Booleovská algebra. Podmíněný příkaz.

Booleovská funkce. Booleovské zákony. Příkaz if/else.

Tomáš Bayer | bayertom@natur.cuni.cz

Katedra aplikované geoinformatiky a kartografie, Přírodovědecká fakulta UK.

Obsah přednášky

1. Úvod

Spojité veličiny:

Proměnné veličiny, které nabývají “nekonečného” množství hodnot.

Lze je popsat spojitými proměnnými a vyjádřit reálnými datovými typy.

Diskrétní veličiny:

Proměnné veličiny, které nabývají konečného množství hodnot.

Lze je popsat diskrétními proměnnými a vyjádřit celočíselnými datovými typy.

Booleovská algebra:

Proměnné veličiny nabývají hodnot 0,1.

Hodnotu 0 lze interpretovat jako nepravda, hodnotu 1 jako pravda.

V informatice hraje velmi výraznou roli při konstrukci relací.

2. Booleovská funkce

Booleovská funkce n proměnných $f : B^n \to B$,

$$y = f(x_1, ..., x_n), \qquad x_i, y \in B.$$

Unární funkce, $n = 1$, $f : B^1 \to B$,

$$y = f(x_1).$$

Jednoprvková booleovská algebra

$$B^1 = \{0, 1\}^1.$$

Binární funkce, $n = 2$, $f : B^2 \to B$,

$$y = f(x_1, x_2).$$

Dvouprvková booleovská algebra

$$B^2 = \{0, 1\}^2 = \{0, 1\} \times \{0, 1\} = \{\{0, 0\}, \{0, 1\}, \{1, 0\}, \{1, 1\}\}.$$

Pravdivostní tabulka:

Schematické znázornění prvků a funkčních hodnot.

Sousední prvky se liší právě o 1 hodnotu.

x_1	y
x_1'	$f(x_1')$
x_1	$f(x_1)$

x_1	x_2	y
x_1'	x_2'	$f(x_1', x_2')$
x_1'	x_2	$f(x_1', x_2)$
x_1	x_2'	$f(x_1, x_2')$
x_1	x_2	$f(x_1, x_2)$

3. Unární booleovská funkce

Unární booleovská funkce $f : B^1 \rightarrow B$

$$y = f(x_1), \qquad x_1, y \in B.$$

Definiční obor unární funkce dán 2^1 hodnotami.

Existuje 2^{2^1} unárních booleovských funkcí.

x_1	$f_1(x_1)$	$f_2(x_1)$	$f_3(x_1)$	$f_4(x_1)$
0	0	1	0	1
1	0	0	1	1

Přehled unárních booleovských funkcí:

- Falsum.
- Negace.
- Aserce.
- Verum.

V programování používána negace.

4. Přehled unárních booleovských funkcí

Falsum

$$f_1(x_1) \to \{0\}.$$

Libovolné hodnotě x přiřazuje hodnotu False.

Negace

$$f_2(x_1) \quad = \quad \begin{cases} 1, & \text{pro } x_1 = 0, \\ 0, & \text{pro } x_1 = 1. \end{cases}$$

Nejznámější a nejčastěji používaná unární funkce.
Přiřazuje $f(x_1)$ opačnou hodnotu než x_1 ($\overline{x}_1$ doplňek k x_1).
Označována: $'$ $\bar{\ }$, !, not, ~.

Aserce

$$f_3(x_1) \quad = \quad \begin{cases} 1, & \text{pro } (x_1) = 1, \\ 0, & \text{pro } (x_1) = 0. \end{cases}$$

Aserce přiřazuje hodnotě $f(x_1)$ hodnotu proměnné x_1.

Verum $y = f_4(x)$

$$f_4(x_1) \to \{1\}.$$

Libovolné hodnotě x_1 přiřazuje hodnotu True.

5. Binární booleovská funkce

Booleovská funkce $f : B^2 \to B$ dvou proměnných

$$y = f(x_1, x_2), \qquad x_1, x_2, y \in B.$$

Definiční obor binární funkce dán 2^2 hodnotami.
Existuje $2^{2^2} = 16$ binárních booleovských funkcí.

Nejčastěji používané booleovské binární funkce:

- Konjunkce.
- Disjunkce.
- Negace konjunkce.
- Negace disjunkce.
- Ekvivalence.
- Nonekvivalence.
- Implikace.

