

## Negace v logickém programování

## Negativní znalost

- logické programy vyjadřují **pozitivní znalost**
- **negativní literály**: pozice určena definicí Hornových klauzulí  
⇒ nelze vyvodit **negativní** informaci z logického programu
  - každý predikát definuje úplnou relaci
  - negativní literál **není** logickým důsledkem programu
- relace vyjádřeny explicitně v nejmenším Herbrandově modelu
  - $nad(X, Y) : \neg na(X, Y). \quad na(c, b).$
  - $nad(X, Y) : \neg na(X, Z), nad(Z, Y). \quad na(b, a).$
  - nejmenší Herbrandův model:  $\{na(b, a), na(c, b), nad(b, a), nad(c, b), nad(c, a)\}$
- ani program ani model nezahrnují negativní informaci
  - $a$  není nad  $c$ ,  $a$  není na  $c$
  - i v realitě je negativní informace vyjádřena explicitně zřídka, např. jízdní řád

## Předpoklad uzavřeného světa

- neexistence informace chápána jako opak:  
**předpoklad uzavřeného světa (closed world assumption, CWA)**
- převzato z databází
- určitý vztah platí **pouze** když je vyvoditelný z programu.
- „odvozovací pravidlo“ ( $A$  je (uzavřený) term):  $\frac{P \neq A}{\neg A}$  (CWA)
- pro SLD-rezoluci:  $P \neq nad(a, c)$ , tedy lze podle CWA odvodit  $\neg nad(a, c)$
- problém: není rozhodnutelné, zda daná atomická formule je logickým důsledkem daného logického programu.
  - nelze tedy určit, zda pravidlo CWA je aplikovatelné nebo ne
- CWA v logickém programování obecně nepoužitelná.

## Negace jako neúspěch (*negation as failure*)

- slabší verze CWA: **definitivně neúspěšný (finitely failed) SLD-strom** cíle :  $\neg A$   
:  $\frac{\neg A}{\neg A}$  má definitivně (konečně) neúspěšný SLD-strom (**negation as failure, NF**)
- **normální cíl**: cíl obsahující i negativní literály
  - :  $\neg nad(c, a), \neg nad(b, c).$
- rozdíl mezi CWA a NF
  - program  $nad(X, Y) : \neg nad(X, Y)$ , cíl :  $\neg \neg nad(b, c)$
  - neexistuje odvození cíle podle NF, protože SLD-strom :  $\neg nad(b, c)$  je nekonečný
  - existuje odvození cíle podle CWA, protože neexistuje vyvrácení :  $\neg nad(b, c)$
- CWA i NF jsou nekorektní:  $A$  není logickým důsledkem programu  $P$
- řešení: definovat programy tak, aby jejich důsledkem byly i negativní literály  
**zúplnění logického programu**

## Podstata zúplnění logického programu

- převod všech **if** příkazů v logickém programu na **iff**
  - $nad(X, Y) : \neg na(X, Y)$ .
  - $nad(X, Y) : \neg na(X, Z), nad(Z, Y)$ .
  - lze psát jako:  $nad(X, Y) : \neg(na(X, Y)) \vee (na(X, Z), nad(Z, Y))$ .
  - zúplnění:  $nad(X, Y) \leftrightarrow (na(X, Y)) \vee (na(X, Z), nad(Z, Y))$ .
  - $X$  je nad  $Y$  **právě tehdy, když alespoň jedna z podmínek platí**
  - tedy **pokud žádná z podmínek neplatí,  $X$  není nad  $Y$**
- kombinace klauzulí je možná pouze pokud mají identické hlavy
  - $na(c, b)$ .
  - $na(b, a)$ .
  - lze psát jako:  $na(X_1, X_2) : \neg X_1 = c, X_2 = b$ .
  - $na(X_1, X_2) : \neg X_1 = b, X_2 = a$ .
  - zúplnění:  $na(X_1, X_2) \leftrightarrow (X_1 = c, X_2 = b) \vee (X_1 = b, X_2 = a)$ .

### $IF(q, P)$

$$na(X_1, X_2) : \neg \exists Y (X_1 = c, X_2 = b, f(Y)) \vee (X_1 = b, X_2 = a, g).$$

- $q/n$  predikátový symbol programu  $P$   $\quad \quad \quad \underline{na(c, b) : -f(Y)} \quad \quad \quad \underline{na(b, a) : -g}$
- $X_1, \dots, X_n$  jsou „nové“ proměnné, které se nevyskytují nikde v  $P$
- Necht'  $C$  je klauzule ve tvaru  
 $q(t_1, \dots, t_n) : \neg L_1, \dots, L_m$   
kde  $m \geq 0$ ,  $t_1, \dots, t_n$  jsou termy a  $L_1, \dots, L_m$  jsou literály.  
  
