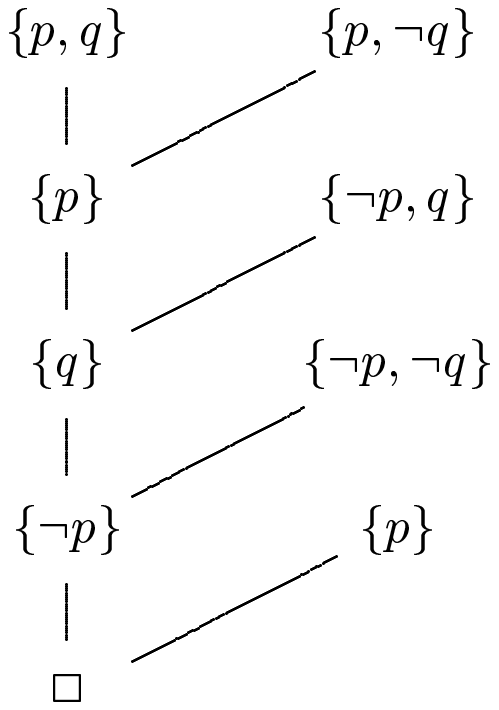
Lineární rezoluce

- další způsob zjemnění rezoluce; místo stromu směřujeme k lineární struktuře důkazu
- *Lineární rezoluční odvození (důkaz)* C z S je posloupnost dvojic $< C_0, B_0 >, \ldots, < C_n, B_n >$ taková, že $C = C_{n+1}$ a
  1. C_0 a všechna B_i jsou z S nebo nějaké C_j, kde $j < i$,
  2. každé $C_{i+1}, i \leq n$, je rezolventou C_i a B_i.
- C je *lineárně odvoditelná* z S ($S \vdash_{\mathcal{L}} C$), existuje-li lineární odvození C z S. *Lineární vyvrácení* S je odvození $S \vdash_{\mathcal{L}} \square$.
- Prvky S budeme nazývat *vstupní klauzule*, C_i *střední* a B_i *boční klauzule*.

Lineární rezoluce: příklad

- příklad: lineární vyvrácení množiny
  $S = \{\{p, q\}, \{p, \neg q\}, \{\neg p, q\}, \{\neg p, \neg q\}\}$



- lineární rezoluce je korektní a úplná

Hornovy klauzule a Prolog

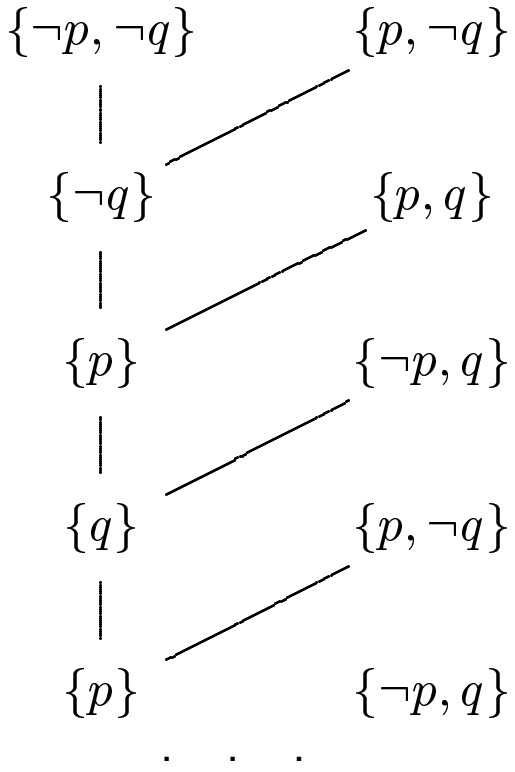
- omezení na určitý typ klauzulí používaných v programovacím jazyce Prolog
- *Hornova klauzule* je klauzule s nejvýše jedním pozitivním literálem.
  Příklad: $C = \{p, \neg q, \neg r\}$
  Běžný zápis ve výrokové logice $p \vee \neg q \vee \neg r$ lze ekvivalentními úpravami převést na $p \vee \neg(q \wedge r)$ a dále na $p \Leftarrow q \wedge r$, což v Prologu zapisujeme
  `p :- q, r.`
- typy Hornových klauzulí:
  – *programové* s právě jedním pozitivním literálem dále dělíme na
    * *fakta* bez negativních lit. (př.: $\{p\}$, v Prologu `p.`)
    * *pravidla* s alespoň jedním negativním lit. (př.: $\{p, \neg q\}$, `p :- q.`)
  – *cíle* bez pozitivního literálu (př.: $\{\neg p\}$, `:- p.` resp. `?- p.`)
- Každá nesplnitelná množina neprázdných Hornových klauzulí musí obsahovat alespoň jeden fakt a alespoň jeden cíl.

