Logické programování

Principy

Šárka Vavrečková

Ústav informatiky, Filozoficko-přírodovědecká fakulta Slezské univerzity v Opavě

sarka.vavreckova@fpf.slu.cz

7. listopadu 2008

Definice

Nechť M je množina klauzulí klauzulární logiky. Označíme $\mathcal{R}(M)$ množinu klauzulí, pro kterou platí:

- $M \subseteq \mathcal{R}(M)$,
- *Jestliže klauzule C vznikne uplatněním rezolučního odvozovacího pravidla na klauzule C_i a C_j, kde $C_i \in M,\ C_j \in M$, pak $C \in \mathcal{R}(M)$ (na každou unifikovatelnou dvojici klauzulí z M uplatníme rezoluční pravidlo a výslednou rezolventu zařadíme do $\mathcal{R}(M)$).*

Rezoluční uzávěr množiny klauzulí M n-tého stupně je množina klauzulí $\mathcal{R}_n(M)$ definovaná rekurzívně:

$$\begin{aligned} \mathcal{R}_0(M) &= M, \\ \mathcal{R}_i(M) &= \mathcal{R}\left(\mathcal{R}_{i-1}(M)\right),\ 1 \leq i \leq n \end{aligned}$$

Věta (Robinsonův rezoluční princip)

Množina klauzulí klauzulární logiky M je nesplnitelná, jestliže existuje přirozené číslo n takové, že $\mathcal{R}_n(M)$ obsahuje prázdnou klauzuli $\rightarrow$.

Důkaz

- Převedeme množinu klauzulí $\mathcal{M}$ do predikátové logiky, mezi klauzulemi je vztah konjunkce:
  $C_1 \,\&\, C_2 \,\&\, \ldots \,\&\, C_k$,
- totéž provedeme s množinami $\mathcal{R}_i(\mathcal{M})$, $1 \leq i \leq n$,
- prázdnou klauzuli $\rightarrow$ převedeme na $true \rightarrow false$,
- rezoluční pravidlo zachovává splnitelnost,
- důkaz provedeme matematickou indukcí „zpětně“ od $\mathcal{R}_n(\mathcal{M}) \ni (true \rightarrow false)$.

Věta (Robinsonův rezoluční princip)

Množina klauzulí klauzulární logiky M je nesplnitelná, jestliže existuje přirozené číslo n takové, že $\mathcal{R}_n(M)$ obsahuje prázdnou klauzuli $\rightarrow$.

Důkaz

- Převedeme množinu klauzulí $\mathcal{M}$ do predikátové logiky, mezi klauzulemi je vztah konjunkce:
  $C_1 \,\&\, C_2 \,\&\, \ldots \,\&\, C_k$,
- totéž provedeme s množinami $\mathcal{R}_i(\mathcal{M}),\ 1 \leq i \leq n$,
- prázdnou klauzuli $\rightarrow$ převedeme na $true \rightarrow false$,
- rezoluční pravidlo zachovává splnitelnost,
- důkaz provedeme matematickou indukcí „zpětně“ od $\mathcal{R}_n(\mathcal{M}) \ni (true \rightarrow false)$.

Věta (Robinsonův rezoluční princip)

Množina klauzulí klauzulární logiky M je nesplnitelná, jestliže existuje přirozené číslo n takové, že $\mathcal{R}_n(M)$ obsahuje prázdnou klauzuli $\rightarrow$.

Důkaz

- Převedeme množinu klauzulí $\mathcal{M}$ do predikátové logiky, mezi klauzulemi je vztah konjunkce:
  $C_1 \,\&\, C_2 \,\&\, \ldots \,\&\, C_k$,
- totéž provedeme s množinami $\mathcal{R}_i(\mathcal{M}),\ 1 \leq i \leq n$,
- prázdnou klauzuli $\rightarrow$ převedeme na $true \rightarrow false$,
- rezoluční pravidlo zachovává splnitelnost,
- důkaz provedeme matematickou indukcí „zpětně“ od $\mathcal{R}_n(\mathcal{M}) \ni (true \rightarrow false)$.

