AD4M33AU
Automatické uvažování

Rezoluční kalkulus
pro logiku prvního řádu

Petr Pudlák

Logika prvního řádu

(Někdy nepřesně nazývaná predikátová logika.)

- Výhody
  - Vyšší vyjadřovací schopnost jazyka,
  - V podstatě veškerou současnou matematiku lze postihnout uvnitř logiky prvního řádu.
- Nevýhody
  - Podstatně těžší dokazování/rozhodování tvrzení.
  - Úloha nalezení modelu (ekvivalentně vyvrácení domněnky) není algoritmicky řešitelná.

Univerzum

- Logika 1. řádu má za cíl umožnit popis struktur s mnoha (i nekonečně mnoho) objekty.
- Jazyk logiky 1. řádu umožňuje pomocí proměnných, funkčních a predikátových symbolů popisovat vztahy mezi těmito objekty.

Jazyk predikátové logiky (TPTP)

- *proměnná* ::= symbol začínající <u>velkým</u> písmenem
- *funční_symbol* ::= symbol začínající <u>malým</u> písmenem
- *term* ::= *proměnná*
  | *funkční_symbol* (arita 0)
  | *funkční_symbol* *'('* *term*$_1$ *','* ... *','* *term*$_n$ *')'* (arita n>0)
- *predikátový_symbol* ::= symbol začínající <u>malým</u> písmenem
- *atomická_formule* ::= *predikátový_symbol* (arita 0)
  | *predikátový_symbol* *'('* *term*$_1$ *','* ... *','* *term*$_n$ *')'* (arita n>0)
- *formule ::=*
  atomická_formule
  | *'(' '~'* *formule* *')'*
  | *'('* *formule* binární logická spojka *formule* *')'*
  | *'!['* *proměnná* *']'* *formule* (univerzální kvantifikátor)
  | *'?['* *proměnná* *']'* *formule* (existenční kvantifikátor)

Volné proměnné

- Definice: Množina *FV(t)* **volných proměnných termu** *t* je definována následovně:
  - $FV(X) = \{ X \}$
  - $FV(f(t_1,...,t_n)) = FV(t_1) \cup ... \cup FV(t_n)$

Volné proměnné

- Definice: Množina $FV(\varphi)$ **volných proměnných formule** φ je definována následovně:
  - $FV(p(t_1,...,t_n)) = FV(t_1) \cup ... \cup FV(t_n)$
  - $FV(\neg\varphi) \quad = FV(\varphi)$
  - $FV(\varphi \mid \psi) \quad = FV(\varphi) \cup FV(\psi)$
    - *a stejně pro všechny další spojky*
  - $FV((\forall X)\varphi(X)) \quad = FV(\varphi) \setminus \{ X \}$
  - $FV((\exists X)\varphi(X)) \quad = FV(\varphi) \setminus \{ X \}$

Vázané proměnné

- Proměnné, které se vyskytují ve formuli, a které nejsou volné, obvykle označujeme **vázané**.
- Vázané proměnné můžeme ve formuli přejmenovávat za nové proměnné, které ve formuli ještě nejsou (příklad).
- Stejná proměnná může být ve formuli současně volná i vázaná.
- V takovém případě obvykle vázanou proměnnou přejmenováváme, abychom předešli nedorozumění.

Uzavřené a otevřené formule

- Formuli φ nazveme **uzavřenou**, pokud $FV(\varphi) = \{\}$.
- Formuli ψ nazveme **otevřenou**, pokud neobsahuje vázané proměnné (jinak řečeno, neobsahuje kvantifikátory).

Interpretace

- Interpretace a ohodnocení přiřazují význam symbolům jazyka v rámci nějaké struktury.
- **Interpretace** *M* definuje:
  - Neprázdnou množinu prvků $\mathbf{D}_M$.
  - Pro každý funkční symbol *f* arity *n* definuje <u>totální</u> funkci $f_M : D_M^n \rightarrow D_M$.
  - Pro každý predikátový symbol *p* arity *n* definuje relaci $p_M \subseteq D_M^n$.
- **Ohodnocení** *e* pro danou interpretaci *M* je zobrazení přiřazující každé proměnné prvek z D_M.

