Prednášky z Matematiky (4) – Logiky pre informatikov

Ján Kľuka, Jozef Šiška

Katedra aplikovanej informatiky
FMFI UK Bratislava

Letný semester 2016/2017

8. prednáška

SAT solver a algoritmus DPLL
Štruktúry

10. apríla 2017

Obsah 8. prednášky

3.13
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 3.108

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

- Tabuľkovou metódou skúmame *všetky* ohodnotenia výrokových premenných
- Preskúmanie ohodnotení trvá $\mathrm{O}(s2^N)$ 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 $\mathrm{O}(k2^N)$, kde k je počet klauzúl
- Tabuľka slúži aj ako dôkaz nesplniteľnosti

3.13.1
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: $O(s+N)$;

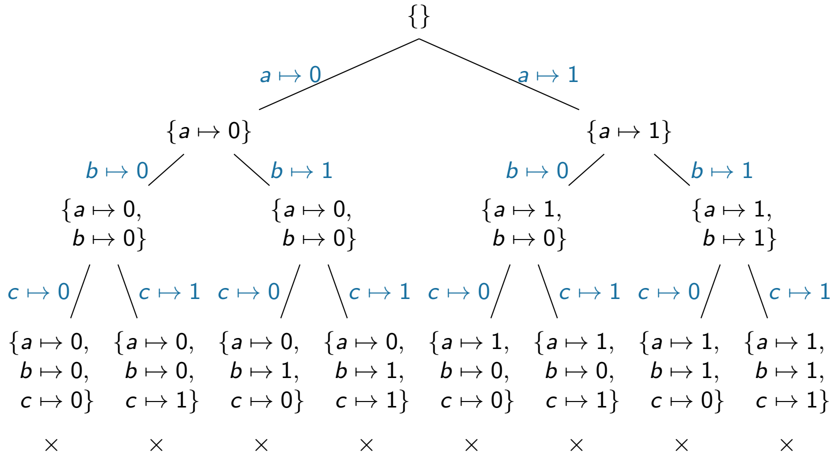
N — počet premenných,

s — súčet veľkostí klauzúl

Strom prehľadávania ohodnotení

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

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



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

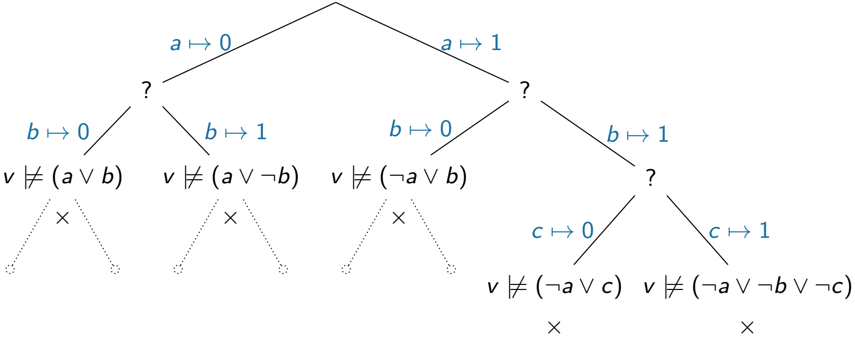
3.13.2
Optimalizácia backtrackingu

Priebežné vyhodnocovanie klauzúl

- Každý uzol prehľadávaného stromu ohodnotení je *čiastočné 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

$S = \{(a \vee b), (a \vee \neg b), (\neg a \vee b), (\neg a \vee \neg b \vee \neg c), (\neg a \vee c)\}$
$\times$: $v \not\models S$



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
  - $\{a \mapsto 1, \dots\} \models (a \lor b)$
  - $\{a \mapsto 1, \dots\} \models (a \lor \neg b)$

  Tieto klauzuly sú pre zistenie splniteľnosti vo všetkých v' *nepodstatné*, môžeme ich vynechať
- v' splní klauzulu $(\ell_1 \lor \cdots \lor \neg a \lor \cdots \lor \ell_n)$ obsahujúcu $\neg a$ vtt v' 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 $(\neg a \lor \neg b \lor \neg c)$

  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

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

Analogicky môžeme S zjednodušiť podľa $\neg a$ na

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

Zjednodušenie množiny klauzúl podľa literálu

Definícia 3.109

Nech p je výroková premenná.
Komplementom literálu p je $\neg p$. *Komplementom literálu* $\neg p$ je p.
Komplement literálu ℓ označujeme $\overline{\ell}$.

