

# Prednášky z Matematiky (4) — Logiky pre informatikov

Ján Klúka, Jozef Šiška

Katedra aplikovanej informatiky  
FMFI UK Bratislava

Letný semester 2017/2018

## 7. prednáška

SAT solver a algoritmus DPLL  
Syntax relačnej logiky prvého rádu

9. apríla 2018

# Obsah 7. prednášky

## 2 Výroková logika

- Logické pojmy a slovné úlohy

- Problém výrokovologickej splniteľnosti (SAT)

  - Naivný backtracking

  - Optimalizácia backtrackingu

  - DPLL

## 3 Logika prvého rádu

- Syntax relačnej logiky prvého rádu

- Formalizácia v logike prvého rádu

  - Jednoduchá formalizácia

  - Základné idiómy

  - Nutné a postačujúce podmienky

  - Idiómy s rovnosťou

## 2.11

# Logické pojmy a slovné úlohy

# Logické pojmy a slovné úlohy

## Cvičenie 2.98 (3. úloha zo 6. sady úloh)

Prechádzate sa v labyrinte a zrazu sa ocitnete na križovatke, ...

Každú cestu stráži strážnik a každý z nich vám povie niečo o cestách:

**Strážnik zlatej cesty:** „Táto cesta vedie priamo do stredu labyrintu. Navyše, ak vás kamienky dovedú do stredu, tak vás do stredu dovedie aj mramor.“

...

Viete, že všetci strážnici stále klamú.

- i Môžete si byť istí, že niektoré z ciest vedú do stredu labyrintu?  
Ak áno, ktorú cestu si vyberiete?
- ii Viete o niektorých cestách s určitosťou povedať,  
že do stredu labyrintu nevedú? Ak áno, ktoré to sú?
- iii Je o niektorých cestách nemožné povedať,  
či do stredu labyrintu vedú alebo nevedú? Ak áno, o ktorých?

## 2.12

# Problém výrokovologickej splniteľnosti (SAT)

# Problém SAT

- *Problémom výrokovologickej splniteľnosti (SAT)* je problém určenia toho, či je daná množina výrokových formúl splniteľná
- Zvyčajne sa redukuje na problém splniteľnosti množiny klauzúl (teda formuly v CNF)
- *SAT solver* je program, ktorý rieši problém SAT

## Príklad 2.99

Je množina klauzúl  $S$  splniteľná?

$$S = \{(a \vee b), (a \vee \neg b), (\neg a \vee b), (\neg a \vee \neg b \vee \neg c), (\neg a \vee c)\}$$

# Tabuľková metóda

- Tabuľkovou metódou skúmame všetky ohodnotenia výrokových premenných
- Preskúmanie ohodnotení trvá  $O(s2^N)$  krokov, kde  $N$  je počet premenných a  $s$  je súčet veľkostí klauzúl
  - ▶  $2^N$  ohodnotení, pre každé treba zistiť, či sú všetky klauzuly splnené
- Celú tabuľku si pamätáme (píšeme na papier)
- Tabuľka zaberá priestor  $O(k2^N)$ , kde  $k$  je počet klauzúl
- Tabuľka slúži aj ako dôkaz nespľniteľnosti



## 2.12.1

# Naivný backtracking

# Naivný backtracking v Pythone

```
#!/usr/bin/env python3
class Sat(object):
    def __init__(self, n, clauses):
        self.n, self.clauses, self.solution = n, clauses, None
    def checkClause(self, e, c):
        return any( ( e[abs(lit)] if lit > 0 else not e[abs(lit)] )
                    for lit in c )
    def check(self, e):
        return all( self.checkClause(e, cl) for cl in self.clauses )
    def solve(self, i, e):
        if i >= self.n:
            if self.check(e):
                self.solution = e
                return True
            return False
        for v in [True, False]:
            e[i] = v
            if self.solve(i+1, e):
                return True
        return False
Sat(20, [[]]).solve(0, {})
```

Čas:  $O(s2^N)$ , priestor:  $O(s+N)$ ;

$N$  — počet premenných,

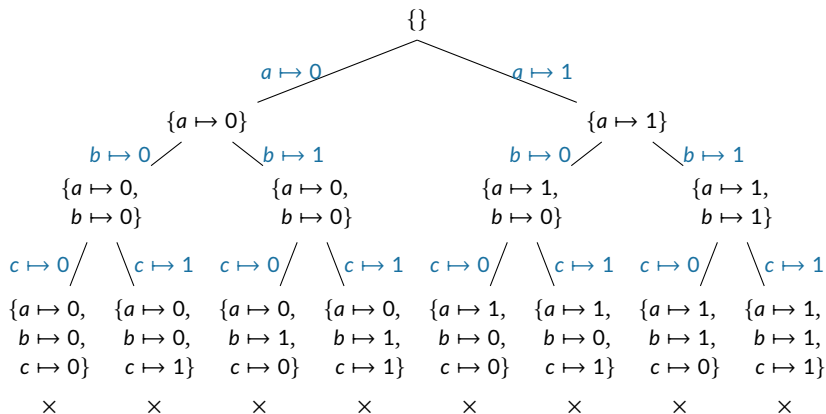
$s$  — súčet veľkostí klauzúl

# Strom prehľadávania ohodnotení

$$S = \{(a \vee b), (a \vee \neg b), (\neg a \vee b), (\neg a \vee \neg b \vee \neg c), (\neg a \vee c)\}$$

$$\times: v \models S$$

$$f := 0, t := 1$$



# Naivné C++

```
#include <iostream>
int N = 10; bool e[50];
bool check() {
    return false; // kontrola splnenia všetkých klauzúl
}
bool solve1(int i) {
    if (i >= N) {
        if (check())
            return true;
        return false;
    }
    e[i] = false;
    if (solve1(i+1)) return true;
    e[i] = true;
    return solve1(i+1);
}
int main(int argc, char *argv[]) {
    N=atoi(argv[1]);
    std::cout << "N=" << N << std::endl;
    solve1(0);
    return 0;
}
```