Prolog

- Prologovský program P: fakta + pravidla. Zajímá nás, co lze z P odvodit – obecně zda z něj vyplývá konjunkce faktů $q_1, q_2, \ldots, q_n$. Zadáme dotaz $? - q_1, q_2, \ldots, q_n.$ a obdržíme příslušnou odpověď (zpravidla yes/no). Prolog přidá cílovou klauzuli $G = \{\neg q_1, \neg q_2, \ldots, \neg q_n\}$ k P a zjišťuje nesplnitelnost $P \cup \{G\}$.
- *Věta*: je-li P prologovský program a $G = \{\neg q_1, \neg q_2, \ldots, \neg q_n\}$ cílová klauzule, pak všechna q_i jsou důsledkem P právě tehdy, když $P \cup \{G\}$ je nesplnitelná.
- pro prologovské programy (Hornovy klauzule) je v důkazu nesplnitelnosti nezbytné použít zadaný cíl; budeme tedy klást další požadavky na lineární rezoluci
- Nechť P je množina programových klauzulí a G cílová klauzule. *Lineární vstupní rezoluce (LI-rezoluce)* množiny $S = P \cup \{G\}$ je lineární vyvrácení S, které začíná klauzulí G a bočními klauzulemi jsou pouze klauzule z P.

LI-rezoluce: vlastnosti

- LI-rezoluce je korektní, ale pro obecné klauzule není úplná;
  příklad: $S = \{\{p, q\}, \{p, \neg q\}, \{\neg p, q\}, \{\neg p, \neg q\}\}$ je nesplnitelná, ale LI-rezoluční vyvrácení S neexistuje



- LI-rezoluce je úplná pro Hornovy klauzule

‚Implementace' Hornových klauzulí a rezoluce

- upravíme doposud používané struktury a mechanismy tak, aby byly dobře strojově zpracovatelné
- množiny reprezentující klauzule nahradíme *uspořádanými klauzulemi (definite clauses)*, např. místo $\{p, \neg q, \neg r\}$ použijeme $[p, \neg q, \neg r]$
- rezoluce pro uspořádané klauzule: mějme
  $G = [\neg p_1, \neg p_2, \ldots, \neg p_n]$ a
  $H = [q, \neg q_1, \neg q_2, \ldots, \neg q_m]$,
  rezolventou G a H pro $p_i = \neg q$ bude uspořádaná klauzule
  $[\neg p_1, \neg p_2, \ldots, \neg p_{i-1}, \neg q_1, \neg q_2, \ldots, \neg q_m, \neg p_{i+1}, \ldots, \neg p_n]$
- Je-li $P \cup \{G\}$ množina uspořádaných klauzulí, pak *LD-rezoluční vyvrácení* $P \cup \{G\}$ je posloupnost $< G_0, C_0 >, \ldots, < G_n, C_n >$ uspořádaných klauzulí taková, že $G_0 = G$, $G_{n+1} = \square$, $C_i \in P$ a každé $G_i, 1 \leq i \leq n$, je rezolventou uspořádaných klauzulí G_{i-1} a C_{i-1}.

LD a SLD-rezoluce

- úplnost LD-rezoluce: nechť P je množina programových klauzulí, G cílová klauzule. Je-li $P \cup \{G\}$ nesplnitelná, pak existuje LD-rezoluční vyvrácení $P \cup \{G\}$ začínající G.
- další problém při strojovém zpracování: výběr literálu, na kterém se bude rezolvovat
- *selekční pravidlo* R je libovolná funkce, která vybere literál z každé uspořádané cílové klauzule
- *SLD-rezoluce* je lineární vstupní rezoluce se selekčním pravidlem
- *SLD-rezoluční vyvrácení* $P \cup \{G\}$ pomocí selekčního pravidla R je LD-rezoluční vyvrácení $< G_0, C_0 >, \ldots, < G_n, C_n >$, $G_0 = G$, $G_{n+1} = \square$, kde $R(G_i)$ je literál, na kterém rezolvujeme v $i+1$-ním kroku. *Pozn.*: bez explicitní definice R se vybírá nejlevější literál.

