Zastavování výpočtů
(Terminace)

Petr Štěpánek

S využitím materialu Krysztofa R. Apta

2007

Analýza zastavování (terminace) výpočtů je součastí verifikace programů v Prologu.

Není třeba zdůrazňovat, že problém zastavování výpočtů je nerozhodnutelný.

Můžeme tedy použít jen heuristické metody k důkazu zakončování některých programů v čistém Prologu, které jsou použitelné i pro některé definitní programy v Prologu.

V předchozí kapitole jsme se přesvědčili, že použití levého výběrového pravidla a prohledávání výsledných stromů do hloubky odlišuje čistý Prolog od Logických programů.

Proto nelze bezprostředně využít úplnost SLD-rezoluce, která spojuje deklarativní a procedurální interpretaci logických programů k analýze ukončování výpočtů prologovských programů.

I když Věta o úplnosti SLD-rezoluce zaručuje existencí řešení daného dotazu Q, systém implementace čistého Prologu se s tímto řešením může minout, pokud všechny úspěšné uzly leží napravo od nějaké nekonečné větve ve výsledném stromu čistého Prologu. V takovém případě říkáme, že dotaz Q diverguje.

Zastavování výpočtů je tedy v čistém Prologu nové samostatné téma.

Budeme presentovat metodu (metody) pro dokazování terminace programů logických a programů v čistém Prologu.

První, co každého napadne je ztotožnit terminaci programu P s konečností *všech možných* SLD-derivací pro počáteční dotaz Q (přesněji pro úlohu $P \cup \{Q\}$) vzhledem k levému výběrovému pravidlu.

Vzhledem k tak silnému požadavku terminuje jen málo jednoduchých programů. Při analýze terminace bude třeba množinu uvažovaných derivací různými způsoby omezit.

Definice. (první pokus)

Říkáme, že program P zastavuje, jestliže všechny jeho SLD-derivace začínající základním dotazem jsou konečné.

Tedy, SLD-stromy pro základní dotazy zastavujících programů jsou konečné a libovolná strategie prohledávání těchto stromů vždy skončí po konečném počtu kroků nezávisle na výběrovém pravidle.

Pokud jde o programy v (čistém) Prologu, zajímají nás důkazy zastavování výpočtů nejen pro všechny základní dotazy, ale i pro nějaké třídy zamýšlených dotazů, které nejsou základní.

Metody dokazování terminace programů, které uvedeme, dovolují takovou třídu dotazů, které nejsou nutně základní, popsat pro každý program.

Jak ukážeme, některé Prologovské programy například `SUM`, `LIST` a `APPEND` jsou zakončující podle uvedené definice.

K tomu budeme potřebovat určitou výbavu:

- multi-množiny a jejich fundované uspořádání,
- Koenigovo lemma o konečně se větvících stromech a
- stupňová zobrazení.

Multi-množiny a jejich uspořádání

Multi-množina (anglicky *multiset* nebo *bag*) je konečná neuspořádáná posloupnost. Multi-množinu sestávající z prvků $a_0, a_1, a_2, \ldots, a_n$, budeme značit $bag(a_0, a_1, a_2, \ldots, a_n)$.

Množiny lze představit stejným způsobem, ale multi-množina se od množiny se liší tím, že v multi-množině je opakování prvků dovoleno.

Množiny jsou tedy speciálním případem multi-množin.

Definice. (uspořádání multi-množin sestávajících z přirozených čísel)

Uvažujme třídu multi-množin přirozených čísel, (tedy posloupností přirozených čísel případně s opakováním).

Definujeme relaci $\ll$ mezi takovými multi-množinami jako tranzitivní uzávěr relace $<$, kde $X < Y$ jestliže X vznikne z Y nahrazením nějakého prvku a z Y konečnou (případně prázdnou) multi-množinou přirozených čísel Z takovou, že každý prvek Z je menší než a.

Symbolicky, relaci $<$ definujeme následovně

$X < Y$ právě když $X = (Y - \{a\}) \cup Z$ pro nějaké $a \in Y$ a Z takové, že $b < a$ pro všechny prvky $b \in Z$.

Relace $\ll$ je pak uspořádání na třídě multi-množin.

Označíme a jako $old(X,Y)$ a Z jako $new(X,Y)$.

Příklad.

Mějme dvě multi-množiny

$X = bag(3,3,6,7)$ a $Y = bag(2,2,2,3,7,5)$.