# Trochu lepšie C++

```
#include <iostream>
int N = 10;
bool check2(unsigned long long e) {
    return false; // kontrola splnenia všetkých klauzúl
}
bool solve2() {
    unsigned long long e, m = 1ULL << N;
    for (e=0; e < m ; ++e) {
        if (check2(e))
            return true;
    }
    return false;
}
int main(int argc, char *argv[]) {
    N=atoi(argv[1]);
    std::cout << "N=" << N << std::endl;
    solve2();
    return 0;
}
```

# Čas

Čas prehľadávania stromu ohodnotení v závislosti od počtu literálov

Riešenie	10	20	30	35
python	0m0.028s	0m0.877s	14m49.221s	> 7h
cpp1	0m0.001s	0m0.012s	0m11.085s	5m07.995s
cpp2	0m0.001s	0m0.008s	0m03.441s	1m50.086s

## 2.12.2

# Optimalizácia backtrackingu

# Priebežné vyhodnocovanie klauzúl

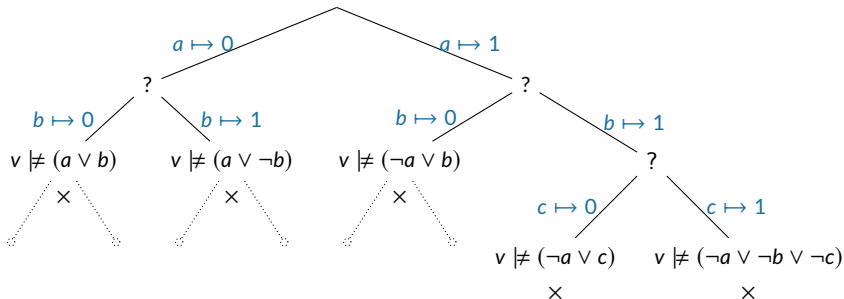
- Každý uzol prehľadávaného stromu ohodnotení je **čiasťočné ohodnotenie**
- Ohodnotenie v uzle je **rozšírením** ohodnotenia v rodičovi
- Niektoré klauzuly sa dajú vyhodnotiť aj v čiastočnom ohodnotení
  - ▶ Napríklad v čiastočnom ohodnotení  $v = \{a \mapsto 0, b \mapsto 1\}$  vieme určiť splnenie  $(a \vee b)$ ,  $(a \vee \neg b)$ ,  $(\neg a \vee b)$  z našej  $S$
- Ak je niektorá nesplnená, môžeme „backtracknúť“ — zastaviť prehľadávanie vetvy a vrátiť sa o úroveň vyššie



# Prehľadávanie s priebežným vyhodnocovaním

$$S = \{(a \vee b), (a \vee \neg b), (\neg a \vee b), (\neg a \vee \neg b \vee \neg c), (\neg a \vee c)\}$$

$$\times: v \models S$$



# Zjednodušenie množiny klauzúl podľa literálu

Nech  $v$  je čiastočné ohodnotenie, v ktorom  $v(a) = 1$ .

Čo vieme o splnení klauzúl z  $S$  každým rozšírením  $v'$  ohodnotenia  $v$ ?

- $v'$  určite splní každú klauzulu obsahujúcu literál  $a$ 
  - ▶  $\{a \mapsto 1, \dots\} \models (a \vee b)$
  - ▶  $\{a \mapsto 1, \dots\} \models (a \vee \neg b)$

Tieto klauzuly sú pre zistenie splniteľnosti vo všetkých  $v'$  **nepodstatné**, môžeme ich vynechať

- $v'$  splní klauzulu  $(\ell_1 \vee \dots \vee \neg a \vee \dots \vee \ell_n)$  obsahujúcu  $\neg a$   
vtt  $v'$  splní **zjednodušenú** klauzulu  $(\ell_1 \vee \dots \vee \dots \vee \ell_n)$ 
  - ▶  $\{a \mapsto 1, \dots\} \models (\neg a \vee \neg b \vee \neg c)$  vtt  $\{a \mapsto 1, \dots\} \models (\neg b \vee \neg c)$
  - ▶ Mimochoďom,  $(\neg b \vee \neg c)$  je rezolventa  $a$  a  $(\neg a \vee \neg b \vee \neg c)$

**Stačia** nám zjednodušené klauzuly

# Zjednodušenie množiny klauzúl podľa literálu

Množinu klauzúl

$$S = \{ (a \vee b), (a \vee \neg b), (\neg a \vee b), (\neg a \vee \neg b \vee \neg c), (\neg a \vee c) \}$$

teda môžeme **zjednodušiť podľa  $a$**  na

$$S|_a = \{ b, (\neg b \vee \neg c), c \}.$$

Analogicky môžeme  $S$  zjednodušiť podľa  $\neg a$  na

$$S|_{\neg a} = \{ b, \neg b \}.$$

# Zjednodušenie množiny klauzúl podľa literálu

## Definícia 2.100

Nech  $p$  je výroková premenná.

**Komplementom literálu  $p$**  je  $\neg p$ . **Komplementom literálu  $\neg p$**  je  $p$ .

Komplement literálu  $\ell$  označujeme  $\bar{\ell}$ .