Definícia 3.110

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

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

Tvrdenie 3.111

Nech ℓ 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 $T = \{(a \lor \neg b), (a \lor b \lor c)\}$ podľa $\neg a$ dostaneme $T' := T|_{\neg a} = \{\neg b, (b \lor c)\}$
- 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

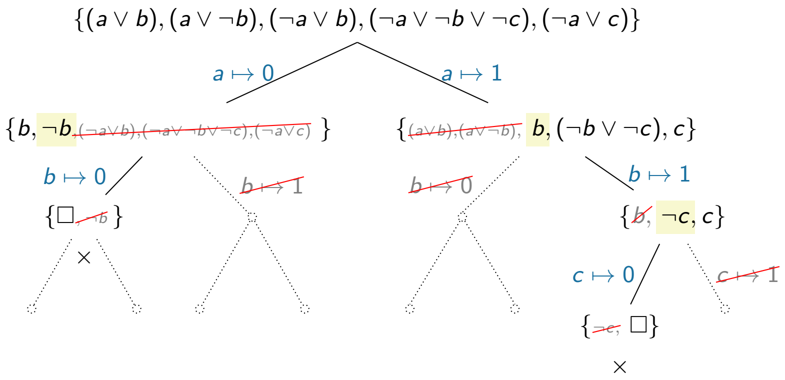
Proces opakovaného rozširovania ohodnotení podľa jednotkových klauzúl a zjednodušovania sa nazýva *propagácia jednotkových klauzúl* (*unit propagation*)

Propagácia jednotkových klauzúl

Dôsledok 3.112

Nech ℓ je literál a S je množina klauzúl obsahujúca jednotkovú klauzulu ℓ ($\ell \in S$). 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:

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

- Hovoríme, že 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:

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

  - 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

  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 3.113

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 3.114

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

3.13.3
DPLL

Algoritmus 3.115 (Davis and Putnam [1960], Davis et al. [1962])

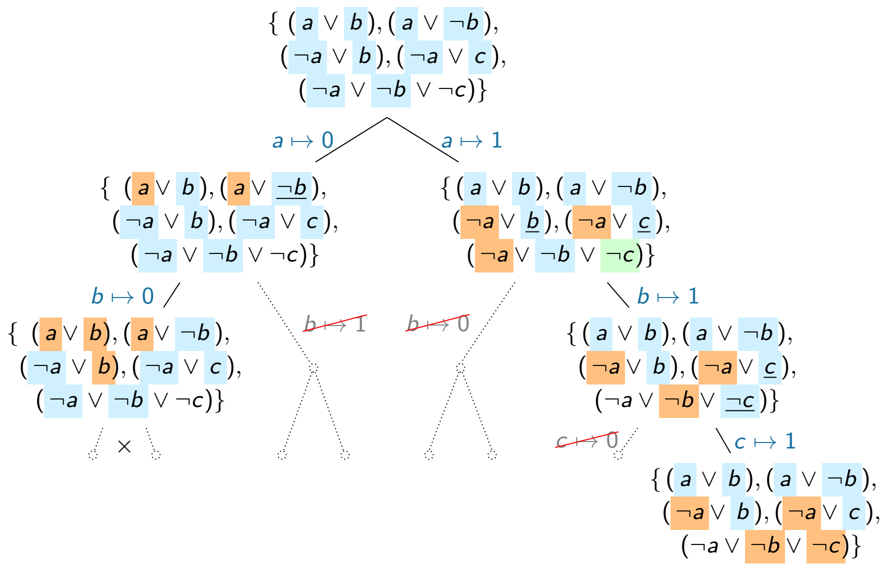
function $\mathrm{DPLL}(\Phi, e)$
    if Φ obsahuje prázdnu klauzulu **then**
        return False
    end if
    if e ohodnocuje všetky premenné **then**
        return True
    end if
    while existuje jednotková (unit) klauzula ℓ vo Φ **do**
        $\Phi, e \leftarrow \textsc{unit-propagate}(\ell,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