Platí $Y \ll X$

protože nahrazením jednoho výskytu trojky v X třemi výskyty dvojky dostaneme

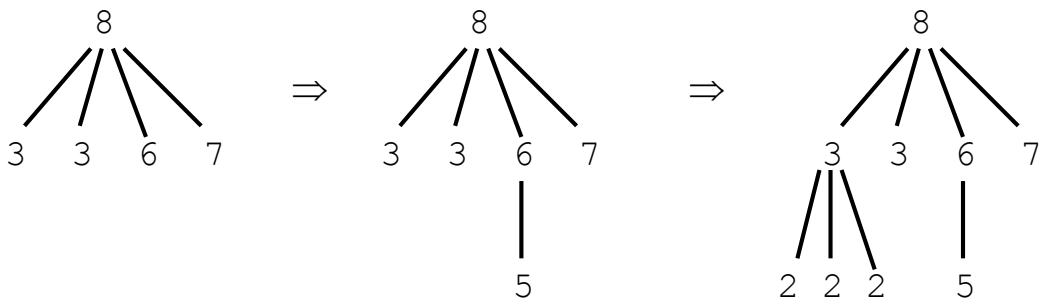
$bag(2,2,2,3,7,5) < bag(3,3,7,5)$

a nahrazením šestky v X pětkou dostáváme

$bag(3,3,7,5) < bag(3,3,6,7)$

tedy $X \ll Y$.

Můžeme to znázornit obrázkem



K důkazu dalších vlastností uspořádání « použijeme následující lemma.

Lemma 1 . (Koenig 1926)

Každý nekonečný a konečně se větvící strom má nekonečnou větev.

Důkaz. Nechť T je nekonečný a konečně se větvíví strom. Nekonečnou větev $\nu = n_0, n_1, n_2, \ldots$ sestrojíme indukcí.

Báze indukce, $i = 0$ pak kořen stromu T vybereme jako první člen n_0 posloupnosti v.

Indukční krok, předpokládejme, že uzly $n_0, n_1, n_2, \ldots, n_i$ pro $i \geq 0$, již byly sestrojeny a n_i je kořenem nekonečného podstromu stromu T. Protože T se konečně větví, uzel n_i má jen konečně mnoho bezprostředních následníků.

Alespoň jeden z nich je kořenem nekonečného podstromu T, a ten vybereme jako další člen n_{i+1} posloupnosti v .

Tímto postupem nakonec sestrojíme nekonečnou větev stromu T.

Definice. (Fundované uspořádání)
Říkáme, že relace uspořádání $<$ je fundovaná, jestliže neobsahuje nekonečnou klesající posloupnost.

Snadno se nahlédne, tranzitivní uzávěr « relace $<$ je fundovaný, právě když sama relace $<$ je fundovaná.

Věta 1. (fundovanost uspořádání multi-množin)
Relace uspořádánání « na třídě multi-množin přirozených čísel je fundovaná.

Důkaz. Podle poznámky za definicí stačí ukázat, že je fundovaná relace $<$.

Předpokládejme, že

$$\xi = m_0, m_1, m_2, \ldots, m_k, \ldots \quad (1)$$

je nekonečná klesající posloupnost multimnožin taková, že pro každé i

$$m_{i+1} < m_i .$$

Indukcí sestrojíme konečně se větvící strom T jehož uzly jsou přirozená čísla, takový, že na každé hladině i jsou prvky multi-množiny m_i listy stromu dosud vytvořeného a každý uzel T je větší než jeho děti.

Jako kořen stromu zvolíme přirozené číslo n větší než všechny prvky m_0 .

Báze indukce, $i = 0$. Strom T (sestávající zatím jen z kořene n) rozšíříme tak, že ke kořeni přidáme všechny prvky multi-množiny m_0 jako jeho děti.

Indukční krok, rozšiřme strom T přidáním všech prvků z multi-množiny $new(m_{i+1}, m_i)$ jako následníky uzlu $old(m_{i+1}, m_i)$. V případě, že $new(m_{i+1}, m_i)$, nedělej nic.

Protože posloupnost (1) je podle předpokladu nekonečná, potom multi-množina $new(m_{i+1}, m_i)$ byla nekonečněkrát neprázdná, protože jinak by v nějakém kroku musela mohutnost m_i ostře poklesnout.

Tedy ke stromu T byly nekonečněkrát přidány nové prvky, to znamená, že T je nekonečný. Z konstrukce v indukčním kroku je zřejmé, že T se konečně větví.

Podle Koenigova lemmatu má T nekonečnou větev, sestávající z klesající posloupnosti přirozených čísel.

Spor, relace « je fundovaná.

Definice. (Stupňová zobrazení)

(*i*) *Stupňové zobrazení* pro program P je funkce $|\,.\,| : \mathrm{HB}_P \to N$, které základním atomům jazyka L_P přiřazuje přirozená čísla.

Je-li $A \in \mathrm{HB}_P$, označujeme $|A|$ *stupeň* A.