## Definícia 2.101

Nech  $\ell$  je literál a  $S$  je množina klauzúl. Potom definujeme

$$S|_{\ell} = \{ (\ell_1 \vee \cdots \vee \ell_n) \mid (\ell_1 \vee \cdots \vee \bar{\ell} \vee \cdots \vee \ell_n) \in S \} \cup \\ \{ C \mid C \in S, \text{ v } C \text{ sa nevyskytuje } \ell \text{ ani } \bar{\ell} \}.$$

## Tvrdenie 2.102

Nech  $\ell$  je literál a  $S$  je množina klauzúl.

Potom  $S \cup \{ \ell \}$  je splniteľná vtt  $S|_{\ell}$  je splniteľná.

# Propagácia jednotkových klauzúl

- Zjednodušením množiny klauzúl sa môže značne zmenšiť priestor spĺňajúcich ohodnotení
- Napríklad zjednodušením  $T = \{(a \vee \neg b), (a \vee b \vee c)\}$  podľa  $\neg a$  dostaneme  $T' := T|_{\neg a} = \{\neg b, (b \vee c)\}$
- $T'$  obsahuje **jednotkovú klauzulu** (unit clause alebo iba **unit**)  $\neg b$
- Preto  $T'$  spĺňajú iba ohodnotenia  $v$ , v ktorých  $v(b) = 0$
- Pre také ohodnotenia môžeme  $T'$  ďalej zjednodušiť podľa  $\neg b$ :  
 $T'' := T'|_{\neg b} = \{c\}$
- $T''$  môžu splniť iba ohodnotenia  $v$ , v ktorých  $v(c) = 1$
- Pre také ohodnotenia môžeme  $T''$  ďalej zjednodušiť podľa  $c$ :  
 $T''' := T''|_c = \{\}$
- $T'''$  je prázdna, teda je splniteľná

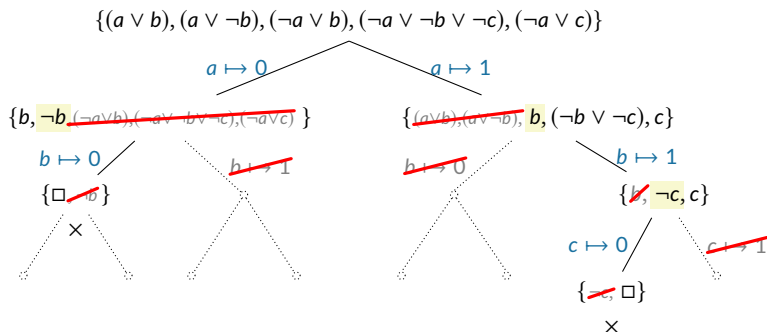
**Propagácia jednotkových klauzúl (unit propagation)** je proces opakovaného rozširovania ohodnotení podľa jednotkových klauzúl a zjednodušovania

# Propagácia jednotkových klauzúl

## Dôsledok 2.103

*Nech  $\ell$  je literál a  $S$  je množina klauzúl obsahujúca jednotkovú klauzulu  $\ell$  ( $\ell \in S$ ). Potom  $S$  je splniteľná vtt  $S|_{\ell}$  je splniteľná.*

# Prehľadávanie so zjednodušovaním klauzúl a unit propagation



# Eliminácia nezmiešaných literálov

- Všimnime si literál  $u$  v množine klauzúl:

$$T = \{(\neg a \vee \neg b \vee c), (\neg a \vee u), (\neg b \vee u), a, b, \neg c\}$$

- Literál  $u$  je **nezmiešaný** (angl. *pure*) v  $T$ :  
 $u$  sa vyskytuje v  $T$ , ale jeho komplement  $\neg u$  sa tam nevyskytuje
- Vynechajme z  $T$  všetky klauzuly obsahujúce  $u$ :

$$T' := T|_u = \{(\neg a \vee \neg b \vee c), a, b, \neg c\}$$

- ▶ Ak nájdeme ohodnotenie  $v \models T'$ ,  
tak  $v_0 := v(u \mapsto 0)$  aj  $v_1 := v(u \mapsto 1)$  sú modelmi  $T'$   
a  $v_1$  je navyše modelom  $T$ , teda  $T$  je splniteľná
- ▶ Ak je  $T'$  nespľniteľná,  
tak je nespľniteľná každá jej nadmnožina, teda aj  $T$

Takže: Z hľadiska splniteľnosti sú klauzuly obsahujúce  $u$  nepodstatné,  
stačí uvažovať  $T|_u$

Analogická úvaha sa dá aplikovať aj na  $\neg u$  a jeho komplement  $u$



# Eliminácia nezmiešaných literálov

## Definícia 2.104

Nech  $\ell$  je literál a  $S$  je množina klauzúl.

Literál  $\ell$  je **nezmiešaný** (*pure*) v  $S$  vtt  $\ell$  sa vyskytuje v niektorej klauzule z  $S$ , ale jeho komplement  $\bar{\ell}$  sa nevyskytuje v žiadnej klauzule z  $S$ .

## Tvrdenie 2.105

Nech  $\ell$  je literál a  $S$  je množina klauzúl.

Ak  $\ell$  je nezmiešaný v  $S$ , tak  $S$  je splniteľná vtt  $S|_{\ell}$  je splniteľná.