Pak označme  $E(C)$  výraz  $\exists Y_1, \dots, Y_k (X_1 = t_1, \dots, X_n = t_n, L_1, \dots, L_m)$   
kde  $Y_1, \dots, Y_k$  jsou všechny proměnné v  $C$ .
- Necht'  $def(q/n) = \{C_1, \dots, C_j\}$ .  
  
Pak formuli  $IF(q, P)$  získáme následujícím postupem:  
 $q(X_1, \dots, X_n) : \neg E(C_1) \vee E(C_2) \vee \dots \vee E(C_j)$  pro  $j > 0$  a  
 $q(X_1, \dots, X_n) : \neg \square$  pro  $j = 0$  [ $q/n$  není v programu  $P$ ].

## Zúplnění programu

- **Zúplnění programu  $P$**  je:  $comp(P) := IFF(P) \cup CET$
- Základní vlastnosti:
  - $comp(P) \models P$
  - do programu je přidána pouze negativní informace
- **IFF(P)**: spojka  $:-$  v  $IF(P)$  je nahrazena spojkou  $\leftrightarrow$
- **IF(P)**: množina všech formulí  $IF(q, P)$   
pro všechny predikátové symboly  $q$  v programu  $P$
- Cíl: definovat  $IF(q, P)$
- $def(p/n)$  predikátu  $p/n$  je množina všech klauzulí predikátu  $p/n$

## Clarkova Teorie Rovnosti (CET)

všechny formule jsou univerzálně kvantifikovány:

1.  $X = X$
2.  $X = Y \rightarrow Y = X$
3.  $X = Y \wedge Y = Z \rightarrow X = Z$
4. pro každý  $f/m$ :  $X_1 = Y_1 \wedge \dots \wedge X_m = Y_m \rightarrow f(X_1, \dots, X_m) = f(Y_1, \dots, Y_m)$
5. pro každý  $p/m$ :  $X_1 = Y_1 \wedge \dots \wedge X_m = Y_m \rightarrow (p(X_1, \dots, X_m) \rightarrow p(Y_1, \dots, Y_m))$
6. pro všechny různé  $f/m$  a  $g/n$ , ( $m, n \geq 0$ ):  $f(X_1, \dots, X_m) \neq g(Y_1, \dots, Y_n)$
7. pro každý  $f/m$ :  $f(X_1, \dots, X_m) = f(Y_1, \dots, Y_m) \rightarrow X_1 = Y_1 \wedge \dots \wedge X_m = Y_m$
8. pro každý term  $t$  obsahující  $X$  jako vlastní podterm:  $t \neq X$

$X \neq Y$  je zkrácený zápis  $\neg(X = Y)$

## Korektnost a úplnost NF pravidla

- **Korektnost NF pravidla:** Necht'  $P$  logický program a :  $\neg A$  cíl.  
Jestliže :  $\neg A$  má definitivně neúspěšný SLD-strom,  
pak  $\forall (\neg A)$  je logickým důsledkem  $\text{comp}(P)$  (nebo-li  $\text{comp}(P) \models \forall (\neg A)$ )
- **Úplnost NF pravidla:** Necht'  $P$  je logický program. Jestliže  $\text{comp}(P) \models \forall (\neg A)$ ,  
pak existuje definitivně neúspěšný SLD-strom :  $\neg A$ .
  - zůstává problém: není rozhodnutelné, zda daná atomická formule je logickým důsledkem daného logického programu.
  - teorém mluví pouze o **existenci** definitivně neúspěšného SLD-stromu
  - definitivně (konečně) neúspěšný SLD-strom sice existuje, ale nemusíme ho nalézt
    - např. v Prologu: může existovat konečné odvození, ale program přesto cyklí (Prolog nenajde definitivně neúspěšný strom)
- Odvození pomocí NF pouze **test**, nelze **konstruovat** výslednou substituci
  - v  $(\text{comp}(P) \models \forall (\neg A))$  je  $A$  všeob. kvantifikováno, v  $\forall (\neg A)$  nejsou volné proměnné

## Stratifikované programy II

- program je  $m$ -stratifikovaný  $\iff m$  je nejmenší index takový, že  $S_0 \cup \dots \cup S_m$  je množina všech predikátových symbolů z  $P$
- **Věta:** Zúplnění každého stratifikovaného programu má Herbrandův model.
  - $p : \neg \neg p$ . nemá Herbrandův model
  - $p : \neg p$ . ale není stratifikovaný
- stratifikované programy nemusí mít **jedinečný** minimální Herbrandův model
  - *cykli* :  $\neg \neg \text{zastavi}$ .
  - dva minimální Herbrandovy modely:  $\{\text{cykli}\}, \{\text{zastavi}\}$
  - důsledek toho, že *cykli* :  $\neg \neg \text{zastavi}$ . je ekvivalentní *cykli*  $\vee$  *zastavi*

