Matematická logika

přednáška jedenáctá

Miroslav Kolařík

Zpracováno dle textu R. Bělohlávka:
Matematická logika – poznámky k přednáškám, 2004.

Obsah

V Prologu je po zadání dotazu G cílem překladače zjistit, zda z programu P plyne dotaz G, tj. zda $P \models G$. Víme, že $P \models G$, právě když $P, \neg G$ je sporná. Dle poslední věty je zjištění spornosti $P, \neg G$ ekvivalentní zjištění zda $\square$ patří do nějaké množiny $R^n(F)$. Výpočet R^n je však obecně náročný a nijak nevyužívá speciální formu klauzulí používanou v Prologu (hornovské klauzule).

Naším následujícím cílem je ukázat speciální formu rezoluční metody, tzv. SLD-rezoluci.

Dále budeme používat následující označení:

P … definitní logický program (konečná množina hornovských klauzulí s jedním pozitivním literálem, které odpovídají faktům a pravidlům – tzv. programové klauzule)

G … cílová klauzule (hornovská klauzule tvaru $\neg G_1 \vee \cdots \vee \neg G_n$, která odpovídá negaci uživatelem zadaného cíle ?- $G_1, \ldots, G_n$)

Definice

Odpověď pro $P \cup \{G\}$ je libovolná substituce $\theta = (x_1/t_1, \ldots, x_m/t_m)$, kde každá proměnná x_i se vyskytuje v G.

Definice

Odpověď θ pro $P \cup \{G\}$ se nazývá **korektní**, jestliže formule $\forall((G_1 \wedge \cdots \wedge G_n)\theta)$ sémanticky plyne z P.

To je základem deklarativní sémantiky logického programu P, kterou je možné chápat jako množinu všech G spolu s θ, kde θ je korektní odpověď pro $P \cup \{G\}$.

To souvisí s odpověďmi Prologu takto: Prolog odpoví na dotaz ?- $G_1, \ldots, G_n$ buď „Yes“ se substitucí θ (je-li prázdná, pak ji Prolog nevypisuje) nebo „No“. Odpověď „Yes“ se substitucí θ je korektní, jestliže θ je korektní odpověď pro $P \cup \{G\}$ v právě definováném smyslu. Odpověď „No“ je korektní, jestliže $\forall((G_1 \wedge \cdots \wedge G_n)\theta)$ sémanticky neplyne z P pro žádnou θ.

Pokoušíme-li se aplikovat rezoluční pravidlo na klauzule, z nichž jedna odpovídá cíli (nemá žádný pozitivní literál) a druhá odpovídá pravidlu nebo faktu (má právě jeden pozitivní literál A), už se nepotýkáme s nedeterminismem implicitně obsaženým v obecném rezolučním pravidle. Z podstaty rezolučního pravidla je jasné, že v cíli hledáme takový G_i, který je možné unifikovat s A (tedy nevybíráme dvojice atomických formulí mající unifikaci jako v obecné rezoluční metodě, ale jen atomické formule, které jsou unifikovatelné s A).

Definice

SLD-odvození z $P \cup G$ je posloupnost $H_0 = G, H_1, \ldots$ cílových klauzulí, posloupnost $C_1, C_2, \ldots$ variant programových klauzulí z P (tj. C_i vzniknou z klauzulí P přejmenováním proměnných) a posloupnost $\theta_1, \theta_2, \ldots$ unifikací takových, že θ_{i+1} je mgu množiny $\{H_i, C_{i+1}\}$ a H_{i+1} je odpovídající rezolventou (tj. vznikne použitím rezolučního pravidla na H_i a C_{i+1} při θ_{i+1}).
SLD-odvození se nazývá **zamítnutí** (refutace) délky n, jestliže jako poslední klauzule je $H_n = \square$.

Poznámka: SLD-odvození odpovídá výpočtu prologovského překladače a je základem tzv. procedurální sémantiky. Souvislost s prologovským zásobníkem je téměř zřejmá – stav zásobníku odpovídá SLD-odvození.

Poznámka: Odvození může být

a) úspěšné (refutací)
b) neúspěšné
    b1) nekonečné
    b2) konečné, nekončící $\square$, které nelze dále prodloužit

Definice

Vypočítaná odpověď pro $P \cup \{G\}$ je substituce θ, pro kterou existuje refutace z $P \cup \{G\}$ délky n s odpovídající posloupností $\theta_1, \dots, \theta_n$ takovou, že θ vznikne z $\theta_1, \dots, \theta_n$ (složením substitucí) vynecháním těch x/t, pro které se x nevyskytuje v G.