## 2.12.3

### DPLL

## Algoritmus 2.106 (Davis and Putnam [1960], Davis et al. [1962])

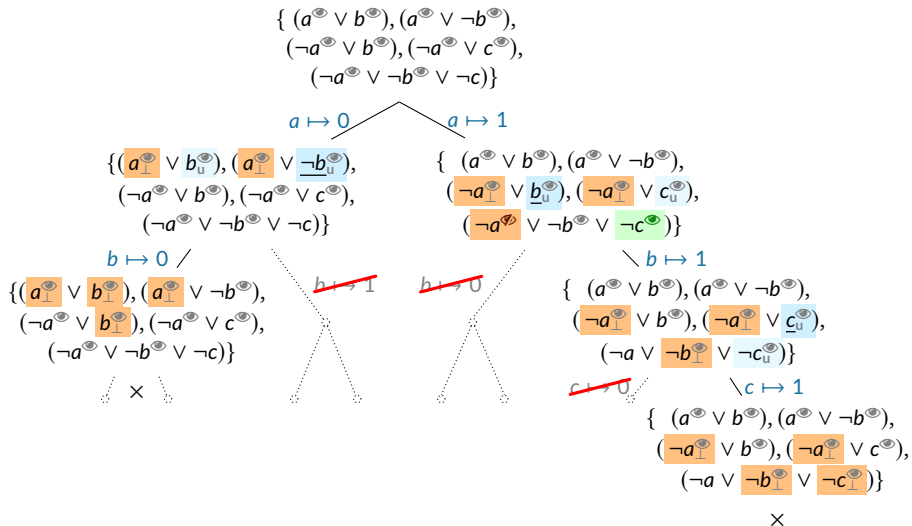
```
1: function DPLL( $\Phi, e$ )
2:   if  $\Phi$  obsahuje prázdnu klauzulu then
3:     return False
4:   end if
5:   if  $e$  ohodnocuje všetky premenné then
6:     return True
7:   end if
8:   while existuje jednotková (unit) klauzula  $\ell$  vo  $\Phi$  do
9:      $\Phi, e \leftarrow \text{unit-propagate}(\ell, \Phi, e)$ 
10:  end while
11:  while existuje nezmiešaný (pure) literál  $\ell$  vo  $\Phi$  do
12:     $\Phi, e \leftarrow \text{pure-literal-assign}(\ell, \Phi, e)$ 
13:  end while
14:   $x \leftarrow \text{choose-branch-literal}(\Phi, e)$ 
15:  return DPLL( $\Phi|_x, e(x \mapsto T)$ ) or DPLL( $\Phi|_{\neg x}, e(x \mapsto F)$ )
16: end function
```

# Technika sledovaných literálov (watched literals)

Aby sme nemuseli zjednodušovať množinu klauzúl:

- Pre každú klauzulu máme 2 sledované literály.
- Sledovaný literál vždy musí byť *nenastavený* alebo *true*.
- Ak nejaký literál nastavíme na *true*: nič nemusíme robiť.
- Ak nejaký literál nastavíme na *false*: musíme nájsť iný. Ak iný nie je, práve sme vyrobili jednotkovú klauzulu (všetky literály okrem toho druhého sledovaného sú *false*).
- Ak backtrackujeme: nič nemusíme robiť (možno sa niektoré sledované literály stali *nenastavenými*).

# Prehľadávanie s unit propagation a sledovaním



# 3.1

## Syntax relačnej logiky prvého rádu

# Štruktúra jednoduchých viet

- Výroková logika **veľmi** zjednodušuje prirodzený jazyk:
  - ▶ skúma iba štruktúru tvrdení tvorenú spojkami,
  - ▶ atomické výroky *nemajú štruktúru*
- Niekedy to neprekáža — konštatovanie globálneho stavu:
  - ▶ Prší.
  - ▶ Cesta je mokrá.
  - ▶ Je pondelok.

# O čom hovoria atomické výroky?

- Atomické výroky často hovoria o **vlastnostiach objektov**

- ▶ Jerry je myš domová.
- ▶ Hlohovský je minister.
- ▶ Logika je ľahká.

alebo **vzťahoch** objektov

- ▶ Dorothy je staršia ako George.
- ▶ Komorník je bohatší ako grófka Agáta.
- ▶ Hlohovský prijal úplatok 250 tisíc eur od Veseliča v roku 2013.

- Výroková logika vynucuje **samostatné výrokové premenné** pre rôzne **kombinácie** objektov, vlastností a vzťahov — neintuitívne, nepraktické
- Existujú ale iné logiky ako výroková
- Logika prvého rádu** rozoznáva štruktúru atomických výrokov



# Štruktúra jednoduchých viet

- Jednoduché vety v prirodzených jazykoch sa delia na  
*podmetovú* a *prísudkovú* časť  
Hlohovský prijal úplatok 250 tisíc eur od Veseliča v roku 2013  
Prísudková časť sa ďalej delí na:  
*prísudok*   *predmet*                      *predmet*           *prísl. urč. času*  
prijal           úplatok 250 tisíc eur   od Veseliča   v roku 2013
- Logika prvého rádu nerozoberá štruktúru atomických výrokov až tak podrobne

# Predikátové symboly a ich argumenty

- **Atomické formuly** (jednoduché výroky) v logike prvého rádu:  
 $\text{predikátový\_symbol}(\text{argument}_1, \text{argument}_2, \dots, \text{argument}_n)$   
**Predikátový symbol** zodpovedá prísudku  
 alebo celej prísudkovej časti:  
 (je) minister, (je) starší (ako), prijal, <, ...  
**Jeho argumenty** zodpovedajú podmetu, predmetu, ...  
 Úloha argumentu je daná *pozíciou*  
 (ako v programovacích jazykoch).
- Predikátový symbol má jednoznačne určenú **aritu** —  
 očakávaný počet argumentov
- **Vždy** musí mať práve toľko argumentov, aká je jeho arita

## Dohoda 3.1

Aritu budeme niekedy písať ako horný index symbolu  
 (minister<sup>1</sup>, starší<sup>2</sup>, prijal<sup>4</sup>, <<sup>2</sup>).

