Několik poznámek o unifikaci

Pod pojmem *unifikace* je neformálně řečeno myšlena speciální transformace, která ze dvou obecně různých výrazů vytvoří identické výrazy, pokud je to možné. Cílem je navíc, aby výsledné identické výrazy byly co možné nejblíže výchozím výrazům a aby bylo možné něčím unifikaci jednoznačně určit. Ukažme si nejdřív motivační příklad, ...

Příklad. Uvažujme rovnici ve tvaru $2x + 3 = 4y + 7$. Pokud se podíváme na sémantiku této rovnice lehce nahlédneme, že má nekonečně mnoho řešení. Pokud bychom měli automatizovat proces „řešení rovnic" libovolných výrazů, nemůžeme sémantiku nikterak využít. Zkoumání sémantiky výrazů je zcela nad rámec možností jakéhokoliv výpočetního systému, protože vyžaduje pracovat s doménami či obory, které jsou obecně nekonečné. Nezbývá nám nic jiného, než se pokusit najít „syntaktické řešení rovnic" založené pouze na zkoumání struktury samotného výrazu bez ohledu na význam jednotlivých symbolů, operací atp.

Na druhou stranu vezmeme-li si výraz $f(x, h(y, y), g(z)) = f(x, u, g(v))$, jeho řešením bez ohledu na význam jednotlivých symbolů může být dvojice přiřazení $\{u = h(y, y), v = z\}$. Dosadíme-li totiž do výrazu na pravé straně výraz $h(y, y)$ za u a výraz z za v, obdržíme identické výrazy. V tomto duchu funguje i již jmenovaná unifikace. Každá substituce, jejíž aplikací na dva či více výrazů získáme identitu se nazývá unifikátor. Obzvlášť zajímavé vlastnosti má tzv. nejobecnější unifikátor, který je až na pojmenování proměnných určen jednoznačně. Na úvod poznamenejme, že unifikátor dvou výrazů nemusí vždy existovat. Intuitivně je zřejmé, že například výrazy $f(x)$ a $g(x, y)$ nelze ve výše uvedeném smyslu unifikovat.

Poznámka. V dalším textu se bude hojně vyskytovat pojem *výraz*. Ačkoliv bychom jej měli precizně definovat, ponecháme jej zcela na intuici. Při implementaci zápočtových příkladů chápejte jako výraz libovolný výraz jazyka Scheme, to jest výraz obsahující čísla, symboly a seznamy. V tomto textu budeme až na výjimky všechny výrazy zapisovat infixově, to jest například výrazu $f(x, y, p(h, x))$ by v jazyku Scheme odpovídal seznam `(f x y (p h x))`.

Definice. *Substituce* je konečná množina ve tvaru $\theta = \{v_1/t_1, \ldots, v_n/t_n\}$, kde $v_1, \ldots, v_n$ jsou navzájem různé proměnné a $t_1, \ldots, t_n$ jsou libovolné výrazy, $v_i \neq t_i$ pro každé $i = 1, \ldots, n$. Pokud jsou všechny $t_1, \ldots, t_n$ rovněž proměnné, substituce se nazývá *přejmenování.* Je-li E výraz, pak aplikací substituce θ na výraz E získáme výraz $E\theta$, ve kterém je každý výskyt libovolné proměnné v_i nahrazen výrazem t_i, nahrazení přitom probíhají *současně,* nikoliv postupně.

Poznámka. Rozdíl mezi současným a postupným nahrazením je zásadní. Uvažujme například výraz $f(x, y)$ a substituci $\theta = \{x/g(y), y/u\}$. Výsledkem substituce tak, jak ji chápeme my, je nový výraz $E\theta = f(g(y), u)$. Kdyby substituce probíhala v pořadí $x/g(y)$, pak y/u, obdrželi bychom výraz $f(g(u), u)$. Postupná substituce je pro účely unifikace nepoužitelná.

Úmluva. Substituci $\theta = \{v_1/t_1, \ldots, v_n/t_n\}$ budeme v jazyku Scheme representovat seznamem tečkových párů `((`v_1 `.` t_1`) (`v_2 `.` t_2`) ⋯ (`v_n `.` t_n`))`, v němž nezáleží na pořadí uvedených substitucí. To jest například substituce $\theta = \{x/g(y), y/u\}$ může mít tvar `((x . (g y)) (y . u))`.