Věta (Robinsonův rezoluční princip)

Množina klauzulí klauzulární logiky M je nesplnitelná, jestliže existuje přirozené číslo n takové, že $\mathcal{R}_n(M)$ obsahuje prázdnou klauzuli $\rightarrow$.

Důkaz

- Převedeme množinu klauzulí $\mathcal{M}$ do predikátové logiky, mezi klauzulemi je vztah konjunkce:
  $C_1 \,\&\, C_2 \,\&\, \ldots \,\&\, C_k$,
- totéž provedeme s množinami $\mathcal{R}_i(\mathcal{M}),\ 1 \leq i \leq n$,
- prázdnou klauzuli $\rightarrow$ převedeme na $true \rightarrow false$,
- rezoluční pravidlo zachovává splnitelnost,
- důkaz provedeme matematickou indukcí „zpětně“ od $\mathcal{R}_n(\mathcal{M}) \ni (true \rightarrow false)$.

Věta (Robinsonův rezoluční princip)

Množina klauzulí klauzulární logiky M je nesplnitelná, jestliže existuje přirozené číslo n takové, že $\mathcal{R}_n(M)$ obsahuje prázdnou klauzuli $\rightarrow$.

Důkaz

- Převedeme množinu klauzulí $\mathcal{M}$ do predikátové logiky, mezi klauzulemi je vztah konjunkce:
  $C_1 \,\&\, C_2 \,\&\, \ldots \,\&\, C_k$,
- totéž provedeme s množinami $\mathcal{R}_i(\mathcal{M}),\ 1 \leq i \leq n$,
- prázdnou klauzuli $\rightarrow$ převedeme na $true \rightarrow false$,
- rezoluční pravidlo zachovává splnitelnost,
- důkaz provedeme matematickou indukcí „zpětně“ od $\mathcal{R}_n(\mathcal{M}) \ni (true \rightarrow false)$.

Důkaz

1. Báze indukce: $\mathcal{R}_n(\mathcal{M})$ obsahuje prázdnou klauzuli $(true \rightarrow false)$, proto je nesplnitelná.
2. Předpoklad indukce: $\mathcal{R}_i(\mathcal{M})$ pro některé i, $1 \leq i \leq n$ je nesplnitelná.
3. Krok indukce: vezmeme formuli $\mathcal{R}_{i-1}(\mathcal{M})$. Z této formule byla odvozena některá $\mathcal{R}_j(\mathcal{M})$ pro $i \leq j \leq n$, o kterých už víme, že jsou nesplnitelné.
   O formulích $\mathcal{R}_s(\mathcal{M})$, $0 \leq s < (i-1)$ víme, že formule $\mathcal{R}_{i-1}(\mathcal{M})$ obsahuje všechny konjunkty (podformule spojené konjunkcí) obsažené v $\mathcal{R}_s(\mathcal{M})$, případně něco navíc.

   Protože rezoluční pravidlo zachovává splnitelnost, pak by v případě splnitelnosti $\mathcal{R}_{i-1}(\mathcal{M})$ byla splnitelná i některá z formulí s vyšším indexem $\Rightarrow$ spor, $\mathcal{R}_{i-1}(\mathcal{M})$ nemůže být splnitelná.

Důkaz

1. Báze indukce: $\mathcal{R}_n(\mathcal{M})$ obsahuje prázdnou klauzuli $(true \rightarrow false)$, proto je nesplnitelná.
2. Předpoklad indukce: $\mathcal{R}_i(\mathcal{M})$ pro některé i, $1 \leq i \leq n$ je nesplnitelná.
3. Krok indukce: vezmeme formuli $\mathcal{R}_{i-1}(\mathcal{M})$. Z této formule byla odvozena některá $\mathcal{R}_j(\mathcal{M})$ pro $i \leq j \leq n$, o kterých už víme, že jsou nesplnitelné.
   O formulích $\mathcal{R}_s(\mathcal{M})$, $0 \leq s < (i-1)$ víme, že formule $\mathcal{R}_{i-1}(\mathcal{M})$ obsahuje všechny konjunkty (podformule spojené konjunkcí) obsažené v $\mathcal{R}_s(\mathcal{M})$, případně něco navíc.

