Prednášky z Matematiky (4) — Logiky pre informatikov

Ján Kľuka, Jozef Šiška

Katedra aplikovanej informatiky
FMFI UK Bratislava

Letný semester 2018/2019

8. prednáška

SAT solver a algoritmus DPLL
Syntax relačnej logiky prvého rádu

8. apríla 2019

Obsah 8. prednášky

2.11
Logické pojmy a slovné úlohy

Logické pojmy a slovné úlohy

- Kedy je potrebné zisťovať splniteľnosť teórie?
- Je pravda: Ak je X falzifikovateľná, tak $T \not\models X$?
- Čo znamená úplná otvorená vetva v table pre $\{\mathbf{T}\,A \mid A \in T\} \cup \{\mathbf{F}\,X\}$?
- Čo znamenajú nejaké, ale nie všetky uzavreté vetvy v table pre $\{\mathbf{T}\,A \mid A \in T\} \cup \{\mathbf{F}\,X\}$?
- Čo znamená uzavreté tablo pre $\{\mathbf{T}\,A \mid A \in T\} \cup \{\mathbf{T}\,X\}$?
- Čo znamená úplná otvorená vetva v table pre $\{\mathbf{T}\,A \mid A \in T\} \cup \{\mathbf{T}\,X\}$?
- Čo znamená úplná otvorená vetva v table pre $\{\mathbf{T}\,A \mid A \in T\} \cup \{\mathbf{T}\,X\}$ aj v table pre $\{\mathbf{T}\,A \mid A \in T\} \cup \{\mathbf{F}\,X\}$?

2.12 Problém výrokovologickej splniteľnosti (SAT)

Problém SAT

Definícia 2.111 (Problém SAT)

Problémom výrokovologickej splniteľnosti (*SAT*) je problém určenia toho, či je daná množina výrokových formúl splniteľná

- Zvyčajne sa redukuje na problém splniteľnosti *klauzálnej* teórie (teda formuly v CNF)
- *SAT solver* je program, ktorý rieši problém SAT

Príklad 2.112

Je množina klauzúl S splniteľná?

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

Tabuľková metóda

Tabuľková metóda:

- Skúma *všetky* ohodnotenia výrokových premenných
- Trvá $\mathrm{O}(s2^N)$ krokov,
  - N je počet premenných a s je súčet veľkostí klauzúl
  - 2^N ohodnotení, pre každé treba zistiť, či sú všetky klauzuly splnené
- Zaberá priestor $\mathrm{O}(k2^N)$
  - k je počet klauzúl
  - Pamätáme si (píšeme na papier) celú tabuľku
- Tabuľka slúži *aj* ako dôkaz prípadnej **ne**splniteľnosti

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: $O(s+N)$;

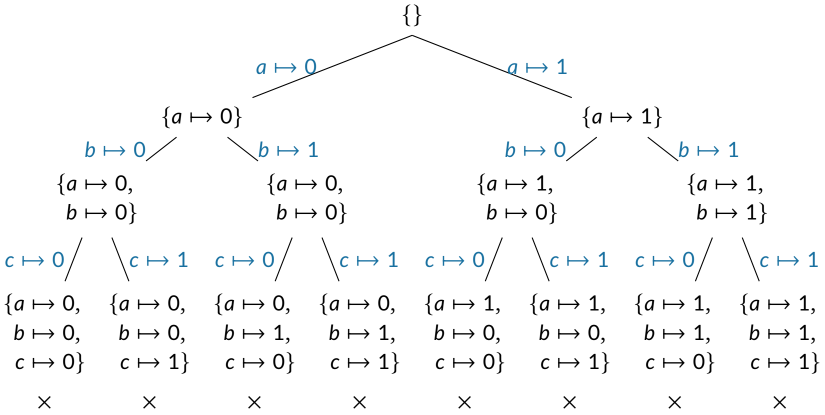
N — počet premenných,

s — súčet veľkostí klauzúl

Strom prehľadávania ohodnotení

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

$\times$ znamená $v \not\models S$ $f := 0, t := 1$



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

Strom ohodnotení:

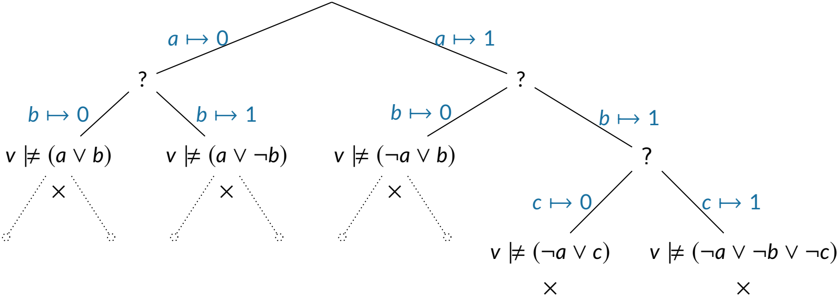
- List — ohodnotenie všetkých premenných
- Každý uzol — ***čiastočné ohodnotenie***
- Ohodnotenie v uzle je ***rozšírením*** ohodnotenia v rodičovi
- Niektoré klauzuly sa dajú vyhodnotiť aj v čiastočnom ohodnotení
  - V čiastočnom ohodnotení $v = \{a \mapsto 0, b \mapsto 1\}$ sa dá určiť splnenie $(a \lor b)$, $(a \lor \neg b)$, $(\neg a \lor b)$ z našej S
- Ak nájdeme nesplnenú, môžeme hneď „backtracknúť" — zastaviť prehľadávanie vetvy a vrátiť sa o úroveň vyššie

Prehľadávanie s priebežným vyhodnocovaním

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

$\times$ znamená $v \not\models S$

? znamená zatiaľ žiadna nesplnená klauzula



Zjednodušenie množiny klauzúl podľa literálu

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

Každé rozšírenie v' ohodnotenia v:

- Splní klauzuly obsahujúce literál a
  - $\{a \mapsto 1, \dots\} \models (a \lor b)$
  - $\{a \mapsto 1, \dots\} \models (a \lor \neg b)$
- Splní klauzulu $(\ell_1 \lor \cdots \lor \neg a \lor \cdots \lor \ell_n)$ obsahujúcu $\neg a$ **vtt** splní **zjednodušenú** klauzulu $(\ell_1 \lor \cdots \lor \cdots \lor \ell_n)$
  - $\{a \mapsto 1, \dots\} \models (\neg a \lor \neg b \lor \neg c)$ vtt $\{a \mapsto 1, \dots\} \models (\neg b \lor \neg c)$
  - Mimochodom, $(\neg b \lor \neg c)$ je rezolventa a a $(\neg a \lor \neg b \lor \neg c)$

Takže:

- Klauzuly s a môžeme **vynechať**
- Klauzuly s $\neg a$ môžeme **zjednodušiť**

Zjednodušenie množiny klauzúl podľa literálu

Množinu klauzúl

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

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

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

Analogicky môžeme S zjednodušiť podľa $\neg a$ na

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

Zjednodušenie množiny klauzúl podľa literálu

Definícia 2.113

Nech p je výroková premenná.

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

Komplement literálu ℓ označujeme $\overline{\ell}$.

Definícia 2.114

Nech ℓ je literál a S je množina klauzúl. Potom definujeme

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

Tvrdenie 2.115

Nech ℓ je literál a S je množina klauzúl.

Potom množiny $S \cup \{\ell\}$ a $S|_{\ell}$ sú ekvisplniteľné.

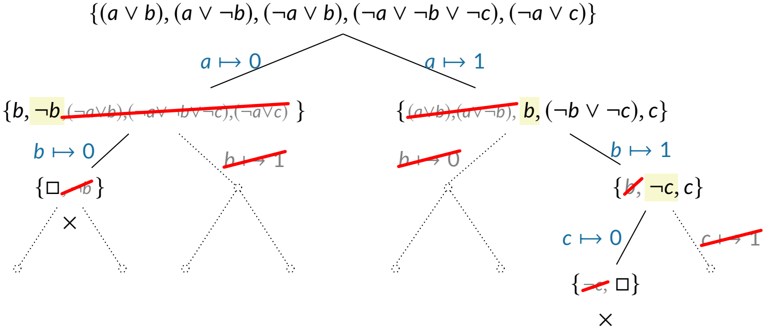
Propagácia jednotkových klauzúl

Zjednodušenie množiny klauzúl $\rightsquigarrow$ ľahšie hľadanie spĺňajúcich ohodnotení:
Nech $T = \{(a \lor \neg b), (a \lor b \lor c)\}$, začnime zjednodušením podľa $\neg a$:

- $T' := T|_{\neg a} = \{\neg b, (b \lor c)\}$
  - $\neg b$ — ***jednotková klauzula*** (*unit clause* alebo iba ***unit***)
  - T' spĺňajú **iba** ohodnotenia v, kde $v(b) = 0$
  - Takže T' zjednodušíme podľa $\neg b$
- $T'' := T'|_{\neg b} = \{c\}$
  - T'' spĺňajú iba ohodnotenia v, kde $v(c) = 1$
  - Takže T'' zjednodušíme podľa c
- $T''' := T''|_{c} = \{\}$
  prázdna, triviálne splniteľná, podľa tvrdenia 2.115 je splniteľná aj T

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

Prehľadávanie so zjednodušovaním klauzúl a unit propagation



Eliminácia nezmiešaných literálov

Všimnime si literál u v množine klauzúl:

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

Literál u je ***nezmiešaný*** (angl. *pure*) v T:
u sa vyskytuje v T, ale jeho komplement $\neg u$ sa tam nevyskytuje
Nech

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

- 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' nesplniteľná, tak je nesplniteľná každá jej nadmnožina, teda aj T

Z hľadiska splniteľnosti sú klauzuly obsahujúce u nepodstatné.
Stačí uvažovať $T|_u$.

Eliminácia nezmiešaných literálov

Definícia 2.116

Nech ℓ je literál a S je množina klauzúl.
Literál ℓ je ***nezmiešaný*** (*pure*) v S vtt ℓ sa vyskytuje v niektorej klauzule z S, ale jeho komplement $\overline{\ell}$ sa nevyskytuje v žiadnej klauzule z S.

Tvrdenie 2.117

Nech ℓ je literál a S je množina klauzúl.
Ak ℓ je nezmiešaný v S, tak S je splniteľná vtt $S|_{\ell}$ je splniteľná.

2.12.3
DPLL

Algoritmus 2.118 (Davis and Putnam [1960], Davis et al. [1962])

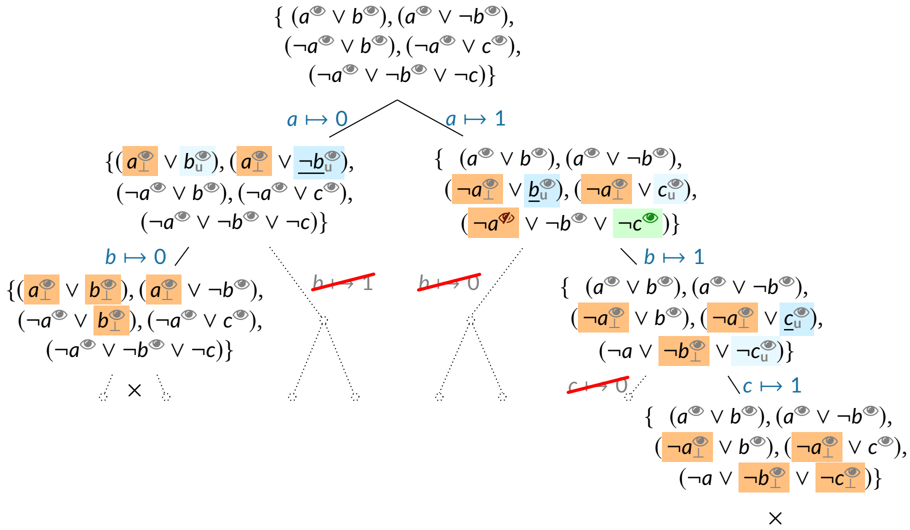
function DPLL(Φ, e)
    if Φ obsahuje prázdnu klauzulu **then**
        return False
    end if
    if e ohodnocuje všetky premenné **then**
        return True
    end if
    while existuje jednotková (unit) klauzula ℓ vo Φ **do**
        $\Phi, e \leftarrow$ unit-propagate(ℓ, Φ, e)
    end while
    while existuje nezmiešaný (pure) literál ℓ vo Φ **do**
        $\Phi, e \leftarrow$ pure-literal-assign(ℓ, Φ, e)
    end while
    $x \leftarrow$ choose-branch-literal(Φ, e)
    return DPLL$(\Phi|_x, e(x \mapsto T))$ **or** DPLL$(\Phi|_{\neg x}, e(x \mapsto F))$
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**:

  $\mathit{predik\acute{a}tov\acute{y}\_symbol}(\mathit{argument}_1, \mathit{argument}_2, \ldots, \mathit{argument}_n)$

  Predikátový symbol ∼ prísudok (alebo celá prísudková časť):

  (je) $\mathtt{minister}$, (je) $\mathtt{star\check{s}\acute{i}}$ (ako), $\mathtt{prijal}$, <, …

  Jeho argumenty ∼ podmet, predmet, …

  Úloha je daná *pozíciou* (ako v prog. 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 ($\mathtt{minister}^1$, $\mathtt{star\check{s}\acute{i}}^2$, $\mathtt{prijal}^4$, $<^2$).

