množica tvori komutativno grupo za to opreacijo.