   Protože rezoluční pravidlo zachovává splnitelnost, pak by v případě splnitelnosti $\mathcal{R}_{i-1}(\mathcal{M})$ byla splnitelná i některá z formulí s vyšším indexem $\Rightarrow$ spor, $\mathcal{R}_{i-1}(\mathcal{M})$ nemůže být splnitelná.

Důkaz

1. Báze indukce: $\mathcal{R}_n(\mathcal{M})$ obsahuje prázdnou klauzuli $(true \rightarrow false)$, proto je nesplnitelná.
2. Předpoklad indukce: $\mathcal{R}_i(\mathcal{M})$ pro některé $i,\ 1 \leq i \leq n$ je nesplnitelná.
3. Krok indukce: vezmeme formuli $\mathcal{R}_{i-1}(\mathcal{M})$. Z této formule byla odvozena některá $\mathcal{R}_j(\mathcal{M})$ pro $i \leq j \leq n$, o kterých už víme, že jsou nesplnitelné.
   O formulích $\mathcal{R}_s(\mathcal{M}),\ 0 \leq s < (i-1)$ víme, že formule $\mathcal{R}_{i-1}(\mathcal{M})$ obsahuje všechny konjunkty (podformule spojené konjunkcí) obsažené v $\mathcal{R}_s(\mathcal{M})$, případně něco navíc.

   Protože rezoluční pravidlo zachovává splnitelnost, pak by v případě splnitelnosti $\mathcal{R}_{i-1}(\mathcal{M})$ byla splnitelná i některá z formulí s vyšším indexem $\Rightarrow$ spor, $\mathcal{R}_{i-1}(\mathcal{M})$ nemůže být splnitelná.

Důsledek

Vyhodnocení dotazu v bázi

Z Robinsonova rezolučního principu vychází použití nepřímého odvození v logických programovacích jazycích:

- vytvoříme bázi,
- dotaz znegujeme a přidáme k bázi,
- pokusíme se dojít k prázdné klauzuli,
- když k ní dojdeme, vrátíme $true$, jinak $false$.

Výpočetní strom

Prohledávání výpočetního stromu

Počet prvků množiny $\mathcal{R}_n(\mathcal{M})$ roste moc rychle (až exponenciálně). Jestliže postupujeme sekvenčně (v každém kroku zpracujeme jednu dvojici klauzulí a jejich rezolventu – jednu klauzuli – přidáme do množiny), lze vytvořit výpočetní strom, jehož uzly obsahují množiny $\mathcal{R}_i(\mathcal{M})$, ve kterých hledáme prázdnou klauzuli.

Tento strom prohledáváme

- do šířky – podle množiny $\mathcal{R}_i(\mathcal{M})$ postupně tvoříme celou množinu $\mathcal{R}_{i+1}(\mathcal{M})$, pak celou $\mathcal{R}_{i+2}(\mathcal{M})$, atd.
  - nevýhoda: zabere moc místa v paměti, „stack overflow“
- do hloubky – stanovíme způsob výběru dvojice klauzulí, na které použijeme rezoluční pravidlo, rezolventu přidáme do množiny, atd., negenerujeme celou $\mathcal{R}_{i+1}(\mathcal{M})$.
  - nevýhoda: náhodou se můžeme trefit do větve, ve které se prázdná klauzule nenachází a je nekonečná.