Význam predikátových symbolov

Unárny predikátový symbol (teda s aritou 1)

zvyčajne predstavuje **vlastnosť**

$\mathtt{minister}^1(arg_1)$	arg_1 je minister
$\mathtt{myš\_domová}^1(arg_1)$	arg_1 je myš domová
$\mathtt{ľahká}^1(arg_1)$	arg_1 je ľahká

Binárny, ternárny, … predikátový symbol (s aritou 2, 3, …)

predstavuje **vzťah**

$\mathtt{starší}^2(arg_1, arg_2)$	arg_1 je starší ako arg_2
$\mathtt{medzi}^3(arg_1, arg_2, arg_3)$	arg_1 sa nachádza medzi arg_2 a arg_3
$\mathtt{prijal}^4(arg_1, arg_2, arg_3, arg_4)$	arg_1 prijal arg_2 od arg_3 v čase 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, ...:
  $\texttt{starší}^2(arg_1, arg_2)$ — 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 konštánt**.
  - $\texttt{Dorothy}$, $\texttt{Hlohovský}$, $\texttt{Jerry}$, 0, 1, 2, $\dots$, $\texttt{rok2013}$
- Môžu byť **argumentmi predikátových symbolov** v atomických formulách
  - $\texttt{minister}(\texttt{Hlohovský})$, $\texttt{starší}(\texttt{Dorothy}, \texttt{George})$, $\texttt{prijal}(\texttt{Hlohovský}, \texttt{úplatok250000€}, \texttt{Veselič}, \texttt{rok2013})$
- Samé o sebe **nie sú formulami**, nemajú pravdivostnú hodnotu.
- Dva symboly konštánt môžu označovať ten istý objekt:
  - $\texttt{stvrty\_prezident\_SR}$ a $\texttt{Andrej\_Kiska}$
- **Rovnostné atómy** — špeciálny druh atomických formúl:
  - $\texttt{stvrty\_prezident\_SR} \doteq \texttt{Andrej\_Kiska}$

Výrokové spojky a atomické formuly

- V logike prvého rádu môžeme atomické formuly **spájať výrokovými spojkami** rovnako ako vo výrokovej logike:
  - $((\mathtt{matka}(\mathtt{Dorothy}) \land \mathtt{syn}(\mathtt{George})) \to \mathtt{starší}(\mathtt{Dorothy}, \mathtt{George}))$
  - $(\mathtt{zomrel}(\mathtt{Stephen\_Hawking}) \to \neg\, \mathtt{najznámejší\_žijúci\_fyzik} \doteq \mathtt{Stephen\_Hawking})$
  - $(\mathtt{prijal}(\mathtt{Hlohovský}, \mathtt{úplatok250000€}, \mathtt{Veselič}, \mathtt{rok2013}) \to \neg\mathtt{dôveryhodný}(\mathtt{Hlohovský}))$
- 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 indivíduových 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**
  - $\mathtt{starší}(x, \mathtt{Dorothy})$, $\mathtt{minister}(z_5)$

  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*
  - $\forall x\,\mathtt{starší}(x, \mathtt{Dorothy})$ — každý je starší ako Dorothy
- Kvantifikovaná formula nemusí byť atomická:
  - $\forall x(\mathtt{starší}(x, \mathtt{Dorothy}) \lor \neg\mathtt{starší}(\mathtt{George}, x))$

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*
  - $\exists x\, \mathtt{starší}(x, \mathrm{George})$ — niekto je starší ako George
- Kvantifikovaná formula nemusí byť atomická:
  - $\exists x (\mathtt{starší}(x, \mathrm{George}) \land \mathtt{starší}(\mathrm{Virginia}, x))$
- Kvantifikovaná formula môže obsahovať ďalšie kvantifikátory:
  - $\exists x\, \forall y\, \mathtt{starší}(x, \mathrm{y})$
  - $\forall x \big(\exists y\, \exists u\, \exists z (\mathtt{prijal}(x, u, \mathrm{y}, \mathrm{z}) \land \mathtt{úplatok}(u)) \to \neg\mathtt{dôveryhodný}(x)\big)$