Interpretace a model

- Pro danou interpretaci *M* a ohodnocení *e* můžeme dosadit za proměnné dané formule φ, vyčíslit hodnoty všech termů, dosadit relací a vypočítat pravdivost formule.
- Značení: M,e ⊧ φ
- Jestliže je formule φ pravdivá v dané interpretaci *M* <u>pro všechna ohodnocení</u>, říkáme, že **M je model φ**, značíme: **M ⊧ φ**

Sémantický důsledek

- Jestliže ψ platí ve všech interpretacích, ve kterých platí φ, říkáme, že
  ψ je sémantický důsledek φ.
- Značení: $\boldsymbol{\varphi \models \psi}$
- Pro množinu formulí $\boldsymbol{A}$ také píšeme $\boldsymbol{A \models \varphi}$, pokud φ platí ve všech interpretacích, ve kterých jsou splněny všechny formule z $\boldsymbol{A}$.
- Pozorování: Relace ⊧ na formulích je reflexivní a tranzitivní.

Sémantický důsledek – příklady

- $p(X) \models p(f(X))$
- $p(X) \models p(f(Y))$
- $(\forall X)\ p(X) \models p(X)$
- $p(X) \models (\forall X)\ p(X)$
- $p(X) \models (\exists X)\ p(X)$
- $(\exists U)(\forall V)\ q(V,U) \models (\forall X)(\exists Y) q(X,Y)$

Vztah mezi výrokovou logikou a logikou 1. řádu

- Na výrokovou logiku můžeme nahlížet jako na speciální případ logiky 1. řádu, kde se omezíme pouze na predikáty arity 0.
- V takovém případě formule neobsahují termy.
- Nulární predikáty odpovídají výrokovým proměnným.

Substituce

- **Substituce** je zobrazení, které každé proměnné přiřazuje nějaký term (v rámci daného jazyka).
- Značení: θ, σ, ...
- Toto zobrazení lze přirozeně rozšířit na termy a formule:
  Každou <u>volnou</u> proměnnou v zápisu termu či formule nahradíme termem předepsaným danou substitucí (příklad).
- <u>Věta:</u> Pro lib. substituci θ a lib. formuli φ platí $\varphi \models \varphi\theta$

Problém sémantického zjišťování pravdivosti

- Ve výrokové logice je sémantické zjišťování pravdivosti sice neefektivní, ale možné.
- V logice 1. řádu můžeme v některých případech najít konečnou interpretaci, ve které je daná formule splněná .
- Nemůžeme ale sémanticky dokázat pravdivost formule ve všech interpretacích, protože možných interpretací je nekonečně mnoho a mohou být nekonečné.
- V logice 1. řádu tedy nezbytně potřebujeme deduktivní kalkuly.

Problém: nekonečně mnoho možných substitucí

- Máme-li formuli $(\forall X)\varphi(X)$, můžeme odvodit $\varphi(c)$, $\varphi(f(X))$, $\varphi(f(c))$, $\varphi(f(f(f(f(f(h(X,c)))))))$, atd.
- Jak poznat, které substituce pro důkaz potřebujeme a které ne?
- **Rezoluční kalkulus** (originálně pro logiku 1. řádu) vymyslel John Alan Robinson (1965), toto řeší.
- Určuje, jaké nejobecnější substituce je třeba provádět.

Rezoluční kalkulus

- Podobně jako ve výrokové logice existuje rezoluční kalkulus pro logiku 1. řádu.
- Rezoluční kalkulus slouží ke zjištění, zda je daná množina formulí sporná, tedy zda v žádné interpretaci nelze splnit všechny dané formule.
- Rezoluční kalkulus vyžaduje, aby všechny formule byly tzv. **klauzule**.

Literály a klauzule pro logiku 1. řádu

- **Atomická formule** je predikát aplikovaný na termy.
  Literál je atomická formule nebo její negace.
- **Klauzule** je disjunkce libovolného počtu literálů. Klauzule neobsahují kvantifikátory a proměnné jsou implicitně univerzálně kvantifikované.
- Příklad: f(X) | g(c) | ~f(h(X,Y)) | u
- **Prázdná klauzule** je klauzule, která neobsahuje žádný literál. Značení ☐ či ⊥.
- Klauzule se obvykle uvažují jako množiny literálů.