4.1
Syntax výrokovej logiky s rovnosťou

Štruktúra výrokov — objekty a vzťahy

Jazyk výrokovej logiky nie je najpohodlnejší na zápis problémov, v ktorých sa opakujú *vlastnosti* alebo *vzťahy*, ktoré sa dajú aplikovať na viacero *objektov*:

- V probléme typickej americkej rodiny sme mali napríklad vlastnosť „x je dcéra“ alebo vzťah „x je staršia/-í ako y“. Objektmi vlastností a vzťahov boli Dorothy, George, Howard a Virginia
- V probléme vraždy v dreadburskom panstve sme mali napríklad vzťahy „x je bohatší/-ia ako y“, „x nenávidí y“. Objektmi boli Agáta, komorník (Butler) a Karol (Charles)
- V online bazári vzniká vzťah „x kupuje od y tovar z“. Objektmi sú rôzni konkrétni predávajúci a kupujúci, rôzne konkrétne tovary

Štruktúra výrokov — jednoznačne určené objekty

V niektorých vzťahoch je ku každému objektu *práve jeden* objekt (alebo hodnota) — teda súvisiaci objekt vždy existuje a je jednoznačne určený:

- každý človek má práve jednu biologickú matku,
- každý kus tovaru v bazári má práve jednu aktuálnu cenu,
- každý študent dostane za každú úlohu práve jedno hodnotenie,
- súčet každej dvojice čísel je práve jeden.

Takýto jednoznačne určený objekt (hodnotu) vieme pomenovať, aj keď nemá vlastné meno, pomocou zdrojového objektu a vzťahu:

- Emina mama,
- cena tovaru č. 531246,
- Jarkino hodnotenie z midtermu,
- súčet 1 a 1.

Krok k štruktúrovanejším výrokom

Výroková logika veľmi zjednodušuje prirodzený jazyk:

- skúma iba štruktúru tvrdení tvorenú spojkami,
- atomické výroky *nemajú štruktúru*

Spravme teraz **malý** krok k logike, ktorá vyjadrí zložitejšie tvrdenia. Zachyťme:

- **konkrétne objekty**,
- **vlastnosti** a **vzťahy**,
- nepriamo pomenované **jednoznačne určené objekty**

ale **nepokúšajme** sa zatiaľ o zámená (niekto), či číslovky (všetci, práve dve)

Symboly jazyka výrokovej logiky s rovnosťou

Definícia 4.1

Symbolmi jazyka výrokovej logiky s rovnosťou $\mathcal{L}$ sú:

- *mimologické* symboly:
  - *symboly konštánt* z nejakej spočítateľnej množiny $\mathcal{C}_{\mathcal{L}}$ $(a, b, \dots)$;
  - *funkčné symboly* z nejakej spočítateľnej množiny $\mathcal{F}_{\mathcal{L}}$ $(f, g, \dots)$;
  - *predikátové symboly* z nejakej spočít. množiny $\mathcal{P}_{\mathcal{L}}$ $(P, R, \dots)$;
- *logické symboly*:
  - *logické spojky*: unárna $\neg$, binárne $\land$, $\lor$, $\rightarrow$;
  - *symbol rovnosti* $\doteq$ (niekedy zapisovaný priamo ako $=$);
- *pomocné symboly* (,) a , (ľavá, pravá zátvorka a čiarka);

Množiny $\mathcal{C}_{\mathcal{L}}$, $\mathcal{F}_{\mathcal{L}}$, $\mathcal{P}_{\mathcal{L}}$ sú navzájom disjunktné. Logické a pomocné symboly sa nevyskytujú v symboloch z týchto množín.
Každému symbolu $S \in \mathcal{P}_{\mathcal{L}} \cup \mathcal{F}_{\mathcal{L}}$ je priradená *arita* (počet argumentov) $\mathrm{ar}(S) \in \mathbb{N}^+$.