Výpočetní strom

Prohledávání výpočetního stromu

Počet prvků množiny $\mathcal{R}_n(\mathcal{M})$ roste moc rychle (až exponenciálně). Jestliže postupujeme sekvenčně (v každém kroku zpracujeme jednu dvojici klauzulí a jejich rezolventu – jednu klauzuli – přidáme do množiny), lze vytvořit výpočetní strom, jehož uzly obsahují množiny $\mathcal{R}_i(\mathcal{M})$, ve kterých hledáme prázdnou klauzuli.
Tento strom prohledáváme

- do šířky – podle množiny $\mathcal{R}_i(\mathcal{M})$ postupně tvoříme celou množinu $\mathcal{R}_{i+1}(\mathcal{M})$, pak celou $\mathcal{R}_{i+2}(\mathcal{M})$, atd.
  - nevýhoda: zabere moc místa v paměti, „stack overflow“
- do hloubky – stanovíme způsob výběru dvojice klauzulí, na které použijeme rezoluční pravidlo, rezolventu přidáme do množiny, atd., negenerujeme celou $\mathcal{R}_{i+1}(\mathcal{M})$.
  - nevýhoda: náhodou se můžeme trefit do větve, ve které se prázdná klauzule nenachází a je nekonečná.

Výpočetní strom

Prohledávání výpočetního stromu

Počet prvků množiny $\mathcal{R}_n(\mathcal{M})$ roste moc rychle (až exponenciálně). Jestliže postupujeme sekvenčně (v každém kroku zpracujeme jednu dvojici klauzulí a jejich rezolventu – jednu klauzuli – přidáme do množiny), lze vytvořit výpočetní strom, jehož uzly obsahují množiny $\mathcal{R}_i(\mathcal{M})$, ve kterých hledáme prázdnou klauzuli.
Tento strom prohledáváme

- do šířky – podle množiny $\mathcal{R}_i(\mathcal{M})$ postupně tvoříme celou množinu $\mathcal{R}_{i+1}(\mathcal{M})$, pak celou $\mathcal{R}_{i+2}(\mathcal{M})$, atd.
  - nevýhoda: zabere moc místa v paměti, „stack overflow“
- do hloubky – stanovíme způsob výběru dvojice klauzulí, na které použijeme rezoluční pravidlo, rezolventu přidáme do množiny, atd., negenerujeme celou $\mathcal{R}_{i+1}(\mathcal{M})$.
  - nevýhoda: náhodou se můžeme trefit do větve, ve které se prázdná klauzule nenachází a je nekonečná.

Výpočetní strom

Řešení nevýhod

Obvyklé řešení je:

- používá se prohledávání do hloubky,
- dvojice klauzulí pro rezoluci jsou vybírány lineární metodou – jedna klauzule ve dvojici je rezolventa předchozího kroku, tedy poslední přidaná.

Výpočetní strom tak můžeme zredukovat tak, že v uzlech budou jen klauzule.

Lineární výpočetní strom

Definice

Lineární výpočetní strom klauzule je takový strom, kde:

- *všechny uzly jsou ohodnoceny klauzulemi,*
- *kořen je ohodnocen cílovou klauzulí,*
- *pro všechny uzly stromu platí: jestliže je uzel ohodnocen klauzulí C, pak každý jeho potomek je ohodnocen klauzulí vzniklou uplatněním rezolučního odvozovacího pravidla na klauzuli C a některou klauzuli znalostní báze.*

Příklad pro klauzulární logiku

1. $A, B \to C$
2. $D \to B$
3. $\to A$
4. $C \to$

Chceme dokázat $D \to$

5.	$\to D$	PM
6.	$\to B$	R(2,5)
7.	$A \to C$	R(1,6)
8.	$\to C$	R(3,7)
9.	$\to$	R(4,8)

5.	$\to D$	PM
6.	$\to B$	R(2,5)
7.	$A \to C$	R(1,6)
8.	$A \to$	R(4,7)
9.	$\to$	R(3,8)

Příklad pro klauzulární logiku

1. $A, B \to C$
2. $D \to B$
3. $\to A$
4. $C \to$

Chceme dokázat $D \to$

5.	$\to D$	PM
6.	$\to B$	R(2,5)
7.	$A \to C$	R(1,6)
8.	$\to C$	R(3,7)
9.	$\to$	R(4,8)

5.	$\to D$	PM
6.	$\to B$	R(2,5)
7.	$A \to C$	R(1,6)
8.	$A \to$	R(4,7)
9.	$\to$	R(3,8)