# Význam predikátových symbolov

**Unárny** predikátový symbol (teda s aritou 1)

zvyčajne predstavuje **vlastnosť**

$\text{minister}^1(\text{arg}_1)$   $\text{arg}_1$  je minister

$\text{myš\_domová}^1(\text{arg}_1)$   $\text{arg}_1$  je myš domová

$\text{ľahká}^1(\text{arg}_1)$   $\text{arg}_1$  je ľahká

**Binárny, ternárny, ...** predikátový symbol (s aritou 2, 3, ...)

predstavuje **vzťah**

$\text{starší}^2(\text{arg}_1, \text{arg}_2)$   $\text{arg}_1$  je starší ako  $\text{arg}_2$

$\text{medzi}^3(\text{arg}_1, \text{arg}_2, \text{arg}_3)$   $\text{arg}_1$  sa nachádza

medzi  $\text{arg}_2$  a  $\text{arg}_3$

$\text{prijal}^4(\text{arg}_1, \text{arg}_2, \text{arg}_3, \text{arg}_4)$   $\text{arg}_1$  prijal  $\text{arg}_2$   
od  $\text{arg}_3$  v čase  $\text{arg}_4$

# Výber predikátových symbolov

- Predikátový symbol predstavuje vlastnosť alebo vzťah, ktorého **pravdivosť** pre dané argumenty sa dá určiť **jednoznačne**
  - ▶ Napríklad pravdivosť vzťahu *byť vyšší ako* sa **dá** určiť jednoznačne.
  - ▶ Naopak pravdivosť vlastnosti *byť vysoký* sa **nedá** určiť jednoznačne.
    - Takýmito neostrými vlastnosťami sa zaoberajú *fuzzy* logiky.
- Často zanedbávame detaily —  
pomocné slovesá, predložky, skloňovanie, rod, ...:  
 $\text{starší}^2(arg_1, arg_2)$  —  $arg_1$  je starší/staršia/staršie ako  $arg_2$

# Konkrétne argumenty predikátov — konštanty

- V prirodzenom jazyku *vlastné mená* označujú konkrétne, známe objekty alebo ľudí.
- V logike prvého rádu konkrétne, pevne dané objekty alebo hodnoty označujeme **symbolmi konštant**.
  - ▶ Dorothy, Hlohovský, Jerry, 0, 1, 2, ..., rok2013
- Môžu byť **argumentmi predikátových symbolov** v atomických formulách
  - ▶ `minister(Hlohovský)`, `starší(Dorothy, George)`,  
`prijal(Hlohovský, úplatok250000€, Veselič, rok2013)`
- Samé o sebe **nie sú formulami**, nemajú pravdivostnú hodnotu.
- Dva symboly konštant môžu označovať ten istý objekt:
  - ▶ `stvrty_prezident_SR` a `Andrej_Kiska`
- **Rovnostné atómy** — špeciálny druh atomických formúl:  
`stvrty_prezident_SR`  $\doteq$  `Andrej_Kiska`

# Výrokové spojky a atomické formuly

- V logike prvého rádu môžeme atomické formuly **spájať výrokovými spojkami** rovnako ako vo výrokovej logike:
  - ▶  $((\text{matka}(\text{Dorothy}) \wedge \text{syn}(\text{George})) \rightarrow \text{starší}(\text{Dorothy}, \text{George}))$
  - ▶  $(\text{zomrel}(\text{Stephen\_Hawking}) \rightarrow \neg(\text{najznámejší\_žijúci\_fyzik} \doteq \text{Stephen\_Hawking}))$
  - ▶  $(\text{prijal}(\text{Hlohovský}, \text{úplatok}250000\text{€}, \text{Veselič}, \text{rok}2013) \rightarrow \neg\text{dôveryhodný}(\text{Hlohovský}))$
- Máme ale aj zaujímavejšie možnosti...

# Premenné a otvorené atomické formuly

- Atomické formuly nemusia vyjadrovať iba vlastnosti *konkrétnych* objektov označených konštantami
- Argumentami predikátových symbolov môžu byť aj **symboly individuových premenných** (skrátene *premenné*)

## Dohoda 3.2

Ako premenné budeme zvyčajne používať malé písmená z konca abecedy  $u, v, w, x, y, z$  s prípadnými dolnými indexmi.

- Zastupujú objekty zo sveta, o ktorých chceme vysloviť nejakú vlastnosť alebo vzťah, ale nemôžeme ich označiť konštantami
- Atomické formuly s premennými nazývame **otvorené**
  - ▶  $\text{starší}(x, \text{Dorothy}), \text{minister}(z_5)$Nepredstavujú plnohodnotné výroky, ale *výrokové formy*
- Premenné a formuly s nimi nadobúdajú význam pomocou *kvantifikátorov*

# Všeobecný kvantifikátor

- **Všeobecný kvantifikátor**  $\forall$  predstavuje zámená „každý“, „všetci“, „pre všetky“, ...
- *Viaže* premennú, ktorá za ním nasleduje
- Vyjadruje, že vlastnosť, ktorú nasledujúca formula opisuje pre viazanú premennú, majú *všetky objekty*
  - ▶  $\forall x \text{starší}(x, \text{Dorothy})$  – každý je starší ako Dorothy
- Kvantifikovaná formula nemusí byť atomická:
  - ▶  $\forall x (\text{starší}(x, \text{Dorothy}) \vee \neg \text{starší}(\text{George}, x))$