Věta o korektnosti SLD-rezoluce

Každá vypočítaná odpověď pro $P \cup \{G\}$ je korektní odpovědí pro $P \cup \{G\}$.

Věta: úplnost SLD-rezoluce; odpověď „Yes“

Pro cíl ?- $G_1, \ldots, G_n$ a program P platí, že jestliže $\exists(G_1 \wedge \cdots \wedge G_n)$ sémanticky plyne z P, pak existuje refutace z $P \cup \{G\}$.

Věta: úplnost SLD-rezoluce; odpověď „Yes“ se substitucí

Pro každou korektní odpověď θ pro $P \cup \{G\}$ existuje vypočítaná odpověď σ pro $P \cup \{G\}$, která je (jako substituce) obecnější než θ (tj. $\theta = \sigma\tau$ pro nějakou substituci τ).

Tedy, jestliže pro cíl ?- $G_1, \ldots, G_n$ a program P platí, že $\exists(G_1 \wedge \cdots \wedge G_n)$ sémanticky plyne z P se substitucí θ, pak to lze prokázat syntakticky refutací, která v sobě má substituci obecnější než θ.

Sjednocující pohled

Uživatelský pohled na Prolog je založen na deklarativní sémantice (sémantické vyplývání).

Programátorský pohled na Prolog je založen na rezolučním zásobníku (tak je hledání odpovědi, tj. hledání refutace se substitucí, tj. hledání důkazu implementováno).

Logický pohled na Prolog je založen na rezoluční metodě, speciálně na SLD-rezoluci.

Výše uvedené výsledky ukazují souvislosti mezi jednotlivými pohledy.

Především: sémantické výplývání je přirozené z hlediska intuitivního pochopení, co prologovský překladač hledá a jaký je význam odpovědi. Nedává však návod, jak hledání implementovat. Toto hledání lze v principu realizovat hledáním důkazu v axiomatickém systému, který byl uveden v základním výkladu o PL. Takové hledání je však příliš náročné („příliš nedeterministický“ pojem důkazu v axiomatickém systému; tento pojem není primárně určen pro hledání důkazu). Proto byla navržena rezoluční metoda, která je ještě vhodnější v případě SLD-rezoluce (viz dříve popsaný efekt hornovských klauzulí: v cílové klauzuli jsou všechny literály negativní, jediným literálem, který je možné v programové klauzuli unifikovat je tedy jediný pozitivní literál).

Pozor: úplnost říká, že má-li být odpověď „Yes“, pak je tato odpověď vypočítatelná. To ale neznamená, že ji Prolog vždy najde; překladač prohledává podle algoritmu zásobníku a může se rozběhnout po nekonečné větvi. Prohledává totiž strom možných řešení do hloubky (přitom využívá očíslování formulí). To však není tím, že bychom zatím špatně hledali a příslušný algoritmus, který refutující výpočet vždy najde, ještě neobjevili. Je to jen jeden z projevů jednoho ze základních výsledků vyčíslitelnosti: PL je nerozhodnutelná (problém: „zjisti, zda je formule tautologií“ je nerozhodnutelný, stejně tak problém logického vyplývání). Tato situace je analogická známější situaci – musíme čelit NP-úplnosti některých problémů. Nejde se jim vyhnout, musíme použít heuristiky. Krása výsledků teoretické informatiky a logiky spočívá v tom, že tyto principiální věci nám sdělují (jinak bychom třeba marně hledali efektivní algoritmy, které neexistují). Z tohoto pohledu je tedy algoritmus prologovského překladače heuristikou čelící nerozhodnutelnosti problému, který nás zajímá („předpokládám-li znalosti v databázi, plyne z nich můj dotaz?“).

Poznámka: Všimněme si významu očíslování formulí v prologovském programu, které používá algoritmus prologovského zásobníku. Z pohledu vysvětlené SLD-rezoluce spočívá význam očíslování v tom, že odstraňuje nedeterminismus v kroku, ve kterém je třeba vybrat programovou klauzuli k unifikaci. Zatímco v SLD-rezoluci je připuštěna libovolná možná programová klauzule, algoritmus prologovského zásobníku říká, že se použije ta s nejmenším číslem.