Symboly jazyka relačnej logiky prvého rádu

Definícia 3.3

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

symboly (indivíduových) premenných z nejakej nekonečnej spočítateľnej množiny $\mathcal{V}_{\mathcal{L}}$ (označujeme ich x, y, …);

mimologické symboly, ktorými sú

- ***symboly konštánt*** z nejakej spočítateľnej množiny $\mathcal{C}_{\mathcal{L}}$ (označované a, b, …);
- ***predikátové symboly*** z nejakej spočít. množiny $\mathcal{P}_{\mathcal{L}}$ (ozn. P, R, …);

logické symboly, ktorými sú

- ***logické spojky:*** unárna $\neg$, binárne $\land$, $\lor$, $\to$,
- ***symbol rovnosti*** $\doteq$,
- ***kvantifikátory:*** ***existenčný*** $\exists$ a ***všeobecný*** $\forall$;

pomocné symboly (,) a **,** (ľavá, pravá zátvorka a čiarka).

Množiny $\mathcal{V}_{\mathcal{L}}, \mathcal{C}_{\mathcal{L}}, \mathcal{P}_{\mathcal{L}}$ sú vzájomne disjunktné.
Logické a pomocné symboly sa nevyskytujú v symboloch z $\mathcal{V}_{\mathcal{L}}, \mathcal{C}_{\mathcal{L}}, \mathcal{P}_{\mathcal{L}}$.
Každému symbolu $P \in \mathcal{P}_{\mathcal{L}}$ je priradená ***arita*** $\mathrm{ar}(P) \in \mathbb{N}^+$.

Symboly jazyka relačnej logiky prvého rádu

Poznámka 3.4

Symboly (konštánt, 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.
Symboly premenných z $\mathcal{V}_{\mathcal{L}}$ a konštánt z $\mathcal{C}_{\mathcal{L}}$ 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 $(A \land B)$, $(A \lor B)$, $(A \to B)$ sú formuly z $\mathcal{E}_{\mathcal{L}}$ (***konjunkcia***, ***disjunkcia***, ***implikácia*** A a B).
- Ak x je indivíduová 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, … s prípadnými indexmi.
$(A \leftrightarrow B)$ je skratka postupnosti symbolov $((A \to B) \land (B \to A))$.

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.

Jednoduchá formalizácia

Riešenie

$\mathcal{V}_{\mathcal{L}} = \{x, y, z\}, \mathcal{C}_{\mathcal{L}} = \{\mathrm{Aďa}, \mathrm{Biba}, \mathrm{Ciri}, \mathrm{Dada}\}, \mathcal{P}_{\mathcal{L}} = \{\mathrm{r}^2\}$

$\mathrm{r}(x, y)$ znamená „*x* má rada *y*“

1. $(\mathrm{r}(\mathrm{Biba}, \mathrm{Ciri}) \lor \mathrm{r}(\mathrm{Biba}, \mathrm{Dada}))$
2. $\forall x(\mathrm{r}(\mathrm{Biba}, x) \to \mathrm{r}(\mathrm{Aďa}, x))$
3. $\forall x(\mathrm{r}(x, \mathrm{Ciri}) \to \mathrm{r}(\mathrm{Ciri}, x))$
4. $\exists x(\mathrm{r}(x, \mathrm{Biba}) \land \mathrm{r}(\mathrm{Biba}, x))$
5. $\neg \exists x\, \mathrm{r}(x, x)$
6. $\forall x\, \exists y\, \mathrm{r}(x, y)$
7. $\exists x\, \forall y\, \mathrm{r}(y, x)$

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*.“:
  - $\forall x(P(x) \rightarrow Q(x))$
- „Niekto, kto má vlastnosť *P*, má vlastnosť *Q*.“/ „Niektorý *P* je *Q*.“
  - $\exists x(P(x) \land Q(x))$

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

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

- S dôrazom na zámeno:
  - $\neg\exists x\, (\mathtt{vegán}(x) \land \mathtt{obézny}(x))$
- S dôrazom na negatívne tvrdenie:
  - $\forall x\, \neg(\mathtt{vegán}(x) \land \mathtt{obézny}(x))$
  - $\forall x\, (\neg\mathtt{vegán}(x) \lor \neg\mathtt{obézny}(x))$
  - $\forall x\, (\mathtt{vegán}(x) \to \neg\mathtt{obézny}(x))$