Symboly jazyka logiky s rovnosťou

Poznámka 4.2

Symboly (konštánt, funkčné, predikátové) môžu byť nealfabetické (1, $<$, $+$), či tvorené viacerými znakmi ($\texttt{Virginia}$, $\texttt{dcéra}$, $\texttt{cena}$).

Dohoda 4.3

Aritu budeme niekedy písať ako horný index symbolov ($\texttt{matka}^1$, $<^2$).

Príklady a účel symbolov

Príklad 4.4

Symboly konštánt predstavujú *konkrétne* objekty alebo hodnoty, podobne ako *vlastné mená* v prirodzenom jazyku alebo konštanty v programovacom jazyku:

- $\texttt{Agatha}$, $\texttt{Ema}$, $\texttt{Tovar531246}$, 0, 1

Predikátové symboly predstavujú *vlastnosti* a *vzťahy*:

- $\texttt{kolobežka}^1$, $\texttt{syn}^1$, $\texttt{nenávidí}^2$, $\texttt{kupuje}^3$, $<^2$

Funkčné symboly predstavujú *vzťahy s jednoznačne určenými objektmi*:

- $\texttt{cena}^1$, $\texttt{hodnotenie}^2$, $+^2$

Termy jazyka logiky s rovnosťou

Definícia 4.5

Množina $\mathcal{T}_{\mathcal{L}}$ *termov* jazyka logiky s rovnosťou $\mathcal{L}$ je najmenšia množina postupností symbolov jazyka $\mathcal{L}$, pre ktorú platí:

- každý symbol konštanty $c \in \mathcal{C}_{\mathcal{L}}$ je termom;
- ak f je funkčný symbol s aritou n a $t_1, \ldots, t_n$ sú termy, tak aj $f(t_1, \ldots, t_n)$ je termom.

Inak povedané:

- $\mathcal{C}_{\mathcal{L}} \subseteq \mathcal{T}_{\mathcal{L}}$.
- Ak $f \in \mathcal{F}_{\mathcal{L}}$, $\mathrm{ar}(f) = n$ a $t_1, \ldots, t_n \in \mathcal{T}_{\mathcal{L}}$, tak aj $f(t_1, \ldots, t_n) \in \mathcal{T}_{\mathcal{L}}$.

Dohoda 4.6

Termy označujeme písmenami t, s, r s prípadnými dolnými indexmi.

Termy jazyka logiky s rovnosťou

Príklad 4.7

Termy predstavujú konkrétne objekty — buď priamo pomenované symbolmi konštánt:

- Agatha, Ema, $\mathrm{Tovar531246}$, 0, 1

alebo nepriamo pomenované pomocou jednoznačných vzťahov:

- $\mathrm{matka}(\mathrm{Ema})$, $\mathrm{cena}(\mathrm{Tovar531246})$, $\mathrm{predávajúci}(\mathrm{Tovar531246})$, $+(0, 1)$.

Termy možno ľubovoľne vnárať:

- $\mathrm{matka}(\mathrm{matka}(\mathrm{matka}(\mathrm{Ema})))$, $+(+(1, 0), +(1, 1))$, $\mathrm{cena}(\mathrm{predávajúci}(\mathrm{Tovar531246}))$.

Vidíme, že používanie funkčných symbolov na označenie vzťahov má úskalia. :)

Atomické formuly jazyka logiky s rovnosťou

Definícia 4.8 (Atomické formuly)

Nech $\mathcal{L}$ je jazyk logiky s rovnosťou.

- Ak t_1 a t_2 sú termy, tak postupnosť symbolov $t_1 \doteq t_2$ nazývame *rovnostný atóm* jazyka $\mathcal{L}$.
- Ak P je predikátový symbol s aritou n a t_1, ..., t_n sú termy, tak postupnosť symbolov $P(t_1, \dots, t_n)$ nazývame *predikátový atóm* jazyka $\mathcal{L}$.
- Rovnostné a predikátové atómy jazyka $\mathcal{L}$ spoločne nazývame *atomickými formulami* (skrátene *atómami*) jazyka $\mathcal{L}$.
- Množinu všetkých atómov jazyka $\mathcal{L}$ označujeme $\mathcal{A}_{\mathcal{L}}$.