Unifikátor

- **Unifikátor** dvou termů s a t je taková substituce θ, že $s\theta \equiv t\theta$.
- Unifikátor θ nazveme **nejobecnější** (angl. MGU), pokud pro každý unifikátor σ termů s a t existuje substituce η taková, že $\sigma = \theta \circ \eta$
- Stejně definujeme (nejobecnější) unifikátor dvou atomických formulí.
- (Nejobecnější) unifikátor více termů $s_1,\ldots,s_n$ je substituce θ taková, že $s_1\theta \equiv \ldots \equiv s_n\theta$.

Existence nejobecnějšího unifikátoru

- Věta: Mají-li dva termy (atomické formule) unifikátor, mají nejobecnější unifikátor.
- Existuje algoritmus, který najde nejobecnější unifikátor a nebo zjistí, že neexistuje.
- Věta platí i pro více než dva termy (formule).

Odvozovací pravidlo rezoluce v logice 1. řádu

- **Pravidlo rezoluce:**

$$\frac{\Delta \mid L \qquad \sim K \mid \Gamma}{\Delta\theta \mid \Gamma\theta}$$

kde:

- Δ a Γ jsou disjunkce libovolného počtu literálů (klauzule).
- Klauzule ($\Delta \mid L$) a ($\sim K \mid \Gamma$) mají disjunktní množiny proměnných; pokud nemají, proměnné přejmenujeme, aby měly.
- L a K jsou atomické formule.
- θ je nejobecnější unifikátor atomických formulí L a K.

Nezbytnost tzv. faktorizace

- Příklad: Následující množina klauzulí je sporná, ale nelze odvodit spor pouze rezolucí:
  - $p(X) \mid p(Y)$
  - $\neg p(X) \mid \neg p(Y)$
- Abychom odvodili spor, musíme nejprve (alespoň) jednu klauzuli **faktorizovat** na $p(X)$ nebo $\neg p(X)$.

Pravidlo faktorizace

- **Pravidlo faktorizace:**

$$\frac{\Delta \mid L \mid K}{\Delta\theta \mid L\theta}$$

kde:

  - *Δ* je disjunkce libovolného počtu literálů (klauzule).
  - *L* a *K* jsou literály, oba pozitivní nebo oba negativní.
  - *θ* je nejobecnější unifikátor literálů *L* a *K.*

Více resolvent ze dvou klauzulí

- V logice 1. řádu lze použít rezoluční pravidlo více způsoby na dvě klauzule a dostat různé výsledky.
- Příklad: Z klauzulí *p(c)* | *p(d)* a ¬*p(X)* | *q(X)* odvodíme:
  - *p(d)* | *q(c)* (přes literály *p(c)* a ¬*p(X)*)
  - *p(c)* | *q(d)* (přes literály *p(d)* a ¬*p(X)*)
- Obě odvozené klauzule jsou důležité, ani jedna z nich není silnější než druhá.

Korektnost rezolučního kalkulu: Pokud $A \vdash \varphi$ pak $A \models \varphi$.

- $\Delta \mid L \quad \sim K \mid \Gamma$
  $$\frac{\Delta \mid L \quad \sim K \mid \Gamma}{\Delta\theta \mid \Gamma\theta}$$
- Buď M interpretace taková, že $M \models (\Delta \mid L)$ a $M \models (\sim K \mid \Gamma)$.
- Pro libovolnou substituci, a tedy i MGU θ platí $M \models (\Delta\theta \mid L\theta)$ a $M \models (\sim K\theta \mid \Gamma\theta)$. Buď e ohodnocení v interpretaci M.
  - Pokud $M,e \models L\theta$, pak musí být $M,e \models \Gamma\theta$ a tedy $M,e \models (\Delta\theta \mid \Gamma\theta)$.
  - Pokud $M,e \models \sim L\theta$, pak musí být $M,e \models \Delta\theta$ a tedy $M,e \models (\Delta\theta \mid \Gamma\theta)$.
- Pro faktorizaci je korektnost zřejmá.