Definice. Jsou-li $\theta = \{u_1/s_1, \ldots, u_m/s_m\}$, $\sigma = \{v_1/t_1, \ldots, v_n/t_n\}$, dvě substituce, pak definujeme *složení substitucí* $\theta\sigma$ jako substituci vzniklou z množiny

$$\{u_1/(s_1\sigma), \ldots, u_m/(s_m\sigma), v_1/t_1, \ldots, v_n/t_n\},$$

odstraněním $u_i/(s_i\sigma)$, pro které je $u_i = s_i\sigma$ a odstraněním v_j/t_j, pro které je $v_j \in \{u_1, \ldots, u_m\}$. Jednoduše lze ukázat, že množina všech substitucí tvoří vzhledem k operaci složení substitucí *monoid* – pologrupu s neutrálním prvkem $\epsilon = \{\}$, což je *prázdná substituce.*

Příklad. Mějme substituce $\theta = \{x/f(y), y/z\}$, $\sigma = \{x/a, y/b, z/y\}$. Pak $\theta\sigma = \{x/f(b), z/y\}$, $\sigma\theta = \{x/a, y/b\}$. Skládání substitucí není samozřejmě komutativní.

Definice. Mějme dva výrazy E_1, E_2, substituce θ se nazývá *unifikátor výrazů* E_1, E_2, pokud jsou výrazy $E_1\theta$, $E_2\theta$ *identické.* Unifikátor θ výrazů E_1, E_2 se nazývá *nejobecnější unifikátor,* pokud pro libovolný unifikátor σ výrazů E_1, E_2 existuje substituce τ taková, že platí $\sigma = \theta\tau$.

Příklad. Mějme výrazy $f(x, h(y, y), g(z))$, $f(x, u, g(v))$, jejich unifikátorem je například substituce $\sigma = \{y/g(x), u/h(g(x), g(x)), z/g(x), v/g(x)\}$, není to ale nejobecnější unifikátor. Nejobecnější unifikátor má tvar $\theta = \{u/h(y, y), v/z\}$. Přitom $\sigma = \theta\tau$, kde $\tau = \{y/g(x), z/g(x)\}$.

Úvaha. Jsou-li výrazy E_1, E_2 identické, jejich nejobecnějším unifikátorem je evidentně prázdná substituce $\epsilon = \{\}$. V opačném případě chceme nejobecnější unifikátor stanovit konstruktivně. K tomuto účelu musíme nejprve jednoznačně definovat rozdíl ve výrazech, abychom podle něj mohli vytvořit patřičnou substituci.

K tomuto účelu vypočteme dvojici nejlevějších rozdílů ve výrazech, tzv. *disagreement pair*. Uvažujme výrazy $f(x, h(y, y), g(z))$, $f(x, u, g(v))$. Nejlevější podvýrazy, ve kterých se oba dva výrazy liší jsou výrazy $h(y, y)$ a u. Tato dvojice výrazů $\langle h(y, y), u\rangle$ tedy tvoří disagreement pair výrazů $f(x, h(y, y), g(z))$, $f(x, u, g(v))$. V jazyku Scheme budeme disagreement pair representovat tečkovým párem.

Nyní si již můžeme popsat algoritmus nalezení nejobecnějšího unifikátoru.

Algoritmus (Nalezení nejobecnějšího unifikátoru).

Vstupem algoritmu jsou dva výrazy E_1, E_2.
Výstupem je nejobecnější unifikátor θ.

1. Položme $k = 0$ a $\theta_k = \epsilon = \{\}$.
2. Pokud je θ_k unifikátorem výrazů E_1, E_2, pak je $\theta = \theta_k$ *nejobecnějším unifikátorem* E_1, E_2. V opačném případě najdi disagreement pair D_k výrazů $E_1\theta_k$, $E_2\theta_k$ a pokračuj dalším bodem.
3. Pokud je D_k ve tvaru $\langle v, t\rangle$, či $\langle t, v\rangle$, kde v je proměnná *nevyskytující* se ve výrazu t, pak položme $\theta_{k+1} := \theta_k\{v/t\}$, $k := k + 1$ a dále pokračujeme krokem 2.
   V opačném případě nejsou výrazy E_1, E_2 unifikovatelné.

Upozorněme na fakt, že podmínka v posledním kroku, to jest že v není obsažena ve výrazu t je zásadní a nelze jí vynechat.

Implementační poznámky.
Zadané procedury naprogramujte přesně dle předchozího popisu algoritmů a v souladu se zadáním. Pokud jsou proceduře `disagreement-pair` předány identické výrazy, tedy výrazy, pro které není disagreement pair definován, procedura by měla vracet `#f`. Taktéž procedura `unify` by měla vracet `#f` v případě, že výrazy předané jako argumenty nejsou unifikovatelné.

Příklad. Výsledné použití procedury `unify` by mělo vypadat třeba následovně.

```
; ukázka unifikace
(unify '(p a x (h (g z))) '(p z (h y) (h y)) '(x y z))
(unify '(p (f a) (g x)) '(p y y) '(x y))

((z . a) (x h (g a)) (y g a))     ; výsledek vyhodnocení prvního výrazu
#f                                ; výsledek vyhodnocení druhého výrazu
```