Occur-check
(Test konfliktu proměnných)

Petr Štěpánek

S využitím materialu Krysztofa R. Apta

2007

Při výpočtech v Prologu většina implementací používá zjednodušený algoritmus unifikace, při unifikaci rovnosti `x = t` se neprovádí test zda proměnná `x` nemá výskyt v termu `t`.

Toto zjednodušení se používá i při implementaci unifikačním algoritmu Martelliho a Montanariho který jsme adoptovali i do čistého Prologu.

Výsledkem je nekorektní použití unifikace a pro následné komplikace se obvykle používá anglický termín “Occur-check” , česky asi “Test konfliktních proměnných”.

Pro ilustraci se často používá následující příklad přerušeného výpisu:

```
?- X = f(X).

X = f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f
```

Predikát `=/2` je vestavěný a v Prologu je vnitřně definován jedinou klauzulí:

```
% X = Y <- X a Y jsou unifikovatelné.

X = X.
```

Povšimněme si, že dotaz $s = t$ pro termy s, t je úspěšný, právě když pro libovolnou proměnnou $x \notin Var(s, t)$ je množina rovností

$$\{ x = s , x = t \} \qquad (1)$$

unifikovatelná, ale $\{x/s\}$ je nejobecnější unifikace pro $\{x = s\}$ a iterací dostáváme, že množina (1) je unifikovatelná, právě když je unifikovatelná množina

$$\{ s = t \}$$

tedy, právě když se termy s, t unifikují.

To vysvětluje řádek komentáře ve výše uvedeném programu.

Naším cílem je najít podmínky, za kterých problém testu konfliktních proměnných nenastává a naznačit jak máme postupovat, když tento problém nastane.

Nelze očekávat úplné řešení tohoto problému, protože problém testu konfliktních proměnných je v obecném případě nerozhodnutelný.

Podívejme se blíže na důsledky vynechání testu konfliktních proměnných "Occur-checku" v unifikačním algoritmu Martelliho a Montanariho.

Při pokusu o unifikaci rovnosti $x = t$ v Akci (5) se vynechá test

$$x \in Var(t)$$

a následně i Akce (6).

Pokud je test negativní, zjednodušený algoritmus vydá nejobecnější unifikaci, je-li pozitivní máme ještě dvě (špatné) možnosti:

• provést nabízenou substituci $\{x/t\}$ i na rovnost $x = t$,

• nabízenou substituci neprovést. .

V prvním případě hrozí divergence protože proměnná x je v t a tedy i v $t\{x/t\}$.

Ve druhém případě může nekorektní výsledek vzniknout jako v případě vyřešené rovnosti $x = f(x)$, která určuje substituci $\{x/f(x)\}$ v uvedeném příkladu.

Proto nejprve hledáme podmínky, které zaručují, že i při absenci Occur-checku bude unifikace korektně provedena ve všech Prologovských výpočtech pro daný program a daný dotaz.

Jde tedy o to, zajistit aby v takových výpočtech nemohla být provedena Akce (6) Martelliho a Montanariho algoritmu. V těchto případech nezáleží na tom, kterou z nabízených modifikací unifikačního algoritmu zvolíme, protože obě se chovají stejným způsobem jako původní unifikační algoritmus.

Úmluva. V dalším budeme vynechávat slovo "term" budeme-li mluvit o rovnostech termů, a použijeme znovu značení, které pro dva atomy

$$A \equiv p(t_1, t_2, \ldots, t_n) \quad \text{a} \quad H \equiv p(s_1, s_2, \ldots, s_n)$$

se stejným predikátem, bude množinu rovností

$$\{t_1 = s_1, t_2 = s_2, \ldots, t_n = s_n\}$$

zapisovat jako $A = H$.

Definice. (NSTO množiny)

Říkáme že množina rovností E nepodléhá occur-checku, nebo krátce, že E je NSTO (Not Subject To Occur-chceck), jestliže v žádném výpočtu Martelliho a Montanariho algoritmu pro množinu E nedojde k provední Akce (6).

Definice. (LD-derivace nezávislé na occur-checku)

Nechť P je program a Q je dotaz.

(i) Nechť ξ je LD-derivace pro P a atom A vybraný v ξ. Nechť H je varianta hlavy nějaké klauzule z P a A a H mají stejný predikát. Potom říkáme, že množina rovností $A = H$ je dostupná v ξ.