Příklad pro klauzulární logiku

1. $A, B \rightarrow C$
2. $D \rightarrow B$
3. $\rightarrow A$
4. $C \rightarrow$

Chceme dokázat $D \rightarrow$

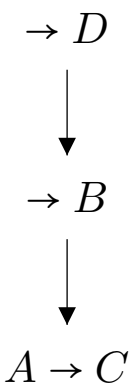
5.	$\rightarrow D$	PM
6.	$\rightarrow B$	R(2,5)
7.	$A \rightarrow C$	R(1,6)
8.	$\rightarrow C$	R(3,7)
9.	$\rightarrow$	R(4,8)

5.	$\rightarrow D$	PM
6.	$\rightarrow B$	R(2,5)
7.	$A \rightarrow C$	R(1,6)
8.	$A \rightarrow$	R(4,7)
9.	$\rightarrow$	R(3,8)

Příklad pro klauzulární logiku

1. $A, B \to C$
2. $D \to B$
3. $\to A$
4. $C \to$

5. $\to D$ PM
6. $\to B$ R(2,5)
7. $A \to C$ R(1,6)
8. $\to C$ R(3,7)
9. $\to$ R(4,8)



Chceme dokázat $D \to$

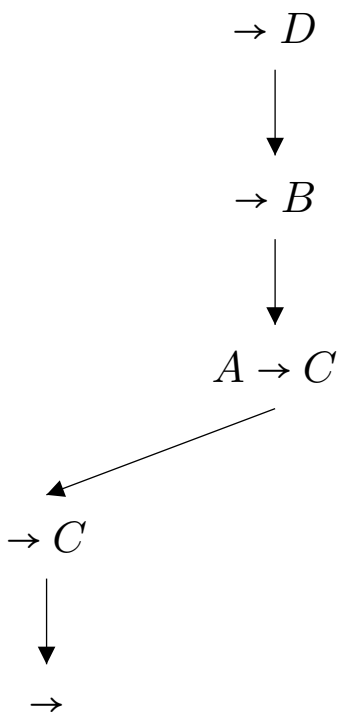
5. $\to D$ PM
6. $\to B$ R(2,5)
7. $A \to C$ R(1,6)
8. $A \to$ R(4,7)
9. $\to$ R(3,8)

Příklad pro klauzulární logiku

1. $A, B \to C$
2. $D \to B$
3. $\to A$
4. $C \to$

Chceme dokázat $D \to$

5.	$\to D$	PM
6.	$\to B$	R(2,5)
7.	$A \to C$	R(1,6)
8.	$\to C$	R(3,7)
9.	$\to$	R(4,8)



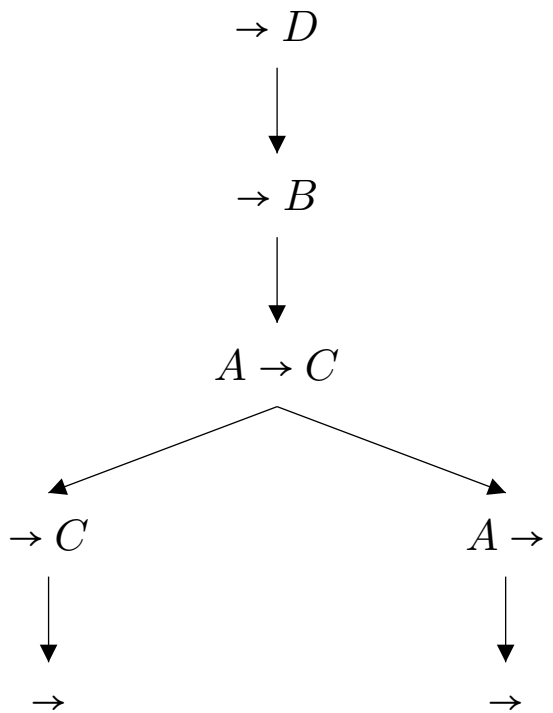
5.	$\to D$	PM
6.	$\to B$	R(2,5)
7.	$A \to C$	R(1,6)
8.	$A \to$	R(4,7)
9.	$\to$	R(3,8)

