

allCHEMIE – Programátorská dokumentace

Ondrej Profant

12. června 2011

Obsah

1	Úvod	3
2	Tvorba vlastního predikátu	3
2.1	Jednoduché podmínky	3
2.2	Rovnice (vyčíslení)	3
2.3	Kompletace	4
3	Soubory a logické členění programu	5
4	Dodatky	5
4.1	Editor	5

1 Úvod

Celý program je velmi úzce spjat s programovacím jazykem Prolog, který spadá do paradigmatu logického programování. To znamená, že kód se skládá z predikátů nad databází již pevně zadaných predikátů (faktů) a snaží se doplnit všechny proměnné. Nicméně věřím, že není nutné podrobně rozumět Prologu pro to, aby jste si AllCHEMIi vylepšili či pochopili, co dělá. Zvláště pak odborné části z chemie jsou vysloveně jednoduché, složitější může být občas aritmetika dopočítávání sloučenin.

Pokud Prolog znáte, tak celý kód je velmi jasně napsaný a nejsou v něm žádné zaludnosti ani složitá rekurze.

V další kapitole je podrobně popsána tvorba predikátu pro výpočet oxidu.

2 Tvorba vlastního predikátu

2.1 Jednoduché podmínky

První a velmi důležitým rozhodnutím je, jak vlastně bude náš predikát deklarován. Celý program dodržuje konvenci, že všechny výpočty jsou deklarovány jako predikát `vstup()`, ale jak určit jaké bude mít argumenty již není tak jednoduché. Pojdme si vytvořit pravidla pro oxidy.

Oxid bude mít jistě 4 argumenty, dva budou prvky a dva počty výskytů. Navíc víme, že druhý prvek bude vždy kyslík. Oxid tedy bude zdefinován takto:

```
vstup( Prvek , N , o , M ). % zatím bez jakýchkoliv pravidel
```

Důležité je dodržovat zásady jazyka Prolog – proměnná je vždy velkým písmenem. Za znakem procenta je komentář, tam si můžete napsat libovolnou poznámku. Dalším jednoduchým omezením (omezujeme množinu všech oxidů na ten, který nám byl zadán) jsou číselné rozsahy. Víme, že kationt se v oxidu vyskytuje jednou nebo dvakrát a oxid může: 1,2,3,4,5,7. To zapíšeme takto:

```
member(N, [1,2]),  
member(M, [1,2,3,4,5,7]),
```

Další věcí, kterou můžeme snadno ověřit jsou informace o prvku. Na to máme predikát `prvek()/6` (podrobný popis je např. v uživatelské příručce). U oxidu ověříme, že prvek není v A8 skupině a jakých může nabývat oxid. čísel. Také si rovnou najdeme jeho plné jméno.

```
prvek( _, Prvek, Jmeno, _, Sk, Cisla ),  
not(Sk == a8), % dulezite je male a v a8, jinak by se jednalo o promennou
```

V jiných případech ještě můžeme ověřit zda je prvek kov, polokov či nekov, zde to nemá smysl. A nyní stačí již jen dopočítat zbylé oxidační číslo prvku.

2.2 Rovnice (vyčíslení)

Nejdříve budeme pokračovat v předchozím příkladu výpočtu oxididů a pak se zmíním o obecnějších věcech.

U oxidu není složitého dopočítat poslední číslo, neb víme:

```
Pstrana is(abs(-2*M)), %absolutni hod. prave strany
X is(Pstrana / N), % (Lstrana := X*N) == Pstrana
nabyva0xCisla(Jmeno,X),
```

Pokud dojdeme až sem, tak jsme korektně vypočísлили oxid, nyní ho stačí vypsát:

```
!, %tzv. řez, zabraní Prologu hledat další alternativy
priponSt(X,Koncovka),
write('oxid '),write(Jmeno),write('-'),write(Koncovka).
```

Aritmetika není zrovna silnou a oblíbenou stránkou Prologu. Celou dobu jsme vlastně pracovali s textovými řetězci (včetně: `member(N,[1,2,3])`) a až speciální predikát `is()`¹ převedl řetězce na čísla a začal s nimi počítat. Jak vidíte, tak se chová normálně operátory jsou: `+`, `-`, `*`, `/`, `//` (celočíslné), `==` (rovná se), `=\=` (nerovná se).

2.3 Kompletace

Nyní doděláme nezbytnosti okolo, ještě budeme potřebovat:

```
:-consult('tabulka.pro'). %pripoji tabulku prvků.
```

Celý kód bude tedy vypadat takto:

```
:-consult('tabulka.pro'). %pripoji tabulku prvků.
```

```
vstup( Prvek, N, o, M ) :-
    member(N,[1,2]),
    member(M,[1,2,3,4,5,7]),
    prvek( _, Prvek, Jmeno, _, Sk, Cisla ),
    not(Sk == a8), % dulezite je male a v a8, jinak by se jednalo o promennou
    Pstrana is(abs(-2*M)), %absolutni hod. prave strany
    X is(Pstrana / N), % (Lstrana := X*N) == Pstrana
    nabyva0xCisla(Jmeno,X),
    !, %tzv. řez, zabraní Prologu hledat další alternativy
    priponSt(X,Koncovka),
    write('oxid '),write(Jmeno),write('-'),write(Koncovka).
```

Co s kódem udělat? Otevřeme si libovolný textový editor (ve Windows třeba Notepad) a vložíme ho. Soubor uložíme např. jako `mujOxid.pro` a ve stejné složce máme soubor `tabulka.pro`. Pak již soubor `mujOxid.pro` stačí spustit v Linuxu z terminálu:

```
prolog -f mujOxid.pro
```

ve Windows poklepáním na daný soubor. Samozřejmě je nutné mít nainstalovaný SWI-Prolog.

¹ Dokumentace SWI-Prologu: <http://www.swi-prolog.org/pldoc/refman/>

3 Soubory a logické členění programu

Soubor	Obsah
tabulka.pro	Obsahuje definice prvků a koncovek
main.pro	Úvodní obrazka, integrovaná nápověda a kompozice zbylých souborů
parser.pro	Predikáty pro převod z chemického vzorce na slovní vyjádření
nazvy.pro	Opačný převod
anionty.pro	Definice aniontů (jak pevně definované, tak generické)
kompozer.pro	Skládání kombinací pro sestavování sloučenin
allchemie.sh	Jednoduchý shellovský skript pro spuštění v UNIXových systémech

4 Dodatky

4.1 Editor

Pokud hledáte editor, který by byl schopen zvýrazňovat syntaxi u Prologu (či dokonce napovídat), tak vás musím poněkud zklamat. Nejlepší podporu mají editory Gedit (standardní textový editor v Gnome) a VIM a ty pouze dosti neohrabaně zvýrazňují část syntaxe.