Príklady atomických formúl

Príklad 4.9

Predikátové atomické formuly predstavujú výroky o vlastnostiach objektov označených termami:

- $\mathrm{bicykel}(\mathrm{Tovar531246})$, $\mathrm{žena}(\mathrm{matka}(\mathrm{Miro}))$, $\mathrm{párne}(+(1, 1))$,

a o vzťahoch objektov:

- $\mathrm{starší}(\mathrm{Howard}, \mathrm{Virginia})$, $\mathrm{dieťa}(\mathrm{Miro}, \mathrm{matka}(\mathrm{Ema}))$, $<(+(1, 1), 0)$, $\mathrm{kupuje}(\mathrm{Jofi22}, \mathrm{Katulienka}, \mathrm{Tovar531246})$.

Rovnostné atómy vyjadrujú, že dva termy označujú ten istý objekt:

- $\mathrm{Butler} \doteq \mathrm{Charles}$, $\mathrm{matka}(\mathrm{Miro}) \doteq \mathrm{matka}(\mathrm{Ema})$, $+(1, 0) \doteq 1$.

Formuly jazyka logiky s rovnosťou

Definícia 4.10

Množina $\mathcal{E}_{\mathcal{L}}$ *formúl* jazyka logiky s rovnosťou $\mathcal{L}$ je najmenšia množina postupností symbolov jazyka $\mathcal{L}$, pre ktorú platí:

- Všetky atomické formuly z $\mathcal{A}_{\mathcal{L}}$ sú formulami.
- Ak A je formula, tak aj $\neg A$ je formula (*negácia* A).
- Ak A a B sú formuly, tak aj $(A \land B)$, $(A \lor B)$, $(A \to B)$ sú formuly (*konjunkcia*, *disjunkcia*, *implikácia* A a B).

Dohoda

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

Formuly jazyka logiky s rovnosťou

Príklad 4.11

Formuly tvoríme z atómov tak, ako doteraz:

$$(\text{dieťa}(\text{Miro}, \text{matka}(\text{Ema})) \to \text{matka}(\text{Miro}) \doteq \text{Ivana})$$

$$\begin{aligned}&(\text{killed}(\text{Charles}, \text{Agatha}) \to \\ &\qquad (\text{hates}(\text{Charles}, \text{Agatha}) \land \neg\text{richer}(\text{Charles}, \text{Agatha})))\end{aligned}$$

$$(\neg\, \text{Charles} \doteq \text{Butler} \to \text{hates}(\text{Agatha}, \text{Charles}))$$

Zjednodušenie zápisu formúl

Dohoda 4.12

Zápis formúl môžeme zjednodušovať nasledujúcim spôsobom:

- Negáciu rovnostného atómu $\neg\, s \doteq t$ skrátene zapisujeme $s \not\doteq t$.
- Vonkajší pár zátvoriek môžeme vždy vynechať, teda napr. namiesto $(a \doteq b \to b \doteq a)$ môžeme písať $a \doteq b \to b \doteq a$.
- Binárnym spojkám priradíme prioritu:
  najvyššiu má $\land$, nižšiu $\lor$, najnižšiu $\to$.
- Ak $Z = (A\, b_1\, B)$ je priamou podformulou $(X\, b_2\, Y)$ (teda $Z = X$ alebo $Z = Y$) a b_1 má vyššiu prioritu ako b_2, môžeme vynechať zátvorky okolo Z.

Príklad 4.13

Namiesto $((P(a,b) \land (P(c,a) \lor P(b,c))) \to (P(a,c) \lor P(c,a)))$ môžeme písať $P(a,b) \land (P(a,c) \lor P(b,c)) \to P(a,c) \lor P(c,a)$.

4.2
Sématika logiky s rovnosťou

Štruktúry

Definícia 4.14

Nech $\mathcal{L}$ je jazyk logiky s rovnosťou.
Štruktúrou pre jazyk $\mathcal{L}$ nazývame dvojicu $\mathcal{M} = (M, i)$, kde

- M je neprázdna množina, nazývaná *doména* štruktúry $\mathcal{M}$;
- i je zobrazenie, nazývané *interpretačná funkcia* štruktúry $\mathcal{M}$, ktoré
  - každému symbolu konštanty c jazyka $\mathcal{L}$ priraďuje prvok $i(c) \in M$;
  - každému funkčnému symbolu f jazyka $\mathcal{L}$ s aritou n priraďuje funkciu $i(f)\colon M^n \to M$;
  - každému predikátovému symbolu P jazyka $\mathcal{L}$ s aritou n priraďuje množinu $i(P) \subseteq M^n$.

