Programowanie w Logice
Środowisko programowania i dialog z systemem (Lista 0)

Przemysław Kobylański

Wstęp

Podczas laboratorium będziemy korzystać z systemu SWI-Prolog. Należy go ściągnąć i zainstalować na prywatnych komputerach ze strony **swi-prolog.org**.

Dalszy opis będzie przedstawiony na przykładzie instalacji pod systemami operacyjnymi Linux i OSX.

W przypadku Linuxa, Prolog dostępny jest w wielu dystrybucjach w postaci pakietu `swipl`. W wyniku jego instalacji program wykonywalny `swipl` zostaje umieszczony w katalogu `/usr/bin` i jest gotowy do uruchamiania.

W przypadku OSX, w katalogu `/Applications` pojawia się nowa aplikacja `SWI-Prolog`. Po jej uruchomieniu zaczyna działanie konsola `swipl-win`. Aby uruchamiać program `swipl` na terminalu, należy w pliku `.bash_profile`, znajdującym się w katalogu domowym, na zmiennej środowiskowej `PATH` dopisać ścieżkę do katalogu:

`/Applications/SWI-Prolog.app/Contents/MacOS`.

Na system programowania Prolog składa się między innymi program `swipl`, który kompiluje programy w Prologu na instrukcje abstrakcyjnej maszyny WAM (Warren Abstract Machine), oraz wykonuje je symulując działanie tej maszyny.

System uruchamia się poleceniem:

```
$ swipl
Welcome to SWI-Prolog (Multi-threaded, 64 bits, Version 7.2.3)
Copyright (c) 1990-2015 University of Amsterdam, VU Amsterdam
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to redistribute it under certain conditions.
Please visit http://www.swi-prolog.org for details.

For help, use ?- help(Topic). or ?- apropos(Word).

?-
```

Po krótkim przedstawieniu się system drukuje zachętę w postaci znaków `?-`, za którymi oczekuje wpisania pytania.

Każde pytanie musi być zakończone kropką. Jeśli zapomnimy o jej wpisaniu, system będzie kontynuował czytanie pytania w kolejnych wierszach zmieniając zachętę na pionową kreskę:

```
?- X is
|    2 +
|    2.
X = 4.
```

W powyższym przykładzie, napis `X = 4.` jest udzieloną przez system odpowiedzią na wprowadzone pytanie.

We wprowadzanych pytaniach możemy używać przecinka jako spójnika dla koniunkcji i średnika jako spójnika dla alternatywy.

Poniższy przykład wylicza pięć początkowych wyrazów ciągu Fibonacciego `F0, F1, F2, F3, F4`:

```
?- F0 is 0, F1 is 1, F2 is F0+F1, F3 is F2+F1, F4 is F3+F2.
F0 = 0,
F1 = F2, F2 = 1,
F3 = 2,
F4 = 3.
```

Jak widać powyżej, wyrazy `F1` i `F2` są sobie równe.

Należy zwrócić uwagę na to, że zmienne w Prologu nazywane są z wielkiej litery (poza pewnymi wyjątkami o których będzie później).

Korzystaliśmy dotychczas z dwuargumentowego predykatu `is/2` zapisywanego infiksowo, to jest podawaliśmy jego nazwę między dwoma jego argumentami.

Więcej informacji o predykacie `is/2` i innych dostępnych predykatach arytmetycznych znajduje się w rozdziale Arithmetic

Należy pamiętać o tym, że zmienne w Prologu zachowują się bardzo specyficznie. Otóż po związaniu wartości ze zmienną nie ma możliwości zmienienia jej. Zatem **zmienne** w Prologu **nie są zmienne**. Jest to własność charakterystyczna dla języków programowania w logice i programowania funkcyjnego.

Przykład próby zmiany wartości zmiennej:

```
?- X is 0, X is X+1.
false.
```

Odpowiedź negatywna bierze się stąd, że oczekujemy spełnienia jednocześnie dwóch sprzecznych warunków `X = 0` i `X = 1`.