Základné idiómy: Zamlčaná a zdanlivá existencia

Zamlčaná existencia

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

Existencia v antecedente s odkazom v konzekvente

- ak je *niekto* vegán, tak (*on*) nie je obézny:
  - „nie je obézny" sa odvoláva na „niekto"
  - „niekto" teraz nevyjadruje existenciu!
  - $\forall x(\mathtt{vegán}(x) \rightarrow \neg\mathtt{obézny}(x))$

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 $\forall x\big(\exists y(\mathtt{kúpil}(x, y) \land \mathtt{karfiol}(y)) \to \mathtt{vegán}(x)\big)$

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 *ne*kúpil tekvicu, tak *nie* je vegán."
  „Pre každého platí, že ak je vegán, tak si kúpil tekvicu."
- Formalizácia je teda $\forall x\big(\mathtt{veg\acute{a}n}(x) \to \exists y(\mathtt{k\acute{u}pil}(x, y) \land \mathtt{tekvica}(y))\big)$

3.2.4
Idiómy s rovnosťou

Idiómy s rovnosťou: Enumerácia

Vymenovanie objektov s vlastnosťou

- V byte č. 14 bývajú Aďa, Biba, Ciri, Dada.
  - $\big(\texttt{býva\_v}(\texttt{Aďa}, \texttt{byt14}) \land \cdots \land \texttt{býva\_v}(\texttt{Dada}, \texttt{byt14})\big)$

  Ekvivalentne:
  Každá z Aďa, Biba, Ciri, Dada býva v byte č. 14.
  - $\forall x\big((x \doteq \texttt{Aďa} \lor \cdots \lor x \doteq \texttt{Dada}) \to \texttt{býva\_v}(x, \texttt{byt14})\big)$
- V byte č. 14 bývajú *iba* Aďa, Biba, Ciri, Dada.
  Každý, kto býva v byte č. 14, je jedna z Aďa, Biba, Ciri, Dada.
  - $\forall x\big(\texttt{býva\_v}(x, \texttt{byt14}) \to (x \doteq \texttt{Aďa} \lor \cdots \lor x \doteq \texttt{Dada})\big)$

Idiómy s rovnosťou: Obmedzenia počtu

Aspoň *k*:

- Jaro si kúpil aspoň dve tekvice.
- Existujú dve *navzájom rôzne* tekvice, ktoré si Jaro kúpil.
  - $\exists t_1 \exists t_2 \big(\boldsymbol{\neg\, t_1 \doteq t_2} \land \texttt{tekvica}(t_1) \land \texttt{tekvica}(t_2) \land \texttt{kúpil}(\texttt{Jaro}, t_1) \land \texttt{kúpil}(\texttt{Jaro}, t_2)\big)$

Najviac jeden:

- Anka si kúpila najviac jednu tekvicu.
- Ekvivalentne: Anka si nekúpila aspoň dve tekvice.
  - $\boldsymbol{\neg}\exists t_1 \exists t_2 \big(\neg\, t_1 \doteq t_2 \land \texttt{tekvica}(t_1) \land \texttt{tekvica}(t_2) \land \texttt{kúpil}(\texttt{Anka}, t_1) \land \texttt{kúpil}(\texttt{Anka}, t_2)\big)$
  - $\forall t_1 \forall t_2 \neg\big(\neg\, t_1 \doteq t_2 \land \texttt{tekvica}(t_1) \land \texttt{tekvica}(t_2) \land \texttt{kúpil}(\texttt{Anka}, t_1) \land \texttt{kúpil}(\texttt{Anka}, t_2)\big)$
  - $\forall t_1 \forall t_2 \big(t_1 \doteq t_2 \lor \neg\texttt{tekvica}(t_1) \lor \neg\texttt{tekvica}(t_2) \lor \neg\texttt{kúpil}(\texttt{Anka}, t_1) \lor \neg\texttt{kúpil}(\texttt{Anka}, t_2)\big)$
  - $\forall t_1 \forall t_2 \big((\texttt{tekvica}(t_1) \land \texttt{tekvica}(t_2) \land \texttt{kúpil}(\texttt{Anka}, t_1) \land \texttt{kúpil}(\texttt{Anka}, t_2)) \to t_1 \doteq t_2\big)$
- 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: neúplnost, složitost, nutnost*. Academia, 2002. Prístupné aj na http://www1.cuni.cz/~svejdar/book/LogikaSve2002.pdf.