(ii) Předpokládejme, že všechny množiny rovností dostupné ve všech LD-derivacích pro $P \cup Q$ jsou NSTO, potom říkáme, že $P \cup Q$ nezávisí na occur-checku.

• pojem nezávislosti na occur-checku nezávisí na pořadí kluzulí.

• definice předpokládá určitý algoritmus unifikace, ale dovoluje odvodit korektní důsledky.

• nedeterminismus MM unifikačního algoritmu však dovoluje modelovat výpočty různých jiných algorimů unifikace.

• v LD-derivaci ξ se nepředpokládá žádný určitý unifikační algoritmus.

• z vět o unifikaci vyplývá, že dostupná množina rovností, která je unifikovatelná, je NSTO. Nezávislost na occur-checku je tedy záležitostí dostupných množin, které nejsou unifikovatelné.

• v definici nezávislosti se uvažují všechny LD-derivace pro $P \cup Q$, tedy se uvažují všechny dostupné množiny rovností, které mohou být použity při navracení (backtrackingu).

Definice. (Lineární množiny termů)

(i) Množina termů je *lineární*, jestliže každá proměnná má v ní nejvýše jeden výskyt.

(ii) Množina rovností termů je *zleva lineární*, jestliže množina termů na levých stranách je lineární.

Definice. (Vztahy proměnných v rovnostech)

Nechť E je množina rovností, na E *definujeme relaci* $->_E$ následovně : $e_1 ->_E e_2$ jestliže levá strana rovnosti e_1 a pravá strana rovnosti e_2 mají společnou proměnnou.

Například $x = f(y) ->_E a = x$ také $e ->_E e$, jestliže rovnost e má na obou stranách stejnou proměnnou.

Lemma. (NSTO)

Předpokládejme, že rovnosti v E mohou být přeorientovány tak, že vzniklá množina rovností F je zleva lineární a relace $->_F$, nemá cykly. Potom E *je* NSTO.

Důkaz. Pokud množina rovností splňuje předpoklady lemmatu, říkáme krátce, že je dobrá. Důkaz se zakládá na dvou Pozorováních.

Pozorování 1. Dobrost se zachovává akcemi Martelliho a Montanariho algoritmu.

Pozorování 1. se dokazuje pro každou Akci 1, 3, 4 a 5 zvlašť, jak bychom předpokládali, nejkomplikovanější bude důkaz pro Akci 5. Tyto důkazy nebudeme provádět.

Pozorování 2. Je-li množina rovností dobrá, potom Akce 6. na ní nemůže být použita.

Důkaz. Uvažujme rovnost $x = t$, kde x se vyskytuje v t . Potom pro libovolnou množinu rovností E , platí

$$x = t \; ->_{E \cup \{x = t\}} \; x = t \quad \text{a} \quad t = x \; ->_{E \cup \{t = x\}} \; t = x$$

Odtud plyne, že pokud na množinu E lze použít Akci 6. potom každá množina rovností F , která vznikne přeorientováním rov ností v E má cyklus. Tedy E není dobrá.

Důkaz NSTO lemmatu bezprostředně plyne z obou pozorování.

Malé zobecnění. Říkáme, že rovnost je napůl základní, jestliže jedna její strana je základní. Množina rovností je napůl základní, pokud je taková každá její rovnost. Platí:

je-li E_1 napůl základní, pak pro libovolnou množinu rovností E_2 platí

$E_1 \cup E_2$ je NSTO, právě když E_2 je NSTO.

Zatím jsme se zabývali nezávislostí na Occur-checku rozborem možných unifikací. Nyní se budeme zabývat nezávislostí v souvislosti s tokem dat.

Správně modované dotazy a programy

Mody indikují, jak mohou být argumenty predikátového symbolu při výpočtu použity.

Definice. (Mody)

Mějme n-árním predikátový symbol p. Modem pro p je funkce, m_p která každému pořadí i argumentu p přiřadí hodnotu + nebo -.

Je-li $m(i)_p = +$, říkáme, že i je *vstupní pozice* p, pokud $m(i)_p = -$, říkmáme, že i je *výstupní pozice* predikátu p vzhledem k m_p.

Modováním rozumíme soubor funkcí m_p, kde každá přísluší k jinému predikátu.

Funkci m_p, zapisujeme v predikátu p názorným způsobem

$$p(m_p(1), m_p(2), \ldots, m_p(n)) .$$

Například `member(-,+)` označuje binární predikát `member`, s první pozicí modovanou jako výstupní a druhou jako vstupní.