(*ii*) Klauzuli z programu P nazveme *rekurentní vzhledem ke stupňovému zobrazení* $|\,.\,|$, jestliže pro každou její základní instanci

$$A \leftarrow \mathbf{A}, B, \mathbf{C}$$

platí $|A| > |B|$

(*iii*) Program P nazýváme *rekurentní vzhledem ke stupňovému zobrazení* $|\,.\,|$, jestliže jsou rekurentní všechny jeho klauzule.
P nazveme *rekurentní*, je-li rekurentní vzhledem k nějakému stupňovému zobrazeni.

Nejprve zobecníme pojem stupně atomu i pro atomy, které nejsou základní.

Definice. (Atomy omezené vzhledem k $|\,.\,|$, omezené dotazy)

(i) Říkáme, že (obecný) atom A *je omezený vzhledem ke stupňovému zobrazení* $|\,.\,|$, jestliže pro nějaké přirozené k, a každou jeho základní instanci A' platí $|A'| \leq k$.
Pro atom A omezený vzhledem k $|\,.\,|$, definujeme *stupeň* A vzhledem k $|\,.\,|$ jako maximum všech stupňů základních instancí $|A'|$.

(ii) Řikáme, že dotaz $Q \equiv A_1, A_2, \ldots, A_n$, *je omezený vzhledem k* $|\,.\,|$ jsou-li omezené všechny jeho atomy.
Stupeň Q definujeme jako multi-množinu $bag(|A_1|, |A_2|, \ldots, |A_n|)$.
Platí-li $|A_i| \leq k$ pro nějaké k a všechna $i \leq n$, říkáme, že Q je *omezený číslem* k.

Povšimněme si, že pro atomický dotaz A, má nyní $|A|$ dvojí význam. Záleží na tom, díváme-li se na A jako na atom nebo jako dotaz.

Nyní můžeme vyslovit postačující podmínku pro zastavování rekurentních programů. Nejprve dokážeme hlavní lemma.

Lemma. (Omezenost 1) Nechť P je program rekurentní vzhledem k nějakému stupňovému zobrazení $|.|$. Nechť Q_1 je dotaz omezený vzhledem k $|.|$ a Q_2 je SLD-rezolventa Q_1 je a klauzule z P.
Potom platí

(*i*) Q_2 je omezený vzhledem k $|.|$ a

(*ii*) $|Q_2| \ll |Q_1|$.

Důkaz je důsledkem tří jednoduchých pozorování.

Pozorování 1. Libovolná instance Q' omezeného dotazu Q je omezená a platí $|Q'| \ll |Q|$.

K důkazu Pozorování 1 stačí nahlédnout, že instance A' omezeného atomu A je také omezená a platí $|A'| < |A|$.

Pozorování 2. Instance c' rekurentní klauzule c je rekurentní.

Stačí si uvědomit, že každá instance klauzule c' je také instancí klauzule c.

Pozorování 3. Pro každou rekurentní klauzuli $H \leftarrow \mathbf{B}$ a libovolné posloupnosti atomů **A** a **C** platí:

Je-li **A**, H, **C** omezená, potom také **A**, **B**, **C** je omezená a platí

$$|\mathbf{A}, \mathbf{B}, \mathbf{C}| \ll |\mathbf{A}, H, \mathbf{C}|.$$

Důkaz Pozorování 3. Nejprve provedeme důkaz za předpokladu, že **A** a **C** jsou prázdné.

Nechť C je libovolný atom v nějaké základní instanci $\mathbf{B}$.

Potom C se vyskytuje v těle nějaké základní instance klauzule

$$H \leftarrow \mathbf{B},$$

například $H\theta \leftarrow \mathbf{B}\theta$ pro nějakou substituci θ.

Podle Pozorování 2 platí $|C| < |H\theta|$, tedy také $|C| < |H|$.

Odtud plyne, že $\mathbf{B}$ je omezená a

$$|\mathbf{B}| \ll |H|.$$

Obecný případ plyne z fundovanosti uspořádání $\ll$.

Důsledek 1. (Zastavování)

(*i*) Nechť P je rekurentní program a Q omezený dotaz. Potom všechny SLD-derivace pro $P \cup \{Q\}$ jsou konečné.

(*ii*) Každý rekurentní program terminuje (zastavuje).

Důkaz. (i) z lemmatu o omezenosti a fundovanosti uspořádání multimnožin.
(*ii*) každý základní dotaz je omezený.

Stupňová zobrazení byla definována pro základní atomy. Je přirozené je rozšířit i na zakladní termy, které se vyskytují v takových atomech.

Příklady, které chceme uvést, budou o seznamech. Indukcí zavedeme stupňové zobrazení pro všechny základní termy, zatím tak, že pro seznamy budou hodnoty určovat délku seznamu. Pro ostatní základní termy budou hodnoty triviální.

Funkce `listsize`

Indukcí definujeme funkci $|\,.\,| : HU_P \rightarrow N$ pro nějaký program P, který zobrazuje Herbrandovo univerzum, tedy množinu základních termů jazyka L_P, do množiny přirozených čísel.