Zmienna może przyjąć nową wartość dopiero po wycofaniu się przed miejsce, w którym została związana i wybraniu innej ścieżki wnioskowania.

Dylemat Hamleta[1] zapisać możemy w postaci następującego pytania:

[1] William Shakespeare. „Hamlet, książę Danii". Akt III, scena 1.

```
?- Decyzja = być; Decyzja = nie_być.
Decyzja = być ;
Decyzja = nie_być.
```

Po ukazaniu się pierwszej odpowiedzi można wprowadzić znak średnik by polecić Prologowi szukanie kolejnych odpowiedzi. W trakcie szukania Prolog wycofuje się przed miejsce gdzie dokonało się związanie `Decyzja = być` i wybiera drugą ścieżkę wnioskowania, na której następuje związanie `Decyzja = nie_być`.

Wartości `być` oraz `nie_być` są pisane z małej litery, zatem reprezentują prologowe stałe.

Podstawową operacją w Prologu jest unifikacja termów, przy czym termami nazywamy zmienne, stałe i termy postaci $f(t_1, \ldots, t_n)$, gdzie f jest funktorem spinającym termy $t_1, \ldots, t_n$ w jeden term złożony.

Przykładem termu złożonego jest `para(a, b)`, w którym połączono dwie stałe `a` i `b` w uporządkowaną parę.

Zunifikować dwa termy, to rozstrzygnąć czy da się podstawić za zmienne w nich występujące takie termy, że po podstawieniu dane dwa termy będą identyczne.

Dla przykładu term `para(a, X)` i term `para(Y, b)` będą identyczne gdy za `X` podstawi się stałą `b` a za `Y` podstawi się stałą `a`.

Z drugiej strony nie jest możliwe zunifikowanie termu `para(a, b)` z termem `para(X, X)`, gdyż za zmienną `X` należałoby jednocześnie podstawić stałą `a` i różną od niej stałą `b`.

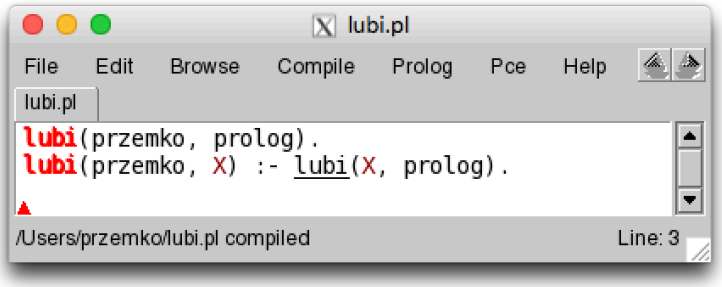
Do unifikowania termów służy w Prologu predykat `=/2` zapisywany infiksowo:

```
?- para(a, X) = para(Y, b).
X = b,
Y = a.

?- para(a, b) = para(X, X).
false.

?- para(Y, para(b, c)) = para(X, X).
Y = X, X = para(b, c).

?- para(a, b) = para(X, Y), Z = para(Y, X).
X = a,
Y = b,
Z = para(b, a).
```

Programy w Prologu można edytować dowolnym edytorem tekstowym. Osobiście polecam edytor Emacs zintegrowanego z SWI-Prologiem (wymaga trybu okienkowego).

Aby edytować plik należy po zachęcie `?-` wpisać pytanie:

```
emacs('nazwa_pliku.pl').
```



Rysunek 1: Okno edytora Emacs.

Zwróć uwagę na konieczność ujęcia nazwy pliku między apostrofy i użycie domyślnego dla SWI-Prologu rozszerzenia `.pl`.

Utwórzmy plik **lubi.pl**, w którym wyrazimy relację kto co lubi.