Je-li třeba stejný predikát modovat několika způsoby, řešíme to přejmenováním daného predikátu.

V dalším budeme vždy předpokládat, že

Ke každému predikátu jednoznačně přiřazen mod.

Tento předpoklad nám dovolí mluvit o vstupních a výstupních pozicích daného atomu. Tento rozdíl nemusí být vždy zřejmý, jsou-li všechny pozice instancovány složenými termy.

Nyní zavedeme omezení na "tok dat" dotazem a klauzulemi programu. Pro zjednodušení budeme psát $p(\mathbf{u},\mathbf{v})$ a budeme předpokládat, že $\mathbf{u}$ je posloupnost termů na vstupních pozicích a $\mathbf{v}$ je posloupnost temů na výstupních pozicích predikátu p .

Definice. (Dobře modované dotazy a klauzule)

(i) Dotaz $p_1(\mathbf{s}_1,\mathbf{t}_1), p_2(\mathbf{s}_2,\mathbf{t}_2), \ldots , p_n(\mathbf{s}_n ,\mathbf{t}_n)$, je *dobře modovaný*, jestliže pro každé $i , 1 \leq i \leq n$ platí

$$Var(\mathbf{s}_i) \subseteq \cup\{Var(\mathbf{t}_j) \mid 1 \leq j \leq i\text{-}1\}$$

(ii) Klauzule

$$p_0(\mathbf{t}_0,\mathbf{s}_{n+1}) \text{ <- } p_1(\mathbf{s}_1,\mathbf{t}_1), p_2(\mathbf{s}_2,\mathbf{t}_2), \ldots , p_n(\mathbf{s}_n ,\mathbf{t}_n)$$

je *dobře modovaná*, jestliže pro $i , 1 \leq i \leq n+1$ platí

$$Var(\mathbf{s}_i) \subseteq \cup\{Var(\mathbf{t}_j) \mid 0 \leq j \leq i\text{-}1\} .$$

(iii) Program je *dobře modovaný*, je-li dobře modovaná každá jeho klauzule.

Jinými slovy: dotaz je dobře modovaný, jestliže

• každá proměnná na vstupní pozici atomu $(i , 1 \leq i \leq n)$ se vyskytuje na výstupní pozici nějakého dřívějšího atomu $(j, 0 \leq j \leq i\text{-}1)$,

a klauzule je dobře modovaná, jestliže

• pro $i , 1 \leq i \leq n$ každá proměnná v i-tém atomu těla klauzule se vyskytuje buď na výstupní pozici hlavy ($j = 0$) nebo na výstupní pozici nějakého dřívějšího atomu $j , 0 \leq j \leq i\text{-}1$.

• pro i = n+1 každá proměnná na výstupní pozici hlavy klauzule se vyskytuje na vstupní pozici hlavy (j = 0) nebo na výstupní pozici nějakého atomu těla klauzule (j , $1 \leq j \leq n$) .

Podle této definice je jednotková klauzule p (**s**,**t**) dobře modovaná, právě když $Var(\mathbf{t}) \subseteq Var(\mathbf{s})$ zatímco dotaz sestávající z jediného atomu je dobře modován, právě když je tento atom základní na svých vstupních pozicích.

• Intuitivně v prologovském výpočtu dobře modovaného dotazu

$$A_1 , A_2 , \ldots , A_n$$

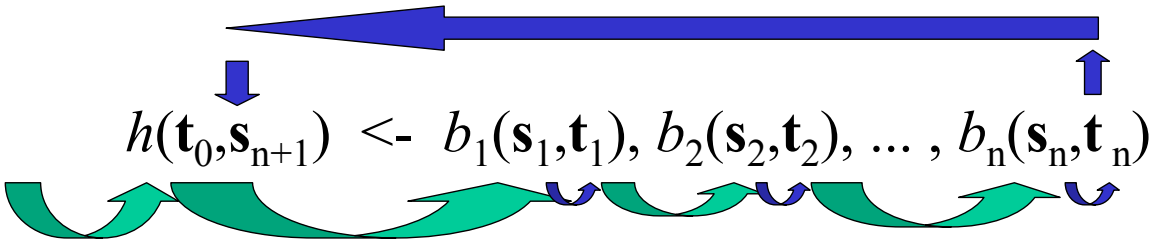
a dobře modovaného programu P "data" procházejí ze vstupních pozic atomu A_1 do výstupních pozic A_1, potom do vstupních pozic A_2 atd. dokud nedosáhnou výstupních pozic A_n .