Příklad pro klauzulární logiku

1. $A, B \to C$
2. $D \to B$
3. $\to A$
4. $C \to$

Chceme dokázat $D \to$

5.	$\to D$	PM
6.	$\to B$	R(2,5)
7.	$A \to C$	R(1,6)
8.	$\to C$	R(3,7)
9.	$\to$	R(4,8)



5.	$\to D$	PM
6.	$\to B$	R(2,5)
7.	$A \to C$	R(1,6)
8.	$A \to$	R(4,7)
9.	$\to$	R(3,8)

Výpočetní strom

Použití v Prologu

- používáme lineární metodu výběru klauzulí pro rezoluci, pro uplatnění rezolučního pravidla vždy vezmeme
  - rezolventu z předchozího kroku a
  - další – vybíráme směrem od začátku báze, atomy pro unifikaci zleva doprava,
- prohledáváme strom do hloubky.

Výpočetní strom

Použití v Prologu

- používáme lineární metodu výběru klauzulí pro rezoluci, pro uplatnění rezolučního pravidla vždy vezmeme
  - rezolventu z předchozího kroku a
  - další – vybíráme směrem od začátku báze, atomy pro unifikaci zleva doprava,
- prohledáváme strom do hloubky.

Výpočetní strom

Použití v Prologu

- používáme lineární metodu výběru klauzulí pro rezoluci, pro uplatnění rezolučního pravidla vždy vezmeme
  - rezolventu z předchozího kroku a
  - další – vybíráme směrem od začátku báze, atomy pro unifikaci zleva doprava,
- prohledáváme strom do hloubky.

Výpočetní strom

Použití v Prologu

- používáme lineární metodu výběru klauzulí pro rezoluci, pro uplatnění rezolučního pravidla vždy vezmeme
  - rezolventu z předchozího kroku a
  - další – vybíráme směrem od začátku báze, atomy pro unifikaci zleva doprava,
- prohledáváme strom do hloubky.

Výpočetní strom

Jak předejít zacyklení výpočtu

1. Ve znalostní bázi nejdřív uvádíme fakty, pak pravidla. Zjednodušíme tak unifikaci, nedochází ke zbytečnému opakování výpočtu a tím někdy i k rekurzivnímu vyhodnocování klauzulí.
2. Jestliže je v těle klauzule atom se stejným predikátem jako atom v hlavě klauzule, třeba i s jinými argumenty, pak takový atom umístíme až na konec těla klauzule.
3. Další pravidla umístění klauzulí v bázi a predikátů v klauzulích – například vzhledem k predikátu `not`.

Výpočetní strom

Jak předejít zacyklení výpočtu

1. Ve znalostní bázi nejdřív uvádíme fakty, pak pravidla. Zjednodušíme tak unifikaci, nedochází ke zbytečnému opakování výpočtu a tím někdy i k rekurzivnímu vyhodnocování klauzulí.
2. Jestliže je v těle klauzule atom se stejným predikátem jako atom v hlavě klauzule, třeba i s jinými argumenty, pak takový atom umístíme až na konec těla klauzule.
3. Další pravidla umístění klauzulí v bázi a predikátů v klauzulích – například vzhledem k predikátu `not`.

Výpočetní strom

Jak předejít zacyklení výpočtu

1. Ve znalostní bázi nejdřív uvádíme fakty, pak pravidla. Zjednodušíme tak unifikaci, nedochází ke zbytečnému opakování výpočtu a tím někdy i k rekurzivnímu vyhodnocování klauzulí.
2. Jestliže je v těle klauzule atom se stejným predikátem jako atom v hlavě klauzule, třeba i s jinými argumenty, pak takový atom umístíme až na konec těla klauzule.
3. Další pravidla umístění klauzulí v bázi a predikátů v klauzulích – například vzhledem k predikátu `not`.