## Normální a stratifikované programy

- **normální program:** obsahuje negativní literály v pravidlech
- **problém:** existence zúplnění, která nemají žádný model
  - $p : \neg \neg p$ . zúplnění:  $p \leftarrow \neg p$
- **rozdělení programu na vrstvy**
  - vynucují použití negace relace pouze tehdy pokud je relace úplně definovaná
  - $a$ .  $a$ .  
 $a : \neg \neg b, a$ .  $a : \neg \neg b, a$ .  
 $b$ .  $b : \neg \neg a$ .  
stratifikovaný není stratifikovaný
- normální program  $P$  je **stratifikovaný**: množina predikátových symbolů programu lze rozdělit do disjunktních množin  $S_0, \dots, S_m$  ( $S_i \equiv$  **stratum**)
  - $p(\dots) : \neg \dots, q(\dots), \dots \in P, p \in S_k \implies q \in S_0 \cup \dots \cup S_k$
  - $p(\dots) : \neg \dots, \neg q(\dots), \dots \in P, p \in S_k \implies q \in S_0 \cup \dots \cup S_{k-1}$

## SLDNF rezoluce: úspěšné odvození

- NF pravidlo:  $\frac{}{\neg C}$  má konečně neúspěšný SLD-strom
- Pokud máme negativní podcíl  $\neg C$  v dotazu  $G$ , pak hledáme důkaz pro  $C$
- Pokud odvození  $C$  selže (strom pro  $C$  je konečně neúspěšný), pak je odvození  $G$  (i  $\neg C$ ) celkově úspěšné

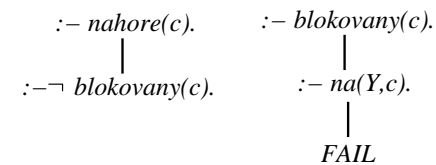
*nahore*( $X$ ) :  $\neg \neg \text{blokovaný}(X)$ .

*blokovaný*( $X$ ) :  $\neg \text{na}(Y, X)$ .

*na*( $a, b$ ).

:  $\neg \text{nahore}(c)$ .

*yes*



$\Rightarrow$  úspěšné odvození

## SLDNF rezoluce: neúspěšné odvození

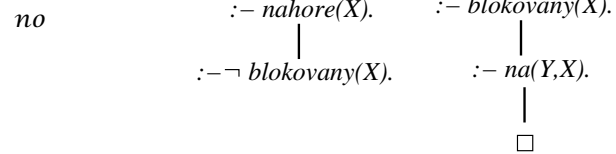
- NF pravidlo: 
$$\frac{:- C. \text{ má konečně neúspěšný SLD-strom}}{\neg C}$$
- Pokud máme negativní podcíl  $\neg C$  v dotazu  $G$ , pak hledáme důkaz pro  $C$
- Pokud existuje vyvrácení  $C$  s prázdnou substitucí (strom pro  $C$  je konečně úspěšný), pak je odvození  $G$  (i  $\neg C$ ) celkově neúspěšné

$nahore(X) : \neg \neg blokovany(X).$

$blokovany(X) : \neg na(Y, X).$

$na(., .).$

$:- nahore(X).$



$\Rightarrow$  neúspěšné odvození

## SLDNF rezoluce: uvážené odvození

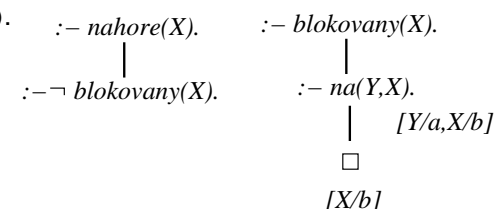
- NF pravidlo: 
$$\frac{:- C. \text{ má konečně neúspěšný SLD-strom}}{\neg C}$$
- Pokud máme negativní podcíl  $\neg C$  v dotazu  $G$ , pak hledáme důkaz pro  $C$
- Pokud existuje vyvrácení  $C$  s neprázdnou substitucí (strom pro  $C$  je konečně úspěšný), pak je odvození  $G$  (i  $\neg C$ ) uvážené

$nahore(X) : \neg \neg blokovany(X).$

$blokovany(X) : \neg na(Y, X).$

$na(a, b).$

$:- nahore(X).$



$\Rightarrow$  uvážené odvození

## SLD<sup>+</sup> odvození

- $P$  je normální program,  $G_0$  normální cíl,  $R$  selekční pravidlo:  
SLD<sup>+</sup>-odvození  $G_0$  je buď konečná posloupnost

$$\langle G_0; C_0 \rangle, \dots, \langle G_{i-1}; C_{i-1} \rangle, G_i$$

nebo nekonečná posloupnost

$$\langle G_0; C_0 \rangle, \langle G_1; C_1 \rangle, \langle G_2; C_2 \rangle, \dots$$

kde v každém kroku  $m + 1$  ( $m \geq 0$ ),  $R$  vybírá **pozitivní literál** v  $G_m$  a dospívá k  $G_{m+1}$  obvyklým způsobem.