$$A_1 \qquad A_2 \quad \ldots \quad A_n$$

$$q_1(\mathbf{s}_1,\mathbf{t}_1),\ q_2(\mathbf{s}_2,\mathbf{t}_2),\ \ldots\ , q_n(\mathbf{s}_n,\mathbf{t}_n),$$

Hypotetický tok dat dotazem $Q \equiv A_1 , A_2 , \ldots , A_n$

• Uvnitř každé dobře modované klauzule z programu P prologovský výpočet začíná přechodem dat do vstupních pozicí hlavy H, odkud přechází do vstupních pozic prvního atomu těla B_1 potom do výstupních pozic B_1 odtud do vstupních pozic B_2 atd. až dosáhne výstupní pozice B_n odkud nakonec přechází do výstupních pozic hlavy H.



Hypotetický tok dat klauzulí

$h(\mathbf{t}_0,\mathbf{s}_{n+1})$ <- $b_1(\mathbf{s}_1,\mathbf{t}_1), b_2(\mathbf{s}_2,\mathbf{t}_2), \ldots, b_n(\mathbf{s}_n,\mathbf{t}_n)$

Efektivní test zda je dotaz nebo klauzule dobře modovaná :

• v dotazu Q je první výskyt proměnné zleva uvnitř nějaké výstupní pozice,

• klauzule $p(\mathbf{s}, \mathbf{t})$ <- $\mathbf{B}$ je dobře modovaná, jestliže první výskyt proměnné zleva v posloupnosti $\mathbf{s}, \mathbf{B}, \mathbf{t}$ je součástí nějaké vstupní pozice v $p(\mathbf{s}, \mathbf{t})$ nebo je součástí nějaké výstupní pozice v $\mathbf{B}$.

Lemma. (Dobrá modovanost)

SLD-rezolventa dobře modovaného dotazu a dobře modované klauzule je dobře modovaná.

Důkaz. Jako v jiných případech vycházíme z toho, že SLD-rezolventa dotazu a klauzule je výsledkem tří operací:

instancí dotazu,
instancí klauzule a
nahrazením nějakého atomu dotazu, řekněme H tělem klauzule s hlavou H .

Lemma je důsledkem dvou pozorování.

Pozorování 1. Instance dobře modovaného dotazu, nebo klauzule je dobře modovaná.

Důkaz. Stačí si uvědomit, že pro libovolnou posloupnost termů $\mathbf{s}, \mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_n$ a substituci θ platí

$$Var(\mathbf{s}) \subseteq \cup\{Var(\mathbf{t}_j) \mid 1 \leq j \leq n\}$$

implikuje

$$Var(\mathbf{s}\theta) \subseteq \cup\{Var(\mathbf{t}_j\theta) \mid 1 \leq j \leq n\}$$

.

Pozorování 2. Je-li **A**, *H*, **C** dobře modovaný dotaz a *H* <- **B** dobře modovaná klauzule, potom **A**, **B**, **C** je dobře modovaný dotaz.

Důkaz. Označme

$\mathbf{A} = p_1(\mathbf{s}_1,\mathbf{t}_1), p_2(\mathbf{s}_2,\mathbf{t}_2), \ldots, p_k(\mathbf{s}_k,\mathbf{t}_k),$

$H = p(\mathbf{s},\mathbf{t}),$

$\mathbf{B} = p_{k+1}(\mathbf{s}_{k+1},\mathbf{t}_{k+1}), \ldots, p_{k+l}(\mathbf{s}_{k+l},\mathbf{t}_{k+l}),$

$\mathbf{C} = p_{k+l+1}(\mathbf{s}_{k+l+1},\mathbf{t}_{k+l+1}), \ldots, p_{k+l}(\mathbf{s}_{k+l+m},\mathbf{t}_{k+l+\mathrm{m}}).$

Pro každé i, $1 \leq i \leq k+l+m$ chceme dokázat

$$Var(\mathbf{s}_i) \subseteq \cup\{Var(\mathbf{t}_j) \mid 1 \leq j \leq i\text{-}1\}\ .$$

Uvažujme tři případy, podle toho, do kterého intervalu i patří.

1) $1 \leq i \leq k$ potom **A** je dobře modovaná protože **A**, *H*, **C** je taková.