SLD-rezoluce, SLD-stromy

- úplnost SLD-rezoluce pro Prolog: nechť P je množina programových klauzulí, G cílová klauzule, R libovolné selekční pravidlo. Je-li $P \cup \{G\}$ nesplnitelná, pak existuje SLD-rezoluční vyvrácení $P \cup \{G\}$ pomocí R.
- Prostor všech možných SLD-derivací při vyhodnocování daného cíle G v Prologu pro program P lze reprezentovat stromem T, který označujeme jako *SLD-strom* pro program P a cíl G.
  T má v kořeni G. Je-li libovolný uzel T označen G', pak jeho bezprostřední následníci jsou označeni výsledkem rezoluce nejlevějšího literálu G' se všemi použitelnými klauzulemi z P.

SLD-stromy: příklad I

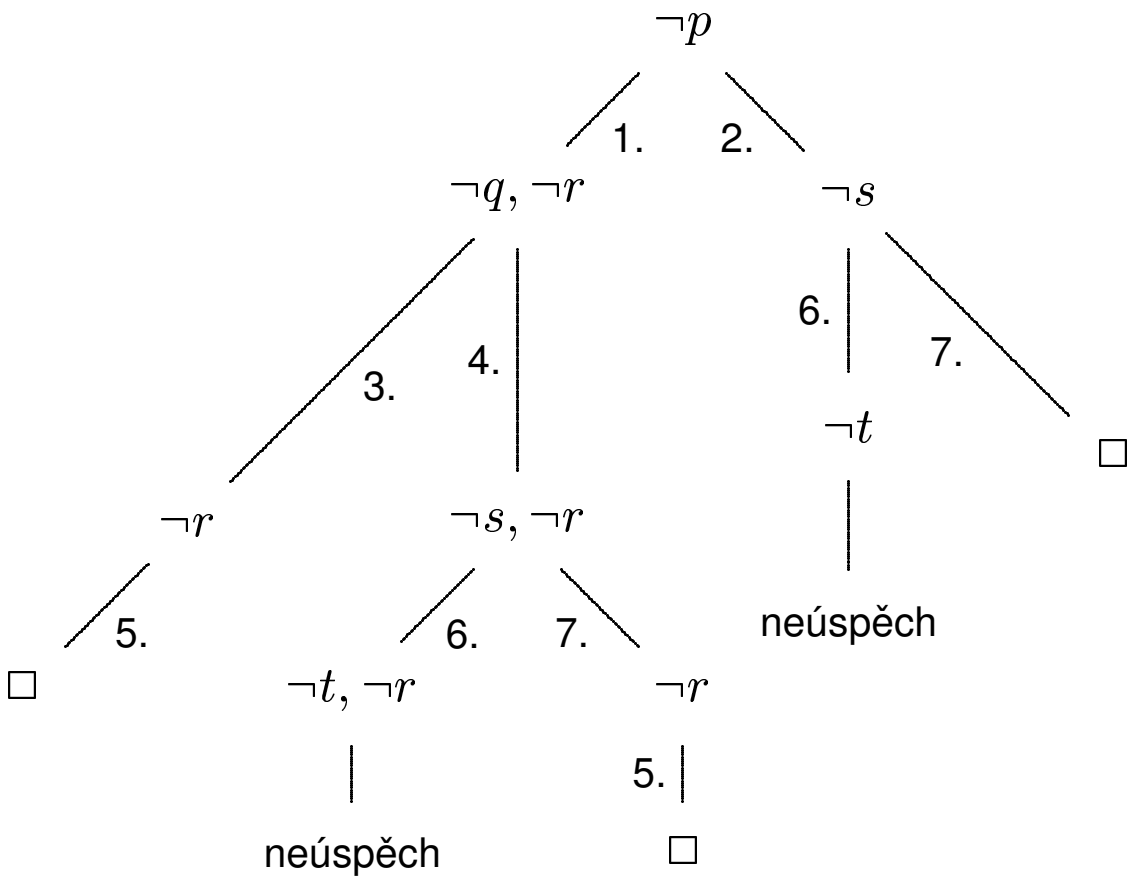
- Příklad: mějme následující program (klauzule jsou pro přehlednost očíslovány):

  1. `p:- q, r.`
  2. `p:- s.`
  3. `q.`
  4. `q:- s.`
  5. `r.`
  6. `s:- t.`
  7. `s.`

  a dotaz `?- p.` (tedy $G = \{\neg p\}$)

SLD-stromy: příklad II

- odpovídající SLD-strom:

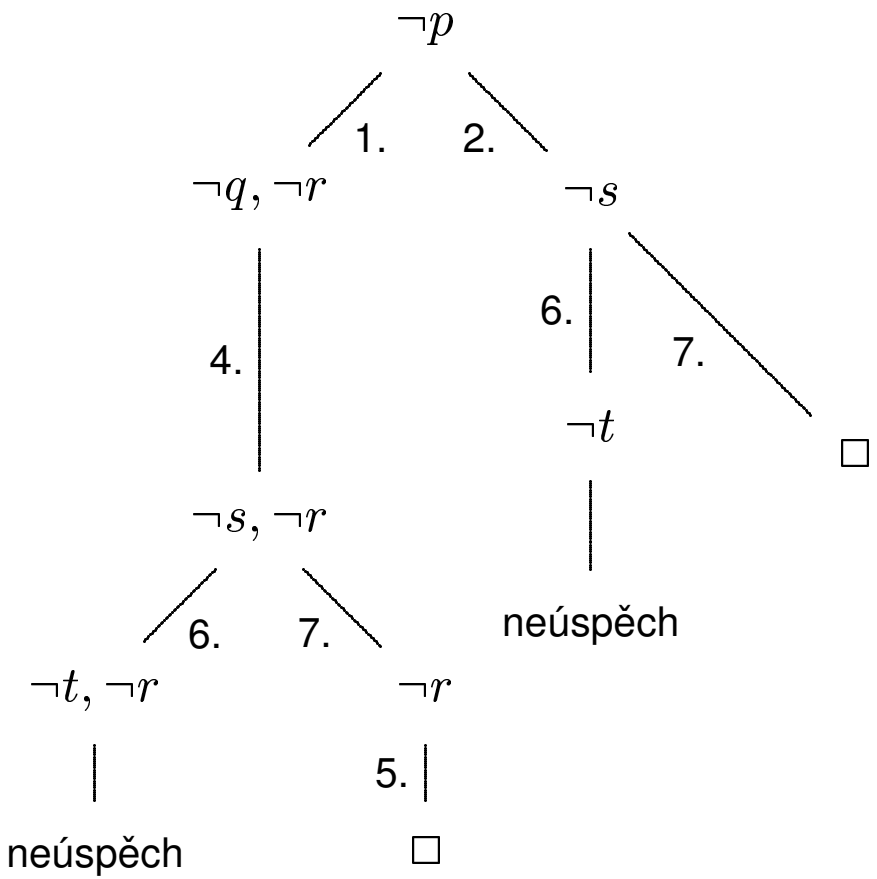


Backtracking

- *úspěšné cesty* v SLD-stromu jsou ty, které končí $\square$, ostatní cesty jsou neúspěšné
- Vyhodnocovací mechanismus Prologu prochází SLD-strom do hloubky zleva doprava a hledá (první) úspěšnou cestu *(backtracking)*.
  Existuje-li více programových klauzulí, se kterými může vybraný literál rezolvovat, vybere se první z těchto klauzulí (dle pořadí, jak jsou uvedeny v programu). Pokud mechanismus při procházení stromu narazí na neúspěšnou větev, dojde k backtrackingu – výpočet se vrátí zpět do nejbližšího uzlu N, kde je k dispozici možnost volby, a pokračuje z N cestou těsně vpravo od té, která právě selhala. Tento proces se opakuje, dokud není nalezena úspěšná větev (resp. do prozkoumání celého stromu).
- vyhodnocení cíle `?- p.` z předchozího příkladu projde hranami označenými 1.,3.,5. a skončí úspěchem (systém odpoví ‚yes‘).

Backtracking: příklad

- Odstraníme-li z předchozího programu 3. klauzuli, získáme pro cíl `?- p.` následující SLD-strom. Vyhodnocení projde přes 1.,4.,6., narazí na neúspěch, vrací se do $\neg s, \neg r$, projde hranami 7.,5. a úspěšně skončí.



Prolog: poznámky

- zadáním středníku (;) po úspěšném vyhodnocení cíle vynutíme backtracking a hledání alternativního důkazu
- odpověď ‚no' systému znamená, že daný cíl není logickým důsledkem programu (resp. že nemá alternativní důkaz – po předchozím zadání středníku)
- prologovská strategie prohledávání stavového prostoru může vést k zacyklení (i v případě, že existují úspěšné větve)

  příklad: program

  ```
  q:- r.
  r:- q.
  q.
  ```

  cyklí pro zadaný cíl `?- q.`