Úplnost rezolučního kalkulu logiky 1. řádu

- Věta: Pokud $\boldsymbol{A} \models \perp$ pak $\boldsymbol{A} \vdash \perp$.
  Neboli: rezolučním kalkulem lze vždy odvodit prázdnou klauzuli ze sporné množiny.
  (Angl. **„refutationally complete“**.)

- Důkaz: Převedením na výrokový případ pomocí Herbrandovy věty a Lifting lemmatu.

- Pozor! Stejně jako u výrokové rezoluce **neplatí** obecně, že pokud $\mathbf{A} \models \varphi$ pak $A \vdash \varphi$.

Úplnost rezolučního kalkulu logiky 1. řádu

- Je-li **A** splnitelná množina klauzulí, pak buďto:
  - Dospějeme při odvozování k množině, která je uzavřená na aplikaci rezoluce (**saturovaná množina**), a pak víme, že ***A*** je splnitelná.
  - Odvozování nikdy neskončí, klauzulí bude stále přibývat, a nedozvíme se nic (příklad).

Použití rezolučního kalkulu

1. Převedeme úlohu na hledání důkazu spornosti množiny formulí.
2. Tzv. **Skolemizací** odstraníme existenční kvantifikátory.
3. Formule převedeme na klauzule.
4. Aplikujeme rezoluční kalkulus.

Skolemizace – odstranění existenčních kvantifikátorů

- Formule

  $(\forall X_1)...(\forall X_n)(\exists Y)\varphi$

  je splnitelná právě když

  $(\forall X_1)...(\forall X_n)\varphi\{Y \mapsto f(X_1,..., X_n)\}$

  je splnitelná formule.
- *f* je nový symbol přidaný do jazyka, který není použit nikde jinde.
- Na této myšlence je založena **Skolemizace** – proces odstranění existenčních kvantifikátorů.

Skolemizace – odstranění existenčních kvantifikátorů

- Postup:

1) Převedeme formuli do **prenexního tvaru** – pomocí vhodných pravidel přesuneme všechny **kvantifikátory na začátek formule**.

– Je-li formule tvaru $(\forall X_1)...(\forall X_n)(\exists Y)\varphi$, přidáme do jazyka nový funkční symbol *f* arity *n* a formuli nahradíme novou formulí $(\forall X_1)...(\forall X_n)\varphi\{ Y \mapsto f(X_1,..., X_n)\}$

1) Bod (2) opakujeme dokud ve formuli jsou existenční kvantifikátory.

- Výsledná formule není ekvivalentní s původní! Pouze je ekvisplnitelná.

Pravidla pro práci s kvantifikátory

- $\neg(\forall X)\varphi(X) \equiv (\exists X)\neg\varphi(X)$
- $\psi \mid ((\forall X)\varphi(X)) \equiv (\forall X)(\psi \mid \varphi(X))$
- $\psi \,\&\, ((\forall X)\varphi(X)) \equiv (\forall X)(\psi \,\&\, \varphi(X))$

  $\psi \Rightarrow ((\forall X)\varphi(X)) \equiv (\forall X)(\psi \Rightarrow \varphi(X))$

  $((\forall X)\varphi) \Rightarrow \psi \equiv (\exists X)(\varphi(X) \Rightarrow \psi)$
- <u>Pozor:</u> Pro ⇔ a ⊕ (XOR) jednoduchá pravidla nejsou, je třeba převést na kombinaci výše uvedených spojek!
- Ve všech případech **nesmí být *X* volná** proměnná v *ψ* (příklad).

Převedení formulí na klauzule

- Po skolemizaci máme formuli v prenexním tvaru, bez existenčních kvantifikátorů.
- Všechny logické spojky přepíšeme pomocí konjunkce, disjunkce a negace.
- Pomocí DeMorganových pravidel přesuneme všechny negace až k výrokovým proměnným.
- Průběžně eliminujeme dvojité negace.
- Distributivním pravidlem roznásobíme konjunkce a disjunkce tak, aby všechny disjunkce byly uvnitř konjunkcí.