2) $k+1 \leq i \leq k+l$. Protože H <- **B** je dobře modovaná klauzule, platí:

$$Var(\mathbf{s}_i) \subseteq \text{Var(s)} \cup \{\text{Var}(\mathbf{t}_j) \mid \text{k+1} < \text{j} < \text{i-1}\}$$

a protože **A**, H, **C** je dobře modované, také platí

$$Var(\mathbf{s}) \subseteq \cup\{Var(\mathbf{t}_j) \mid 1 \leq j \leq k\}$$

složením obou inkluzí dostáváme 2) .

3) $k+l+1 \leq i \leq k+l+m$. Protože **A**, H, **C** je dobře modovaný dotaz, platí

$$Var(\mathbf{s}_i) \subseteq \cup\{Var(\mathbf{t}_j) \mid 1 \leq j \leq k\} \cup Var(\mathbf{t}) \cup (\cup\{\text{Var}(\mathbf{t}_j) \mid k+l+1 < j < i\text{-}1\})$$

přitom také

$$Var(\mathbf{s}) \subseteq \cup\{Var(\mathbf{t}_j) \mid 1 \leq j \leq k\}.$$

protože H <- **B** je dobře modovaná klauzule, nakonec dostáváme

$Var(\mathbf{t}) \subseteq Var(\mathbf{s}) \cup (\cup\{\text{Var}(\mathbf{t}_j) \mid k+1 < j < k+l\})$ odtud plyne 3) .

Důsledek. (Dobrá modovanost)

Nechť P a Q jsou dobře modované, potom všechny dotazy a všechny dotazy (rezolventy) ve všech SLD-derivacích pro $P \cup \{Q\}$ jsou dobře modované.

Odbočka. Předchozí Důsledek dovoluje odvodit dvě tvrzení, která nejsou nutná pro studium dobře modovaných programů, ale mají vlastní význam.

Důsledek. (Vypočtená odpověď)

Nechť P a Q jsou dobře modované, potom každá vypočtená substituce θ pro $P \cup \{Q\}$ je základní.

Důkaz. Nechť **x** je posloupnost všech proměnných, které se vyskytují v Q. Nechť p je nový predikát, jehož četnost se rovná délce **x** a nechť všechny jeho proměnné jsou modované jako vstupní.

Potom Q , $p(\mathbf{x})$ je dobře modovaný dotaz, protože každá proměnná, která se vyskytuje v dobře modovaném dotazu má také výskyt na některé výstupní pozici.

Předpokládejme, že θ je vypočtená odpovědní substituce pro

$$P \cup \{Q\} .$$

Potom $p(\mathbf{x})\theta$ je dotaz v nějaké SLD-derivaci pro

$$P \cup \{Q, p(\mathbf{x})\} .$$

Potom podle předchozího důsledku je $p(\mathbf{x})\theta$ základní dotaz, protože je dobře modovaný. Odtud plyne tvrzení Důsledku.

Podle Věty o korektnosti, je každá vypočtená substituce korektní. Víme, že v obecném případě neplatí obrácená implikace. Je zajímavé, že v případě dobré modovatelnosti korektní a vypočtené substituce splývají.

Věta. (dobrá modovanost)

Nechť P a Q jsou dobře modované. Potom každá korektní substituce je také vypočtená odpovědní substituce.

Důkaz. Nechť θ je korektní substituce pro $P \cup \{Q\}$. Podle Silné věty o úplnosti existuje vypočtená odpovědní substituce η pro $P \cup \{Q\}$, taková, že $Q\eta$ je obecnější než $Q\theta$.

Podle předchozího důsledku je $Q\eta$ základní, tedy $Q\theta$ je také základní a $Q\theta$ a $Q\eta$ jsou totožné. Takže substituce θ a η splývají.

Příklady.

APPEND

```
app([],Ys,Ys).
app([X|Xs],Ys,[X|Xs]) <- app(Xs,Ys,Xs).
```

Je dobře modovaný při `app(+,+,-)`, protože platí

Var`(Ys)` $\subseteq$ Var`([],Ys)`,

Var`(Xs,Ys)` $\subseteq$ Var`([X|Xs],Ys)`,

Var`([X|Zs])` $\subseteq$ Var`([X|Xs],Ys)`Var`(Zs)`.

Podobně se zjistí, že `APPEND` je dobře modovaný i při následujících modech `app(-,-,+)`, `app(+,-,+)`, `app(-,+,+)`. Je dobře modovaný i při `app(+,+,+)` ale připouští jen základní vstupy.