Definice.

(*i*) $|[x|xs]| = |xs| + 1,$

$|f(t_1, t_2, \ldots, t_n)| = 0$ pokud $f \neq [\,.\,|\,.\,],$

hodnoty $|t|$ pro základní termy t nazýváme také `listsize(`t`)`.

Funkce listsize není stupňová ve smyslu původní definice.

(*ii*) Nechť $|\,.\,|$ je stupňové zobrazení. Říkáme, že atom A je ztrnulý vzhledem k $|\,.\,|$ jestliže tato stupňová funkce je konstantní na *ground*(A).

Je zřejmé, že ztrnulé atomy jsou omezené.

Příklad. (Program `LIST`)

```
% list(Xs)← Xs je seznam.

list([]).
list([_|Ts]) ←  list(Ts).
```

Položíme-li

`|list(t)| = |t|,`

je zřejmé, že program `LIST` je rekurentní vzhledem ke stupňovému zobrazení $|\,.\,|$ a zobrazení `listsize` a že pro seznam `t`, je atom

`list(t)`

ztrnulý vzhledem k uvedeným zobrazením.

Tedy podle lemmatu o omezenosti a jeho důsledků, program `LIST` zastavuje a všechny SLD-derivace pro `LIST` ∪ {`list(t)`} jsou konečné.

Aplikace

Pojem zastavování podle první definice je velmi silný. Ukážeme, že jen málo jednoduchých programů zastavuje podle této definice.

MEMBER

```
% member(Element,List) ← Element je prvkem List.
member(X,[X|_]).
member(X,[_|Xs]) ← member(X,Xs).
```

Použijeme-li stupňové zobrazení

$$|member(x,y)| = |y|$$

(a v něm obsažené zobrazení `listsize`), program `MEMBER` je rekurentní, tedy zastavuje a všechny SLD-derivace pro

$$MEMBER \cup \{ member(s,t) \}$$

jsou konečné.

SUBSET

```
% subset(Xs,Ys) ←  každý prvek Xs patří do Ys.
subset([],_).
subset([X|Xs],Ys) ← member(X,Ys),subset(Xs,Ys).
```

\+ program `MEMBER`

Definujeme dvě stupňová zobrazení:

$$|member(x,xs)| = |xs|,$$
$$|subset(xs,ys)| = |xs| + |ys|.$$

abychom ukázali, že program SUBSET je rekurentní.

Potom program `SUBSET` zakončuje a všechny SLD-derivace pro

$$SUBSET \cup \{subset(xs,ys)\},$$

kde `xs` a `ys` jsou základní seznamy, jsou konečné. Vzhledem k nezávislosti na výběrovém pravidle, pro dotaz `{subset(xs,ys)}` můžeme v těle druhé klauzule prohodit pořadí atomů.

V obecném případě můžeme mít na výběr více stupňových zobrazení a pro různé volby můžeme odvodit různé důsledky. Ilustrujeme to následujícími třemi jednoduchými příklady.

APPEND

Různá stupňová zobrazení mohou vést na různé třídy omezených dotazů.

```
% app(Xs,Ys,Zs) ← Zs vznikne spojením Xs a Ys.
app([],Ys,Ys).
app([X|Xs],Ys,[X|Zs]) ← app(Xs,Ys,Zs).
```

|Snadno se ověří, že APPEND je rekurentní vzhledem k následujícím dvěma stupňovým zobrazením:

```
app(xs,ys,zs)| = |xs|
|app(xs,ys,zs)| = |zs|
```

pro každé z nich dostaneme jinou třídu omezených dotazů.

Stupňové zobrazení:

```
|app(xs,ys,zs)| = min(|xs|,|zs|)
```

spojuje výhody obou předchozích zobrazení. Snadno se podle něj ověří, že APPEND je rekurentní program. Naxíc, je-li xs nebo zs seznam, app(xs,ys,zs) je omezený dotaz (i když ne ztrnulý).

Proto z lemmatu o omezenosti a jeho důsledků plyne, že APPEND se zastavuje a je-li jeden z termů xs,ys seznam, potom všechny SLD-derivace pro APPEND ∪ {app(xs,ys,zs)} jsou konečné.

SELECT

```
% select(X,Xs,Zs)← Zs vznikne z Xs vynecháním jednoho
                    výskytu položky X.
select(X,[X|Xs],Xs).
select(X,[Y|Xs],[Y,Zs])← select(X,Xs,Zs).
```

Stejně jako u programu `APPEND` je nejvýhodnější použít stupňové zobrazení

$$|\texttt{select(x,xs,zs)}| = \min(|\texttt{ys}|,|\texttt{zs}|) \qquad (1)$$

Potom `SELECT` je rekurentní vzhledem k (1) a v případě, že jeden z termů `xs,zs` je seznam, všechny SLD-derivace pro

$$\texttt{SELECT} \cup \{\texttt{select(x,ys,zs)}\}$$

jsou konečné.