V informatice používány nejčastěji konjunkce a disjunkce.

6. Přehled Booleovských funkcí v B^2

x_1	x_2	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1

x_1	x_2	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}	f_{16}
0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1

7. Konjunkce

Tzv. logický součin.

Popsána pravdivostní tabulkou

x_1	x_2	$f(x_1, x_2)$
0	0	0
0	1	0
1	0	0
1	1	1

Funkční hodnota $y = 1$, pokud proměnné $x_1 = x_2 = 1$.

Označení $\wedge$, $\cdot$, logický operátor AND.

Zápis $a \wedge b$ čteme jako "`a i b`".

V programovacích jazycích: and, &&, &.

8. Disjunkce

Tzv. logický součet.

Popsána pravdivostní tabulkou

x_1	x_2	$f(x_1, x_2)$
0	0	0
0	1	1
1	0	1
1	1	1

Poslední řádkem tabulky se disjunkce liší od “obyčejného” součtu.

Funkční hodnota $y = 0$, pokud $x_1 = x_2 = 0$..

Označení $\vee$, $+$, logický operátor OR.

Zápis $a \vee b$ čteme jako “`a nebo b`”.

V programovacích jazycích: or, $||$, $|$.

9. Negace konjunkce

Tzv. Schefferova funkce.

Popsána pravdivostní tabulkou

x_1	x_2	$f(x_1, x_2)$
0	0	1
0	1	1
1	0	1
1	1	0

Funkční hodnota $y = 1$, pokud $x_1 \neq 1 \vee x_2 \neq 1$.

Vyjádření operátorem NAND.

10. Negace disjunkce

Tzv. Pierceova funkce.

Popsána pravdivostní tabulkou

x_1	x_2	$f(x_1, x_2)$
0	0	1
0	1	0
1	0	0
1	1	0

Funkční hodnota $y = 1$ pokud $x_1 \neq 0 \wedge x_2 \neq 0$.

Vyjádření operátorem NOR.

11. Nonekvivalence

Popsána pravdivostní tabulkou

x_1	x_2	$f(x_1, x_2)$
0	0	0
0	1	1
1	0	1
1	1	0

Analogie sčítání, označení $\oplus$.

Funkční hodnota $y = 1$ pokud $x_1 \neq x_2$.

Vyjádření operátorem XOR” (eXlucive OR).

12. Implikace

Popsána pravdivostní tabulkou

x_1	x_2	$f(x_1, x_2)$
0	0	1
0	1	1
1	0	0
1	1	1

Důležité pořadí argumentů: x_1 předpoklad, x_2 tvrzení.

Lze interpretovat výrokem

“Jestliže platí x_1, pak platí x_2”.

Funkční hodnota $y = 0$, pokud 1. výrok pravdivý a 2. nepravdivý.

13. Priorita booleovských operací

Vyhodnocování výrazu z leva do prava.

Různá priorita (pořadí vyhodnocení) booleovských funkcí.

Analogie s běžnými aritmetickými operacemi.

Priorita booleovských operací:

1. Negace.
2. Konjunkce.
3. Disjunkce.
4. Ostatní.

Změna priorit operací:

Prostřednictvím závorek, obdoba aritmetických výrazů.

Příklad: platí ekvivalence?:

$f(x_1, x_2, x_3, x_4): x_1 + x_2 \cdot x_3 + x_4 \neq (x_1 + x_2) \cdot (x_3 + x_4).$

$f(x_1, x_2, x_3, x_4): x_1 \cdot x_2 + x_3 \cdot x_4 = (x_1 \cdot x_2) + (x_3 \cdot x_4).$

14. Booleovské unární zákony

Zákon dvojí negace (Double Negation Law):

$$(x')' = x. \tag{1}$$

Zákon vyloučení třetí možnosti (Complement Law):

$$x' + x = 1. \tag{2}$$

Zákon sporu (Complement Law):

$$x' \cdot x = 0. \tag{3}$$

Zákony opakování (Idempotent Law):

$$\begin{aligned} x + x &= x, &(4)\\ x \cdot x &= x. &(5)\end{aligned}$$

Zákony neutrálnosti (Identity Law):

$$\begin{aligned} x + 0 &= x, &(6)\\ x \cdot 1 &= x. &(7)\end{aligned}$$

Zákony agresivnosti (Annulment Law):