Dohoda 4.15

Štruktúry označujeme veľkými *písanými* písmenami $\mathcal{M}$, $\mathcal{N}$, ...
Doménu označujeme rovnakým, ale *tlačeným* písmenom ako štruktúru.

Štruktúry

Príklad 4.16

Nájdime štruktúru pre jazyk $\mathcal{L}_{\text{Rodina}}$ pre zjednodušené rodinné vzťahy so symbolmi konštánt $\mathtt{Ema}$, $\mathtt{Miro}$, $\mathtt{Ivana}$, predikátovými symbolmi $\mathtt{žena}^1$ a $\mathtt{dieťa}^2$, a funkčným symbolom $\mathtt{matka}^2$.

Hodnota termov

Definícia 4.17

Nech $\mathcal{M} = (M, i)$ je štruktúra pre jazyk $\mathcal{L}$.
Hodnotou termu t jazyka $\mathcal{L}$ v štruktúre $\mathcal{M}$ je prvok z M označovaný $t^{\mathcal{M}}$, ktorý je určený nasledovne:

- $a^{\mathcal{M}} = i(a)$, ak a je konštanta,
- $(f(t_1, \dots, t_n))^{\mathcal{M}} = f^{\mathcal{M}}(t_1^{\mathcal{M}}, \dots, t_n^{\mathcal{M}})$, ak f je funkčný symbol a t_1, …, t_n sú termy.

Príklad 4.18

Vyhodnoťme termy `Ivana`, `matka(Miro)`, `matka(matka(Ema))` v štruktúre z predchádzajúceho príkladu.

Splnenie formuly v štruktúre

Definícia 4.19

Nech $\mathcal{L}$ je jazyk výrokovej logiky s rovnosťou.
Relácia *štruktúra $\mathcal{M}$ spĺňa formulu A* (skrátene $\mathcal{M} \models A$) medzi formulami $\mathcal{L}$ a štruktúrami pre $\mathcal{L}$ je definovaná pre každú štruktúru $\mathcal{M} = (M, i)$ induktívne vzhľadom na stupeň formuly nasledovne:

- $\mathcal{M} \models t_1 \doteq t_2$ vtt $t_1^{\mathcal{M}} = t_2^{\mathcal{M}}$,
- $\mathcal{M} \models P(t_1, \ldots, t_n)$ vtt $(t_1^{\mathcal{M}}, \ldots, t_n^{\mathcal{M}}) \in i(P)$,
- $\mathcal{M} \models \neg A$ vtt $\mathcal{M} \not\models A$,
- $\mathcal{M} \models (A \wedge B)$ vtt $\mathcal{M} \models A$ a zároveň $\mathcal{M} \models B$,
- $\mathcal{M} \models (A \vee B)$ vtt $\mathcal{M} \models A$ alebo $\mathcal{M} \models B$,
- $\mathcal{M} \models (A \rightarrow B)$ vtt $\mathcal{M} \not\models A$ alebo $\mathcal{M} \models B$,

pre každú aritu $n > 0$, každý predikátový symbol P s aritou n, všetky termy t_1, t_2, ..., t_n, a všetky formuly A, B.

Splnenie formuly v štruktúre

Príklad 4.20

Zistime, či sú v štruktúre z príkladu 4.16 splnené formuly:

- $\mathrm{dieťa}(\mathrm{Ema}, \mathrm{Ivana})$,
- $\mathrm{matka}(\mathrm{Ema}) \not\doteq \mathrm{Ema}$,
- $\mathrm{dieťa}(\mathrm{Miro}, \mathrm{matka}(\mathrm{Ema})) \to \mathrm{matka}(\mathrm{Miro}) \doteq \mathrm{Ivana}$.

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.

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.