- konečné SLD<sup>+</sup>-odvození může být:

1. **úspěšné:**  $G_i = \square$

2. **neúspěšné**

3. **blokováné:**  $G_i$  je negativní (např.  $\neg A$ )

## SLDNF rezoluce: pojmy

- **Úroveň cíle**
  - $P$  normální program,  $G_0$  normální cíl,  $R$  selekční pravidlo:  
**úroveň cíle**  $G_0$  se rovná
    - $0 \iff$  žádné SLD<sup>+</sup>-odvození s pravidlem  $R$  není blokováno
    - $k + 1 \iff$  maximální úroveň cílů:  $\neg A$ ,  
které ve tvaru  $\neg A$  blokují SLD<sup>+</sup>-odvození  $G_0$ , je  $k$
  - nekonečná úroveň cíle: **blokováné SLDNF odvození**
- **Množina SLDNF odvození** =  $\{\text{SLDNF odvození } G_0\} \cup \{\text{SLDNF odvození } : \neg A\}$ 
  - při odvozování  $G_0$  jsme se dostali k cíli  $\neg A$
- SLDNF odvození cíle  $G$  ?

## SLDNF rezoluce

$P$  normální program,  $G_0$  normální cíl,  $R$  selekční pravidlo:

**množina SLDNF odvození a podmnožina neúspěšných SLDNF odvození** cíle  $G_0$  jsou takové nejmenší množiny, že:

- každé **SLD<sup>+</sup>-odvození**  $G_0$  je SLDNF odvození  $G_0$
- je-li SLD<sup>+</sup>-odvození  $\langle G_0; C_0 \rangle, \dots, G_i$  **blokováno na**  $\neg A$ 
  - tj.  $G_i$  je tvaru :  $-L_1, \dots, L_{m-1}, \neg A, L_{m+1}, \dots, L_n$

pak

- **existuje-li SLDNF odvození** :  $\neg A$  (pod  $R$ ) s prázdnou cílovou substitucí, pak  $\langle G_0; C_0 \rangle, \dots, G_i$  je **neúspěšné SLDNF odvození**
- je-li **každé úplné SLDNF odvození** :  $\neg A$  (pod  $R$ ) **neúspěšné** pak  $\langle G_0; C_0 \rangle, \dots, \langle G_i, \epsilon \rangle, (: -L_1, \dots, L_{m-1}, L_{m+1}, \dots, L_n)$  je **(úspěšné) SLDNF odvození cíle**  $G_0$ 
  - $\epsilon$  označuje prázdnou cílovou substituci

## Typy SLDNF odvození

Konečné SLDNF-odvození může být:

1. **úspěšné**:  $G_i = \square$
2. **neúspěšné**
3. **uvázlé (flounder)**:  
 $G_i$  je negativní ( $\neg A$ ) a :  $\neg A$  je úspěšné s **neprázdnou cílovou substitucí**
4. **blokováno**:  $G_i$  je negativní ( $\neg A$ ) a :  $\neg A$  nemá konečnou úroveň.

## Korektnost a úplnost SLDNF odvození

- **korektnost SLDNF-odvození**:

$P$  normální program, :  $\neg G$  normální cíl a  $R$  je selekční pravidlo:

je-li  $\theta$  cílová substituce SLDNF-odvození cíle :  $\neg G$ , pak

$G\theta$  je logickým důsledkem  $\text{comp}(P)$

- implementace SLDNF v Prologu není korektní
- Prolog neřeší uvázlé SLDNF-odvození (neprázdná substituce)
- použití bezpečných cílů (negace neobsahuje proměnné)

- **úplnost SLDNF-odvození**: SLDNF-odvození **není** úplné

- pokud existuje konečný neúspěšný strom :  $\neg A$ , pak  $\neg A$  platí  
ale místo toho se odvozování :  $\neg A$  může zacyklit, tj. SLDNF rezoluce  $\neg A$  neodvodí  
 $\Rightarrow \neg A$  tedy sice platí, ale SLDNF rezoluce ho nedokáže odvodit