$$\begin{aligned} x + 1 &= 1, &(8)\\ x \cdot 0 &= 0. &(9)\end{aligned}$$

15. Booleovské binární zákony (1/2)

Komutativní zákony (Commutative Laws):

$$x_1 + x_2 = x_2 + x_1, \tag{10}$$

$$x_1 \cdot x_2 = x_2 \cdot x_1. \tag{11}$$

Asociativní zákony (Associative Laws):

$$x_1 + (x_2 + x_3) = (x_1 + x_2) + x_3 = x_1 + x_2 + x_3, \tag{12}$$

$$x_1 \cdot (x_2 \cdot x_3) = (x_1 \cdot x_2) \cdot x_3 = x_1 \cdot x_2 \cdot x_3. \tag{13}$$

Distributivní zákony (Distributive Laws):

$$x_1 \cdot (x_2 + x_3) = x_1 \cdot x_2 + x_1 \cdot x_3, \tag{14}$$

$$x_1 + (x_2 \cdot x_3) = (x_1 + x_2) \cdot (x_1 + x_3). \tag{15}$$

16. Booleovské binární zákony (2/2)

De Morganovy zákony (de Morgan´s Theorem):

$$(x_1 + x_2)' = x_1' \cdot x_2', \tag{16}$$
$$(x_1 \cdot x_2)' = x_1' + x_2'. \tag{17}$$

Absorpční zákony (Absorptive Law):

$$x_1 + (x_1 \cdot x_2) = x_1, \tag{18}$$
$$x_1 \cdot (x_1 + x_2) = x_1. \tag{19}$$

Zákony absorpce negace (Absorptive Law with Negation):

$$x_1 + (x_1' \cdot x_2) = x_1 + x_2, \tag{20}$$
$$x_1 \cdot (x_1' + x_2) = x_1 \cdot x_2. \tag{21}$$

Důkaz, Absorptive Law*:*

$$\begin{aligned} x_1 + (x_1 \cdot x_2) &= (x_1 \cdot 1) + (x_1 \cdot x_2), \parallel 9 \\ &= x_1 \cdot (1 + x_2), \parallel 14 \\ &= x_1 \cdot 1, \parallel 8 \\ &= x_1. \end{aligned}$$

17. Minimalizace booleovské funkce

Vycházíme z boolevských zákonů.

Každou booleovskou funkci lze vyjádřit prostřednictvím konjunkce, disjunkce, negace.

Příklad 1:

$$\begin{aligned} y = f(x_1, x_2, x_3) &= x_1' \cdot x_2' \cdot x_3' + x_1 \cdot x_2' \cdot x_3', \\ &= x_2' \cdot x_3'(x_1 + x_1'), \parallel 2 \\ &= x_2' \cdot x_3' \parallel 7. \end{aligned}$$

Příklad 2:

$$\begin{aligned} y = f(x_1, x_2) &= [(x_1 \cdot x_2)' + (x_1 \cdot x_2)]' + x_1, \\ &= [(x_1 \cdot x_2)']' \cdot (x_1 \cdot x_2)' + x_1 \parallel (18), \\ &= (x_1 \cdot x_2) \cdot (x_1 \cdot x_2)' + x_1 \parallel (2), \\ &= 0 + x_1 \parallel (4), \\ &= x_1. \end{aligned}$$

18. Tautologie

Booleovská formule, je vždy pravdivá

$$f(x_1, ..., x_n) \to \{1\}.$$

Pro libovolnou kombinaci argumentů nabývá hodnoty pravda.
Unární / binární zákony představují tautologie.

Používány při posuzování identity vztahů, důkazech, atd.
Jednoduchá tautologie (Complement Law)

$$x' + x = 1$$

Podmínky v cyklu/rekurzi: nutno se vyhnout tautologiím.
Jinak nekonečný cyklus/rekurze.

19. Kontradikce

Booleovská formule, je vždy nepravdivá

$$f(x_1, ..., x_n) \to \{0\}.$$

Pro libovolnou kombinaci argumentů nabývá hodnoty nepravda.

Kontradikce je negací tautologie.

Tautologie je negací kontradikce.

Jednoduchá kontradikce (Annulment Law)

$$x' \cdot x = 0.$$

Při navrhování podmínek v cyklu/rekurze se vyhnout kontradikcím.