V některých případech se musíme uchýlit k poněkud bizarním stupňovým zobrazením, abychom zachovali striktní podmínku rekurentnosti. Příkladem takového programu je `PALINDROM`.

`PALINDROM`

`% palindrom(Xs)` ← seznam `Xs` je roven svému zrcadlovému obrazu. `palindrom (Xs)` ← `reverse(Xs,Xs).`

`% reverse(Xs,Ys)`←`Ys` vznikne z `Xs` obrácením pořadí položek.
`reverse(X1s,X2s)`← `reverse(X1s,[],X2s).`

`% reverse(Xs,Ys,Zs)`← `Zs` vznikne spojením obráceného seznamu `Xs` se seznamem `Ys`. `reverse([],Xs,Xs).`
`reverse([X|X1s],X2s,Ys)`← `reverse(X1s,[X|X2s],Ys).`

`PALINDROM` je rekurentní vzhledem k následujícímu zobrazení:

```
    |palindrom(xs)| = 2.|xs| + 2
   |reverse(xs,ys)| = 2.|xs| + 1
|reverse(xs,ys,zs)| = 2.|xs| + |ys|
```

Dá se ukázat, že platí i opačná implikace ke tvrzení (*ii*) Důsledku 1. Následující tvrzení uvádíme bez důkazu.

Věta 2. Program zastavuje, právě když je rekurentní.

• Z této věty vyplývá, že rekurentní programy a omezené dotazy jsou příliš omezující, pro analýzu zastavování větších tříd definitních programů a dotazů.

• Nevyužívají faktu, že (čistý) Prolog používá pevné výběrové pravidlo, pravidlo nejlevějšího atomu.

• Připomeňme, že SLD-derivace používající toto levé výběrové pravidlo jsme označovali jako *LD-derivace* .

Při studiu zastavování programů v (čistém) Prologu je třeba studovat vlastnosti LD-derivací.

Motivační příklady.

(i) Předvedeme program P , který zastavuje a takový, že pro jistý dotaz Q jsou všechny LD-derivace pro $P \cup \{Q\}$ konečné, zatímco některé SLD-derivace jsou nekonečné.

APPEND3

```
% app3(Xs,Ys,Zs,Us)  ← Us  je výsledkem spojení seznamů
                          Xs,Ys  a  Zs.
app3(Xs,Ys,Zs,Us)  ← app(Xs,Ys,Vs),  app(Vs,Zs,Us).
```

\+ APPEND

APPEND3 je rekurentní vzhledem k následujícímu stupňovému zobrazení |.|.

```
|app(xs,ys,zs)|  =  min(|xs|,|zs|),

|app3(xs,ys,zs,us)|  =   |xs|+|us|+1  .
```

Tedy podle Věty o zastavování APPEND3 terminuje. Ale typické použití tohoto programu je dotaz `app3(xs,zs,xs,Us)` , kde `xs,ys,zs` jsou seznamy a `Us` je proměnná.

Později dokážeme, že všechny LD-derivace pro

`app3(xs,zs,xs,Us)` (1)

jsou konečné.

To však neplatí pro všechny SLD-derivace. Při použití výběrového pravidla nejpravějšího atomu dostaneme nekonečnou SLD-derivaci. Jako důsledek lemmatu o omezenosti pak vyplývá, že dotaz (1) není omezený, ačkoliv může být vyhodnocen konečným výpočtem v Prologu.

(*ii*) Dalším příkladem je program Naive REVERS , který neterminuje ve smyslu původní definice, ale všechny jeho LD-derivace začínající základním dotazem jsou konečné.

Definice. (Programy zleva zastavující).

Říkáme, že program zleva zastavuje (terminuje), všechny jeho LD-derivace se základním dotazem jsou konečné.

K důkazu, že nějaký program zleva terminuje a k charakterizování dotazů, zavedeme pojmy akceptovatelné klauzule, akceptovatelného programu a omezeného dotazu.

Definice. (Akceptovatelný program)

Je-li dán program P , stupňové zobrazení $|\,.\,|$ a interpretace I , jazyka L_P ,

(*i*) říkáme, že klauzule c z P je *akceptovatelná vzhledem k* $|\,.\,|$ *a interpretaci* I , která je modelem P , jestliže pro libovolnou základní instanci

A <- **A**, B, **B** klauzule c takové, že $I \models$ **A ,** platí

$$|A| > |B|\,.$$

Jinými slovy,

pro každou základní instanci

$$A \text{ <- } B_1 , B_2 , \ldots , B_n \qquad (2)$$

klauzule c platí

$$| A | > | B_j | \text{ pro } 1 \leq j \leq i$$

kde $i \leq n$ je nejmenší index, takový, že

$$I \models B_1 , B_2 , \ldots , B_{i-1} \text{ a } I \not\models B_i \ .$$

(*ii*) Program P je akceptovatelný vzhledem k $|\ .\ |$ a I jsou-li všechny jeho klauzule akceptovatelné vzhledem k $|\ .\ |$ a I .

