

# allCHEMIE – Programátorská dokumentace

Ondrej Profant

23. května 2011

# Obsah

|          |   |          |
|----------|---|----------|
| <b>1</b> | <b>Úvod</b>                               | <b>3</b> |
| <b>2</b> | <b>Tvorba vlastního predikátu</b>         | <b>3</b> |
| 2.1      | Jednoduché podmínky . . . . .             | 3        |
| 2.2      | Rovnice (vyčíslení) . . . . .             | 3        |
| 2.3      | Kompletace . . . . .                      | 4        |
| <b>3</b> | <b>Soubory a logické členění programu</b> | <b>4</b> |

# 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 zadaných predikátů a snaží se doplnit všechny proměnné. Nicméně věřím, že není nutné rozumět Prologu pro to, aby jste si AllCHEMI 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 zadefinován 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. 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 vypsat:

```
!, %tzv. řez, zabrání 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()`<sup>1</sup> 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.pl'). %pripoji tabulku prvků.
```

Celý kód bude tedy vypadat takto:

```
:-consult('tabulka.pl'). %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, zabrání 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 `muj0xid.pl` a ve stejné složce máme soubor `tabulka.pl`. Pak již soubor `muj0xid.pl` stačí spustit v Linuxu z terminálu:

```
prolog -f muj0xid.pl
```

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

## 3 Soubory a logické členění programu

<sup>1</sup> Dokumentace SWI-Prologu: <http://www.swi-prolog.org/pldoc/man?predicate=is%2f2>