Uvedené příklady ukazují na přirozená použití programu `APPEND`. Ale ne všechna použití tohoto programu vyhovují modování. Víme, že `APPEND` může spojovat seznamy, které nejsou základní. V takovém případě uvažované dotazy nejsou modované.

Například pro program `APPEND` a dotaz

Q = app`([X,2],[Y,U],[3,Z,0,Z])`

neexistuje vhodné modování. Přitom tento dotaz je úspěšný s vypočtenou odpovědní substitucí

θ = `{X/3,Z/2,Y/0,U/2}`

odpovědní instance $Q\theta$ je základní. Tento fakt však nelze odvodit z Věty o dobré modovanosti.

```
PERMUTACE
perm([],[]).
perm(Xs,[X|Ys]) <-
.                    app(X1s,[X|X2s],Xs),
.                    app(X1s,X2s,Zs),
.                    perm(Zs,Ys).
```

Při obvyklém použití tohoto programu chceme použít modování `perm(+,-)`, ale při shodném modování obou výskytů predikátu `app` nelze sestrojit dobré modování programu. Přitom při modování `app(-,-,+)` u prvního výskytu a `app(+,+,-)` u druhého, by byl program `PERMUTACE` dobře modovaný.

Toho lze dosáhnout přejmenováním druhého výskytu predikátu `app`.

Z dobré modovanosti pak dostáváme, že pro základní term `s` budou všechny vypočtené odpovědní substituce `θ` pro

`PERMUTACE ∪ {perm(s,t)}`

takové, že `tθ` bude základní.

Program `PERMUTACE` lze využít i pro výpočet permutací některých seznamů, které nejsou základní. V takovém případě však odpovídající dotaz není dobře modovaný.

Podobný problém nastává i u programu `SEKVENCE`.

Jaké výsledky pro nezávislost na Occur-checku můžeme od modovanosti očekávat ?

Definice. (vstupní a výstupní linearita)

Říkáme, že atom je vstupně (výstupně) lineární, jestliže množina termů vyskytujících se na vstupních (výstupních) pozicích je lineární.

Věta. (nezávislost na Occur-checku 1)

Nechť P a Q jsou dobře modované a hlava každé klauzule je výstupně lineární.

Potom $P \cup \{Q\}$ nezávisí na Occur-checku.

Definice. (lepší modovanost)

(i) Dotaz $p_1(\mathbf{s}_1,\mathbf{t}_1), p_2(\mathbf{s}_2,\mathbf{t}_2), \ldots, p_n(\mathbf{s}_n,\mathbf{t}_n)$ je lépe modovaný, jestliže množina $\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_n$ je lineární a pro i, $1 \leq i \leq n$ platí

$$Var(\mathbf{s}_i) \cap (\cup\{Var(\mathbf{t}_j) \mid i \leq j \leq n\}) = 0$$

(ii) Klauzule

$$p_0(\mathbf{s}_0, \mathbf{t}_0) \text{ <- } p_1(\mathbf{s}_1,\mathbf{t}_1), p_2(\mathbf{s}_2,\mathbf{t}_2), \ldots, p_n(\mathbf{s}_n,\mathbf{t}_n)$$

je lépe modovaná, jestliže její tělo jako dotaz je lépe modované a

$$Var(\mathbf{s}_0) \cap (\cup\{Var(\mathbf{t}_j) \mid 1 \leq j \leq n\}) = 0$$

Tedy předpokládáme-li, že vstupní pozice předcházejí výstupní, potom

• dotaz je lépe modovaný, jestliže každá proměnná na výstupní pozici nějakého atomu se nevyskytuje dříve v daném dotazu,

• klauzule je lépe modovaná, jestliže každá proměnná, na výstupní pozici nějakého atomu z jejího těla se nevyskytuje dříve v těle klauzule ani na vstupní pozici hlavy klauzule.

Intuitivně, pojem lepší modovanosti má zabránit „spekulativním vazbám“ mezi proměnnými, které se vyskytují na výstupních pozicích. Vyžaduje se, aby takové proměnné byly „nové“ v tom smyslu, že nebyly použity dříve než prologovský výpočet dosáhne na výstupní pozice, ve kterých se tyto proměnné vyskytují.

Věta. (nezávislost na Occur-checku 2)

Jsou-li P a Q lépe modované a hlava každé klauzule v P je vstupně lineární, potom $P \cup \{Q\}$ nezávisí na Occur-checku.