Program P je akceptovatelný, je-li akceptovatelný vzhledem k nějakému stupňovému zobrazení nějaké interpretaci I .

Předpokladem $I \models \mathbf{A}$ se akceptovatelnost klauzule c vzhledem k $|\ .\ |$ a I liší od pojmu rekurence.

Je-li tento předpokad splněn, potom v každé základní instanci

$\mathbf{A}, B, \mathbf{B}$ klauzule c ,

je-li pravidlem LD-rezoluce dosažen atom B , atomy z $\mathbf{A}$ již byly rezolvovány.

Podle Věty o korektnosti SLD-rezoluce jsou všechny atomy z $\mathbf{A}$ pravdivé v I .

Pro každou základní instanci (2) klauzule c požadujeme, aby stupeň $| A |$ byl větší než stupeň každého atomu v nějakém počátečním úseku těla instance (2). Které atomy B_i bereme do úvahy je určeno modelem I .

Poznámka. Program P je rekurentní vzhledm k $|\,.\,|$ (ve smyslu původní definice) , je-li akceptovatelný vzhledem k $|\,.\,|$ a maximálnímu modelu HB_L .

Pro definici omezenosti dotazu potřebujeme definovat funkci maxima každé množiny přirozených čísel S . Je-li S konečná $max\, S$ je počet jejích prvků a je-li S nekonečná, klademe $max\, S = \omega$.

Definice. (Omezený dotaz)

Je-li dán program P stupňové zobrazení $|\,.\,|$ pro P, interpretace I a přirozené číslo k ,

(i) dotaz Q je omezený číslem k , vzhledem k $|\,.\,|$ a I jestliže pro každou jeho základní instanci **A**, B, **B** takovou, že $I \models$ **A** platí

$$|B| \leq k .$$

Dotaz Q je omezený vzhledem k $|\,.\,|$ a I , je-li omezený pro nějaké k vzhledem k $|\,.\,|$ a I .

(ii) Pro každý dotaz Q sestávající z n atomů definujeme n množin přirozených čísel definovaných následujícím způsobem:

$|Q|_i^I = \{|A_i| \mid A_1, A_2, \ldots, A_n$ je základní instance Q a $I \models A_1, \ldots, A_{i-1}\}$

(iii) ke každému dotazu Q z n atomů definujeme multi-množinu $|Q|_I$ přirozených čísel:

$$|Q|_I = bag(\, max\, |Q|_1^I, \ldots, max\, |Q|_n^I)$$

Množina $|Q|_i^I$ se sestrojí tak, že uvažujeme množinu všech základních instancí dotazu Q takových, že prvních i- 1 atomů je pravdivých v I . Množina $|Q|_i^I$ sestává z i-tých atomů v těchto instancích.

Lemma. (Omezenost 2)

Nechť P je program akceptovatelný vzhledem k nějakému stupňovému zobrazení $|.|$ nějaké interpretaci I. Nechť Q_1 je dotaz omezený k $|.|$ a I. Nechť Q_2 je rezolventa Q_1 a nějaké klauzule z P. Potom platí

(*i*) Q_2 je omezená vzhledem k $|.|$ a I,

(*ii*) $|Q_2|_I \ll |Q_2|_I$.

Důkaz. LD-rezolventa dotazu a klauzule se sestrojí pomocí následujících tří operací:

- instancí dotazu,
- instancí klauzule,
- nahrazením prvního atomu H z dotazu tělem klauzule, jejíž hlava je H.

Lemma se dokáže pomocí pozorování, ke každé z uvedených tří operací.

Pozorování 1. Instance Q' omezeného dotazu je omezená a $|Q'|_I \underline{\ll} |Q|_I$.

Tvrzení plyne z faktu že $|Q'|_i^I \underline{\ll} |Q|_i^I$ platí pro každé i.

Pozorování 2. Instance akceptovatelné klauzule je akceptovatelná.
Důkaz: bezprostředně z definice akceptovatelnosti.

Pozorování 3. Pro každou akceptovatelnou klauzuli A <- **B** a posloupnost atomů **C**, je-li A, **C** omezený dotaz potom **B, C** je také omezený a platí $|\mathbf{B}, \mathbf{C}|_I \ll |A, \mathbf{C}|_I$

.

Důkaz. Nechť $\mathbf{B} = B_1, B_2, \ldots, B_n$ a $\mathbf{C} = C_1, C_2, \ldots, C_m$ pro $n, m \geq 0$.
Tvrzení vyplývá z následujících dvou faktů.