# Existenčný kvantifikátor

- **Existenčný kvantifikátor** *exists* predstavuje frázy „niekto“, „niečo“, „aspoň jedno“, „existuje/je ... také, že ...“, ...
- Vyjadruje, že vlastnosť, ktorú nasledujúca formula opisuje pre viazanú premennú, má *aspoň jeden objekt*
  - ▶  $\exists x \text{ starší}(x, \text{George})$  — niekto je starší ako George
- Kvantifikovaná formula nemusí byť atomická:
  - ▶  $\exists x (\text{starší}(x, \text{George}) \wedge \text{starší}(\text{Virginia}, x))$
- Kvantifikovaná formula môže obsahovať ďalšie kvantifikátory:
  - ▶  $\exists x \forall y \text{ starší}(x, y)$
  - ▶  $\forall x (\exists y \exists u \exists z (\text{prijal}(x, u, y, z) \wedge \text{úplatok}(u)) \rightarrow \neg \text{dôveryhodný}(x))$

# Symboły jazyka relačnej logiky prvého rádu

## Definícia 3.3

*Symbolmi jazyka  $\mathcal{L}$  relačnej logiky prvého rádu sú:*

*symbols (individuových) premenných* z nejakej nekonečnej spočítateľnej množiny  $\mathcal{V}_{\mathcal{L}}$  (označujeme ich  $x, y, \dots$ );

*mimologické symbols:*

*symbols konštánt* z nejakej spočítateľnej množiny  $C_{\mathcal{L}}$  (označované  $a, b, \dots$ );

*predikátové symbols* z nejakej spočít. množiny  $\mathcal{P}_{\mathcal{L}}$  (ozn.  $P, R, \dots$ );

*logické symbols:*

*logické spojky:* unárna  $\neg$ , binárne  $\wedge, \vee, \rightarrow$ ,

*symbol rovnosti*  $\doteq$ ,

*kvantifikátory:* *existenčný kvantifikátor*  $\exists$  a *všeobecný kvantifikátor*  $\forall$ ;

*pomocné symbols*  $(, )$  a  $,$  (ľavá, pravá zátvorka a čiarka).

Množiny  $\mathcal{V}_{\mathcal{L}}, C_{\mathcal{L}}, \mathcal{P}_{\mathcal{L}}$  sú vzájomne disjunktné.

Logické a pomocné symbols sa nevyskytujú v symbols z  $\mathcal{V}_{\mathcal{L}}, C_{\mathcal{L}}, \mathcal{P}_{\mathcal{L}}$ .

Každému symbolu  $P \in \mathcal{P}_{\mathcal{L}}$  je priradená *arita*  $\text{ar}(P) \in \mathbb{N}^+$ .

# Symbody jazyka logiky s rovnosťou

## Poznámka 3.4

Symbody (konštant, funkčné, predikátové) môžu byť nealfabetické (1, <, +), či tvorené viacerými znakmi (Virginia, dcéra).

# Atomické formuly relačnej logiky prvého rádu

## Definícia 3.5 (Term)

Nech  $\mathcal{L}$  je jazyk relačnej logiky prvého rádu.

Symbols premenných z  $\mathcal{V}_{\mathcal{L}}$  a konštánt z  $C_{\mathcal{L}}$  súhrnne nazývame **termy**.

## Definícia 3.6 (Atomické formuly)

Nech  $\mathcal{L}$  je jazyk relačnej logiky prvého rádu.

**Rovnostný atóm** jazyka  $\mathcal{L}$  je každá postupnosť symbolov  $t_1 \doteq t_2$ ,  
kde  $t_1$  a  $t_2$  sú termy.

**Predikátový atóm** jazyka  $\mathcal{L}$  je každá postupnosť symbolov  $P(t_1, \dots, t_n)$ ,  
kde  $P$  je predikátový symbol s aritou  $n$  a  $t_1, \dots, t_n$  sú termy.

**Atomickými formulami** (skrátene **atómami**) jazyka  $\mathcal{L}$   
súhrnne nazývame všetky rovnostné a predikátové atómy jazyka  $\mathcal{L}$ .  
Množinu všetkých atómov jazyka  $\mathcal{L}$  označujeme  $\mathcal{A}_{\mathcal{L}}$ .

# Formuly jazyka relačnej logiky prvého rádu

## Definícia 3.7

Množina  $\mathcal{E}_{\mathcal{L}}$  všetkých **formúl** jazyka relačnej logiky prvého rádu  $\mathcal{L}$  je *najmenšia* množina postupností symbolov jazyka  $\mathcal{L}$ , pre ktorú platí:

- Všetky atomické formuly z  $\mathcal{A}_{\mathcal{L}}$  sú formulami z  $\mathcal{E}_{\mathcal{L}}$  (teda  $\mathcal{A}_{\mathcal{L}} \subseteq \mathcal{E}_{\mathcal{L}}$ ).
- Ak  $A$  je formula z  $\mathcal{E}_{\mathcal{L}}$ , tak aj  $\neg A$  je formula z  $\mathcal{E}_{\mathcal{L}}$  (**negácia**  $A$ ).
- Ak  $A$  a  $B$  sú formuly z  $\mathcal{E}_{\mathcal{L}}$ , tak aj  $(A \wedge B)$ ,  $(A \vee B)$ ,  $(A \rightarrow B)$  sú formuly z  $\mathcal{E}_{\mathcal{L}}$  (**konjunkcia**, **disjunkcia**, **implikácia**  $A$  a  $B$ ).
- Ak  $x$  je individuová premenná a  $A$  je formula z  $\mathcal{E}_{\mathcal{L}}$ , tak aj  $\exists x A$  a  $\forall x A$  sú formuly z  $\mathcal{E}_{\mathcal{L}}$  (**existenčná** a **všeobecná kvantifikácia** formuly  $A$  vzhľadom na  $x$ ).