Podmínka/cyklus/rekurze se neprovedou.

20. Příklad

Ověřte, zda $f(x)$ je tautologie/kontradikce:

$$\begin{aligned} f(x) &= x \cdot (x + (x + x'))', \\ &= x \cdot (x+1)' \parallel 3, \\ &= x \cdot (1') \parallel 9, \\ &= x \cdot 0 \parallel 10, \\ &= 0. \end{aligned}$$

Funkce $f(x)$ je kontradikcí.

Ověřte, zda $f(x_1, x_2)$ je tautologie/kontradikce:

$$\begin{aligned} f(x_1, x_2) &= \left[((x_1 \cdot x_2) + x_1)' \cdot (x_1 \cdot (x_2 + x_1))\right]', \\ &= \left[((x_1 \cdot x_2) + x_1)' \cdot (x_1)\right]' \parallel 20, \\ &= (x_1' \cdot x_1)' \parallel 19, \\ &= 0' \parallel 4, \\ &= 1. \end{aligned}$$

Funkce $f(x_1, x_2)$ je tautologií.

21. Stavební prvky programu

áкladní stavební prvky programu, příkazy:

- *Jednoduché příkazy*
  Základní stavební jednotka programu, elementární.
  Prázdný příkaz: `pass`
  Přiřazovací příkaz: `=`
  Příkaz skoku: `break, continue`
  Příkaz podprogramu (procedury, funkce): `def`
- *Strukturované příkazy*
  Tvořeny jednoduchými či dalšími příkazy.
  Složený příkaz (blok): odsazení
  Příkaz pro větvení programu (podmínka): `if-else, match-case`
  Příkaz pro opakování (cyklus): `for, while`

Vzájemně mohou být kombinovány.

Implementovány prakticky ve všech programovacích jazycích.

Syntaxe +- podobná.

22. Blok příkazů

Posloupnost kroků, které jsou prováděny postupně v zadaném pořadí.
Jednotlivé kroky mohu/nemusí být elementární.
Použit v případě, kdy je nutno provádět více akcí.

Označován jako složený příkaz.
Obsahuje libovolný počet příkazů.

Python: odsazení tabelátorem. : uvádí blok.

```
if x < 0:                # Nasleduje blok
    a = 10
    b = 2.0 * a
    c = a * a + b * b
```

C/C++/Java: použity {}.

```
if (x < 0)               //Zacatek bloku
{
    a = 10
    b = 2.0 * a;
    c = a * a + b * b;
}                        //Konec bloku
```

Pascal, Matlab: begin(), end().

23. Scope

Ne všechny proměnné "existují" po celou dobu programu.

U mnoha jazyků souvislost s blokem (neplatí pro Python).

Scope (Platnost):

Úsek (oblast) zdrojového kódu, kde lze proměnnou použít.

Globální proměnné v Pythonu:

Platnost v souboru + ve všech importovaných (i vně bloku).

```
a = 10                #Globalni promenna
if x < 0:
     b = 10           #Globalni promenna
print(a)              #OK, funguje
print(b)              #OK, funguje
```

Lokální proměnné v Pythonu:

Platnost v těle funkce.

```
def test():
    a = 10            #Lokalni promenna
    print(a)          #OK, funguje
....
test()
print(a)              #Nefunguje, mimo platnost
```

24. Příkazy pro větvení programu

Často označovány jako řídící struktury.

Patří k nejčastěji používaným konstrukcím.

Reagují na situace, ke kterým dochází v průběhu běhu programu.

Bývají nazývány podmíněnými příkazy: něco se koná - když.

Realizují větvení algoritmu.

O tom, která větev se vykoná, rozhoduje hodnota booleovského výrazu.

Typy podmínek:

- neúplná,
- úplná,
- kombinovaná.

Syntakticky podobné ve většině programovacích jazyků: `if-else`.

Podpora `match-case` (Python 3.10+).

25. Neúplná podmínka

Pokud booleovský výraz pravdivý, provede se příkaz/blok příkazů.

Neřeší se, co dělat v případě nesplnění podmínky.

```
if booleovsky_vyraz: #Pokud splneno, proved prikazy v bloku
   prikaz1
   prikaz2
   ...
```

V praxi tato varianta používána spíše u cyklů.