Fakt 1. Pro každé i , $1 \leq i \leq n$ je $|B_1, B_2, \ldots, B_n, C_1, C_2, \ldots, C_m|_i^I$ konečná množina a platí

$max\ |B_1, B_2, \ldots, B_n, C_1, C_2, \ldots, C_m|_i^I < max\ |A, C_1, C_2, \ldots, C_m|_1^I$

Platí

$max\ |B_1, B_2, \ldots, B_n, C_1, C_2, \ldots, C_m|_i^I$ (definice omezeného dotazu)

$=$ $max\ \{|B_i'|, B_1'\ B_2', \ldots, B_n'$ je základní instancí **B** a $I \models B_1', \ldots, B_{i-1}'\}$

(pro nějaké A', A' <- $B_1'\ B_2', \ldots, B_n'$ je základní instancí A <- **B** a $I \models B_1', \ldots, B_{i-1}'$)

$<$ $max\ \{|A'|\ A'$ je základní instancí $\mathbf{A}\}$

(definice akceptovatelné klazule, takže každý prvek předchozí množiny je majorizován prvkem této množiny)

$=$ $max\ |A, C_1, C_2, \ldots, C_m|_{j+1}^I$ (definice omezeného dotazu)

Fakt 2. Pro každé j , $1 \leq j \leq m$ je

$|B_1, B_2, \ldots, B_n, C_1, C_2, \ldots, C_m|_{j+1}^I$ je konečná množina a

$max\ |B_1, B_2, \ldots, B_n, C_1, C_2, \ldots, C_m|_{j+1}^I \leq max\ |A, C_1, C_2, \ldots, C_m|_{j+1}^I$

Důkaz, Platí

$max\ |B_1, B_2, \ldots, B_n, C_1, C_2, \ldots, C_m|_{j+n}^I$ (definice omezeného dotazu)

$=$ $max\{|C_j'|\ |B_1', B_2', \ldots, B', C_1', \ldots, C_m'$ je základní instance **B, C** a $I \models B_1', B_2', \ldots, B_n'\ C_1', C_2', \ldots, C_{j-1}'\}$

(pro nějaké A', A' <- $B_1', B_2', \ldots, B_n'$ je základní instance A <- **B** , I je model P a $S \subseteq R$ implikuje $maxS \leq maxR$)

$\leq$ $max\{|C_j'|\ |\ A', C_1', \ldots, C_{j-1}'$ je základní instancí A, **C** a $\mathbf{I} \models A', C_1', \ldots, C_{j-1}'\}$

$= max \mid A, C_1, C_2, \ldots, C_m|_{j+1}^{I}$ (definice omezeného dotazu)

Z Faktů 1 a 2 dostáváme omezenost dotazu **B**,**C** a

$bag(max|\mathbf{B}, \mathbf{C}|_1^I, \ldots, max|\mathbf{B}, \mathbf{C}|_{n+m}^I \ll bag(max|A, \mathbf{C}|_1^I, \ldots, max|A, \mathbf{C}|_{m+1}^I)$

a to implikuje tvrzení Pozorování 3.

Dostáváme podobné důsledky jako pro rekurentní programy.

Důsledek. (Konečnost 2) Pro akceptovatelný program P a omezený dotaz Q jsou všechny LD-derivace pro $P \cup \{Q\}$ konečné.

Důsledek. (Terminace 2) Každý akceptovatelný program je zleva terminující.

Důkaz. Každý základní dotaz je omezený.

Příklady.

(a) Uvažujme program APPEND3 spolu s funkcí *listsize*. Položíme-li

`|app(xs,ys,zs)| = |xs|`,

`|app3(xs,ys,zs,us)| = |xs| + |ys| + 1`

a užijeme Herbrandovu interpretaci

I = `{app(xs,ys,zs) | |xs|+|ys|= |zs|}`

$\cup$ *ground* `(app3(Xs,Ys,Zs,Us)}`

pak I je model APPEND3: protože platí

`|[]|+|ys|= |ys|` a

`|xs|+|ys|=|zs|` implikuje `|[x|xs]|+|ys|= 1+|zs|= =|[x|zs]|`

Navíc klauzule definující predikát `app3` je triviálně splněna v I.

Víme, že APPEND je rekurentní vzhledem k |.|. Abychom ověřili, že program APPEND3 je akceptovatelný stačí si uvědomit, že

|xs|+|ys|+1 > |xs| a že
|xs|+|ys|= |vs| implikuje |xs|+|ys|+1 > |vs| .

Z důsledku Terminace 2 pak dostáváme, že APPEND3 je zleva terminující.

Navíc pro každé seznamy `xs, ys, zs` a libovolný term `u`, je dotaz `app3(xs,ys,zs,u)` omezený vzhledem k |.| a *I* takže podle důsledku Konečnost 2 jsou všechny LD-derivace pro

APPEND3 ∪ {app3(xs,ys,zs,u)}

konečné.