## Dohoda 3.8

Formuly označujeme písmenami  $A, B, C, \dots$  s prípadnými indexmi.

## 3.2

## Formalizácia v logike prvého rádu

## 3.2.1

# Jednoduchá formalizácia

# Jednoduchá formalizácia

## Príklad 3.9 (podľa Genesereth and Kao [2013])

Sformalizujme v jazyku logiky prvého rádu túto situáciu:

V byte bývajú 4 spolubývajúce: Aďa, Biba, Ciri a Dada.

Niektoré sa kamarátia a niektoré sa nemajú rady,  
ale máme o tom iba tieto nepriame informácie:

- 1 Biba má rada Ciri alebo Dadu.
- 2 Aďa má rada všetkých, ktorých má rada Biba.
- 3 Ciri má rada každého, kto má rád ju.
- 4 Biba má rada niekoho, kto ju má rád.
- 5 Žiadna nemá rada seba samú.
- 6 Každá má rada niekoho.
- 7 Niekoho majú rady všetky.



## 3.2.2

# Základné idiómy

# Základné idiómy: Obmedzená kvantifikácia

Niektoré slovné obraty a ich prvorádové formalizácie sú veľmi bežné, ale pre začiatočníka nie úplne priamočiare:

**Obmedzená kvantifikácia** je všeobecné alebo existenčné tvrdenie, ktoré sa vzťahuje iba na objekty s nejakou vlastnosťou:

- „Každý, kto má vlastnosť  $P$ , má vlastnosť  $Q$ .“ / „Každý  $P$  je  $Q$ .“:  
▶  $\forall x(P(x) \rightarrow Q(x))$
- „Nieкто, kto má vlastnosť  $P$ , má vlastnosť  $Q$ .“ / „Niektorý  $P$  je  $Q$ .“  
▶  $\exists x(P(x) \wedge Q(x))$

# Základné idiómy: Neexistencia

**Neexistencia** je negované existenčné tvrdenie,  
v slovenčine sa často vyjadruje *dvojitým záporom*  
[negatívne zámeno (nikto/nič) a negatívne tvrdenie]:

**Jednoduchá vlastnosť** „Nikto nie je dokonalý“:

- S dôrazom na zámeno:  $\neg \exists x \text{ dokonalý}(x)$
- S dôrazom na negatívne tvrdenie:  $\forall x \neg \text{dokonalý}(x)$

**Viacero vlastností** „Žiaden/nijaký vegán nie je obézny“:

- S dôrazom na zámeno:
  - ▶  $\neg \exists x (\text{vegán}(x) \wedge \text{obézny}(x))$
- S dôrazom na negatívne tvrdenie:
  - ▶  $\forall x \neg (\text{vegán}(x) \wedge \text{obézny}(x))$
  - ▶  $\forall x (\neg \text{vegán}(x) \vee \neg \text{obézny}(x))$
  - ▶  $\forall x (\text{vegán}(x) \rightarrow \neg \text{obézny}(x))$

# Základné idiómy: Zamlčaná a zdanlivá existencia

## Zamlčaná existencia

- každý vegán si kúpil tekvicu:  
▶  $\forall x(\text{vegán}(x) \rightarrow \exists y(\text{kúpil}(x, y) \wedge \text{tekvica}(y)))$
- žiadny vegán si nekúpil syr:  
▶  $\neg \exists x(\text{vegán}(x) \wedge \exists y(\text{kúpil}(x, y) \wedge \text{syr}(y)))$   
▶  $\forall x(\text{vegán}(x) \rightarrow \neg \exists y(\text{kúpil}(x, y) \wedge \text{syr}(y)))$   
▶  $\forall x(\text{vegán}(x) \rightarrow \forall y(\neg \text{kúpil}(x, y) \vee \neg \text{syr}(y)))$   
▶  $\forall x(\text{vegán}(x) \rightarrow \forall y(\text{kúpil}(x, y) \rightarrow \neg \text{syr}(y)))$

## Existencia v antecedente s odkazom v konzekvente

- ak je *niekto* vegán, tak *on* nie je obézny:  
▶  $\forall x(\text{vegán}(x) \rightarrow \neg \text{obézny}(x))$

### 3.2.3

## Nutné a postačujúce podmienky

# Nutné a postačujúce podmienky

- Často sa vyskytujú tvrdenia typu:
  - 1 Vegán je každý, kto si kúpil karfiol.
  - 2 Vegán je iba ten, kto si kúpil tekvicu.
- Hlavná veta („Vegán je ...“) vyjadruje nejakú **vlastnosť**
- Vedľajšia veta („kto si ...“) vyjadruje **podmienku**, ktorá súvisí s touto vlastnosťou
- Aký je rozdiel medzi týmito podmienkami?