Podmínku uvádět v "kladném" tvaru, ne v negaci!

```
if x > 0:     #OK        if not(x<=10)  #Spatne
   x += 10
```

Vnořená podmínka: podmínka uvnitř těla jiné podmínky

```
if x < 0:
   if y > 0:              #Vnorena podminka
     x -= 10

if (x < 0) and (y > 0):  #Spojeni obou podminek do jedne
   x -= 10
```

26. Úplná podmínka

Říkáme, co se bude dít při splnění/nesplnění podmínky (řešeny obě situace).

Kromě podmíněného příkazu použití i u rekurze.

Vznikne rozšířením neúplné podmínky o konstrukci `else` + blok.

Blok vykonán, pokud podmínka nebude splněna (netestuje se).

```
if booleovsky_vyraz: #Pokud splneno, proved prikazy v tomto bloku
   prikaz1
   prikaz2
   ...
else:                #Jinak proved prikazy v tomto bloku
   prikaz3
   prikaz4
   ...
```

Příklad 1: Ukázka úplné podmínky

```
if x > 0:
   x += 10
else:
   x -= 10
```

Vnořená podmínka:

```
if x > 0:
   x += 10
   if a > 100:       #Vnorena podminka
      a *= 10
   else:             #Parovano s nejblizsim if
      a/=10
else:
   x -= 10
```

27. Kombinovaná podmínka

Dvě varianty mnohdy nestačí, výběr z více **vylučujících** se variant.

Každá, s výjimkou poslední, testována.

Umožňuje realizovat složitější větvení programu: více než 2 varianty.

Použit příkaz `elif`, zkrácení `else if` (netypické).

```
if booleovsky_vyraz1:       #Otestuj 1. podminku
   prikaz1
   prikaz2
   ...
elif booleovsky_vyraz2:     #Pokud nesplnena, otestuj 2. podminku
   prikaz3
   prikaz4
   ...
elif booleovsky_vyraz3:     #Pokud nesplnena, otestuj 3. podminku
   prikaz5
   prikaz6
   ...
else:                       #Default, pokud nesplnena zadna z predchozich
   prikaz7
   prikaz8
   ...
```

Pozor na pořadí podmínek: rozšiřující, ne zužující.

28. Ternární operátor

Operátor má tři argumenty: 2 hodnoty a výraz.

Umožňuje zapsat úplnou podmínku stručnějším, avšak méně přehledným, způsobem.

```
hodnota1 if condition else hodnota2;
```

Je -li výraz vyhodnocen jako pravdivý, je vrácena `hodnota1`, jinak `hodnota2`.

Podmínku

```
if a<b:
   c = a + 10
else:
   c = a - 10;
```

lze zapsat jako

```
c = a+10 if a<b else a-10
c = (a+10) if a<b else (a-10)
```

Závorky nepovinné, slouží pro zdůraznění podmínky.

29. Příkaz match-case

Přepínač, větvení programu do více větví (vylučující se podmínky).

Počet větví není omezen, v každé vykonán nějaký příkaz.

Přehlednější varianta `if-else`.

```
match vyraz:                      #Vyraz
        case const1:              #Navesti 1
            prikaz
        case const2:              #Navesti 2
            prikaz
        case _:                   #Navesti 3, vychozi, nepovinne
            prikaz
```

Vyhodnocením výrazu celočíselná hodnota.

Návěští: celočíselná nebo znaková, unikátní hodnota.

Dle hodnoty návěští provedeny příkazy v bloku.

Pokud nenalezeno odpovídající návěští, vykonán kód nacházející v návěští `default`.

```
match x:
        case 'a':               #if
            return 0
        case 'b':               #elif
            return 1
        case _:                 #else
            return 2
```

30. Příklad: kvadratická rovnice

```
import math
from math import sqrt

a = int(input("a: "))   #Standardni vstup
b = int(input("b: "))
c = int(input("c: "))

D = b * b - 4. * a * c  #Diskriminant

if D < 0:    #0 reseni
    print ("Nema reseni v R.")

elif D == 0: #1 reseni
    x = (-b + sqrt(D))/(2. * a)
    print ("Dvojnasobny koren: ", x)

else:        #2 reseni
    x1 = (-b - sqrt(D))/(2. * a)
    x2 = (-b + sqrt(D))/(2. * a)
    print ("Dva koreny: ", x1, " a", x2)
```