(b) Program PERMUTACE

```
% perm(Xs,Ys) <- Ys je permutací seznamu Xs .
perm([],[]).
perm(Xs, [X|Ys]) <-
                  app(X1s,[X|X2s],Xs),
                  app(X1s,X2s,Zs),
                  perm(Zs,Ys).
```

+ APPEND

• PERMUTACE není rekurentní: stačí vzít základní `xs,x,ys` a SLD-derivaci pro PERMUTACE ∪ {perm(xs,[x|ys]} sestrojenou následujícím způsobem: ve druhém kroku vyber prostřední atom `app(x1s,X2s,zs)` a potom v každém kroku používej rekurzivní klauzuli programu APPEND. Tak vznikne nekonečná SLD-derivace.

• Tedy PERMUTACE není terminující program a podle důsledku1 Zastavování 1 nemůže být rekurentní.

Víme, že

I_{APP} = {app(xs,ys,zs) | |xs|+|ys|= |zs|}

je model programu APPEND.

Program PERMUTACE je akceptovatelný vzhledem ke stupňovému zobrazení |.| a interpretaci I_{PERM} definovaným následovně:

|perm(xs,ys)| = |xs|+1,

|app(xs,ys,zs)| = min(|xs|,|zs|),

I_{PERM} = *ground*(perm(Xs,Ys)) $\cup$ I_{APP}.

Již jsme ukázali, že program APPEND je rekurentní vzhledem k |.|. Pro nerekursivní klauzuli programu PERMUTACE je akceptovatelnost zřejmá, pro rekurzivní klauzuli ji ověříme jednoduchým výpočtem.


```
perm(xs,ys) <-                                  {|xs|+1}

app(x1s,[x|x2s],xs),                    {min(|x1s|,|xs|)}
     {|x1s|+1+|x2s|=|xs|}

app(x1s,x2s,zs),                        {min(|x1s|,|zs|)}
     {|x1s|+|x2s|= |zs|}

perm(zs,ys).                                    {|zs|+1}
```

Ukázali jsme, že program PERMUTACE je akceptovatelný, podle důsledku 2 (Terminace 2) terminuje. Navíc pro libovolný seznam s a term t je atom perm(s,t) ztrnulý a tedy omezený. Podle důsledku Konečnost 2, to znamená, že všechny LD-derivace pro PERMUTACE {perm(s,t)} jsou konečné.

Volba stupňového zobrazení a modelu programu může ovlivňovat množinu dotazů, jejichž zastavování, lze dokázat.

Uvažujme znovu program SEKVENCE.

```
% sequence(Xs) ← Xs je seznam 27 položek.
sequence([_,...27krát...],Ss).

question(Ss) ← sublist([1,_,1,_,1],Ss),
               sublist([2,_,_,2,_,_,2],Ss),
               sublist([3,_,_,_,3,_,_,_,3],Ss),
                  ...
               sublist([9,_,9krát,9,_,9krát,_,9]).
```

+ SUBLIST

Snadno se ověří, že SEKVENCE je rekurentní program, tedy je akceptovatelný pro model HB_L vzhledem ke stupňovému zobrazení.

$|question(xs)| = |xs|+23$

$|sequence(xs)| = 0$

$|sublist(xs,ys)| = |xs|+|ys|+1$

$|app(xs,ys,zs)| = \min(|xs|,|zs|)$

Podle důsledku 1 Konečnost 1 , pro všechny základní termy `s` jsou všechny SLD-derivace (tedy i LD-derivace) pro SEKVENCE $\cup$ {`question(s)`} konečné.

Nicméně s touto volbou stupňového zobrazení není obecnější dotaz `question(Ss)` omezený. Proto nemůžeme použít důsledek 2 Konečnost 2 k důkazu terminace tohoto dotazu vzhledem k levému výběrovému pravidlu.

Tato situace je podobná terminaci dotazu programu APPEND3.

Abychom dokázali silnější podmínku terminace, změníme stupňové zobrazení a model programu tak, že

$$|\texttt{question(xs)}| = 50,$$

a zvolíme kterýkoli model I programu SEKVENCE, pro který pro libovolný základní term `s` platí

$I \models$ `sequence(s)` právě když `s` je seznam o 27 položkách.

Potom program SEKVENCE je akceptovatelný vzhledem k $|.|$ a I.

Podle důsledku 2 Konečnost 2 jsou všechny LD-derivace pro

$$\text{SEKVENCE} \cup \{\texttt{question(Ss)}\}$$

konečné.

Vztah zleva ukončujících a akceptovatelných programů. Dá se dokázat

Věta.

Program je zleva zakončující, právě když je akceptovatelný.