# Postačujúca podmienka

Prvé tvrdenie „Vegán je každý, kto si kúpil karfiol.“

- Hovorí, že na to, aby niekto bol vegánom, *stačí*, aby platila podmienka, že si kúpil karfiol
- Kúpenie si karfiolu je teda **postačujúcou** podmienkou vegánstva
- Ekvivalentne:
  - „Pre každého platí, že je vegán, ak si kúpil karfiol.“
  - „Pre každého platí, že ak si kúpil karfiol, tak je vegán.“
- Formalizácia je teda  $\forall x(\exists y(\text{kúpil}(x, y) \wedge \text{karfiol}(y)) \rightarrow \text{vegán}(x))$

# Nutná podmienka

Druhé tvrdenie „Vegán je *iba* ten, kto si kúpil tekvicu.“

- Hovorí, že na to, aby niekto bol vegánom, *nevyhnutne* preňho platí podmienka, že si kúpil tekvicu (keby si ju nekúpil, nebol by vegánom)
- Kúpenie si tekvice je teda **nutnou** podmienkou vegánstva
- Ekvivalentne:
  - „Pre každého platí, že je vegán, *iba* ak si kúpil tekvicu.“
  - „Pre každého platí, že ak si *nekúpil* tekvicu, tak *nie* je vegán.“
  - „Pre každého platí, že ak je vegán, tak si kúpil tekvicu.“
- Formalizácia je teda  $\forall x(\text{vegán}(x) \rightarrow \exists y(\text{kúpil}(x, y) \wedge \text{syr}(y)))$



## 3.2.4

# Idiómy s rovnosťou

# Idiómy s rovnosťou: Enumerácia

## Vymenovanie objektov s vlastnosťou

- V byte č. 14 bývajú Aďa, Biba, Ciri, Dada.
  - ▶  $(\text{býva\_v}(\text{Aďa}, \text{byt14}) \wedge \dots \wedge \text{býva\_v}(\text{Dada}, \text{byt14}))$

Ekvivalentne:

Každá z Aďa, Biba, Ciri, Dada býva v byte č. 14.

- ▶  $\forall x((x \doteq \text{Aďa} \vee \dots \vee x \doteq \text{Dada}) \rightarrow \text{býva\_v}(x, \text{byt14}))$
- V byte č. 14 bývajú *iba* Aďa, Biba, Ciri, Dada.
 

Každý, kto býva v byte č. 14, je jedna z Aďa, Biba, Ciri, Dada.

  - ▶  $\forall x(\text{býva\_v}(x, \text{byt14}) \rightarrow (x \doteq \text{Aďa} \vee \dots \vee x \doteq \text{Dada}))$

# Idiómy s rovnosťou: Obmedzenia počtu

## Aspoň k:

- Jaro si kúpil aspoň dve tekvice.
- Existujú dve *navzájom rôzne* tekvice, ktoré si Jaro kúpil.
  - ▶  $\exists t_1 \exists t_2 (\neg t_1 \doteq t_2 \wedge \text{tekvica}(t_1) \wedge \text{tekvica}(t_2) \wedge \text{kúpil}(\text{Jaro}, t_1) \wedge \text{kúpil}(\text{Jaro}, t_2))$

## Najviac jeden:

- Anka si kúpila najviac jednu tekvicu.
- Ekvivalentne: Anka si nekúpila aspoň dve tekvice.
  - ▶  $\neg \exists t_1 \exists t_2 (\neg t_1 \doteq t_2 \wedge \text{tekvica}(t_1) \wedge \text{tekvica}(t_2) \wedge \text{kúpil}(\text{Anka}, t_1) \wedge \text{kúpil}(\text{Anka}, t_2))$
  - ▶  $\forall t_1 \forall t_2 \neg (\neg t_1 \doteq t_2 \wedge \text{tekvica}(t_1) \wedge \text{tekvica}(t_2) \wedge \text{kúpil}(\text{Anka}, t_1) \wedge \text{kúpil}(\text{Anka}, t_2))$
  - ▶  $\forall t_1 \forall t_2 (t_1 \doteq t_2 \vee \neg \text{tekvica}(t_1) \vee \neg \text{tekvica}(t_2) \vee \neg \text{kúpil}(\text{Anka}, t_1) \vee \neg \text{kúpil}(\text{Anka}, t_2))$
  - ▶  $\forall t_1 \forall t_2 ((\text{tekvica}(t_1) \wedge \text{tekvica}(t_2) \wedge \text{kúpil}(\text{Anka}, t_1) \wedge \text{kúpil}(\text{Anka}, t_2)) \rightarrow t_1 \doteq t_2)$
- Teda ekviv.: Všetky tekvice, ktoré si Anka kúpila, sú rovnaké.

# Literatúra

Martin Davis and Hillary Putnam. A computing procedure for quantification theory. *J. Assoc. Comput. Mach.*, 7:201–215, 1960.

Martin Davis, George Logemann, and Donald Loveland. A machine program for theorem-proving. *Communications of the ACM*, 5(7):394–397, 1962.

Michael Genesereth and Eric Kao. *Introduction to Logic*. Morgan & Claypool, 2013. ISBN 9781627052481.

Christos H. Papadimitriou. *Computational complexity*. Addison-Wesley, 1994. ISBN 978-0-201-53082-7.

Raymond M. Smullyan. *Logika prvého rádu*. Alfa, 1979. Z angl. orig. *First-Order Logic*, Berlin-Heidelberg: Springer-Verlag, 1968 preložil Svätoslav Mathé.

Vítězslav Švejdar. *Logika: neúplnosť, složitost, nutnost*. Academia, 2002. Prístupné aj na <http://www1.cuni.cz/~svejdar/book/LogikaSve2002.pdf>.