W tym celu wydajmy Prologowi polecenie:

```
?- emacs('lubi.pl').
true.
```

Po chwili ukazuje się okno edytora. Wpisz w nim dwie linijki programu jak na rysunku 1.

Aby skompilować program wybierz opcję `Compile buffer` z menu `Compile`.

Po skompilowaniu programu można powrócić do terminala i zadać Prologowi pytanie, kogo lubi Przemko:

```
?- lubi(przemko, X).
X = prolog ;
X = przemko ;
false.
```

Jak widać Przemko (wyrażony stałą `przemko`) lubi tylko Prolog i samego siebie. Dlaczego?

Otóż wpisany przez nas program składa się z dwóch klauzul (zdań):

- Faktu wyrażającego to, że Przemko lubi Prolog.
- Reuły wyrażającej implikację, że jeśli jakiś `X` lubi Prolog, to Przemko lubi tego `X`.

Nic więc dziwnego, że przy tak sformułowanym programie (Przemko lubi Prolog i tylko tych, którzy też lubią Prolog), wynika, że Przemko lubi tylko Prolog i samego siebie[2].

[2]Być może są jeszcze inne osoby lubiące Prolog, ale zgodnie z **zasadą zamkniętego świata**, prawdziwe jest tylko to, co wyrażono w programie.

Okno edytora można zamknąć wybierając opcję `Quit` z menu `File`.

Jeśli w bieżącym katalogu (można go sprawdzić wydając Prologowi polecenie `pwd`) znajduje się plik z programem (zawartość bieżącego katalogu można obejrzeć wydając polecenie `ls`), to można skompilować go podając nazwę pliku w kwadratowych nawiasach (można pominąć domyślne rozszerzenie nazwy):

```
?- [lubi].
true.
```

To samo można zrobić korzystając z predykatu `consult/1`:

```
?- consult(lubi).
true.
```

Aby zakończyć pracę z systemem Prolog należy wydać polecenie `halt`:

```
?- halt.
$
```

Polecenia

Polecenie 1

Użyte predykaty

Predykat `true` wyraża prosty warunek, który jest zawsze spełniony.

Wpisz cel

```
?- true.
```

Wyjaśnienie

System odpowiada **true** gdyż warunek `true` jest spełniony.

Polecenie 2

Użyte predykaty

Predykat `fail` zawsze zawodzi.

Wpisz cel

```
?- fail.
```

Wyjaśnienie

System odpowiada **false** ponieważ warunek `fail` zawiódł.

Polecenie 3

Użyte predykaty

Do wyrażenia koniunkcji warunków stosuje się przecinek.

Wpisz cel

```
?- true, fail.
```

Wyjaśnienie

System odpowiada **false** ponieważ nie jest prawdą, że oba warunki `true` i `fail` zachodzą.

Polecenie 4

Użyte predykaty

Do wyrażenia alternatywy warunków stosuje się średnik.

Wpisz cel

```
?- true; fail.
```

Wyjaśnienie

System odpowiada **true** ponieważ przynajmniej jeden z warunków `true` i `fail` zachodzi.

Po wydrukowaniu **true** system czeka na naszą decyzję czy poszukiwać kolejnych odpowiedzi (w tym celu trzeba nacisnąć średnik) czy zakończyć poszukiwania (w tym celu trzeba nacisnąć klawisz **[Enter]**).

Jeśli naciśniemy średnik, to system wydrukuje **false** bo nie ma innej alternatywnej odpowiedzi twierdzącej.

Polecenie 5

Użyte predykaty

Warunki można grupować korzystając z nawiasów.

Wpisz cel

```
?- (true, fail); (fail; true).
```

Wyjaśnienie

System odpowiada **true** ponieważ chociaż pierwszy człon alternatywy (koniunkcja `true, fail`) zawodzi, to jednak drugi jej człon (alternatywa `fail; true`) jest spełniony.

Polecenie 6

Użyte predykaty

W Prologu można korzystać ze zmiennych, przy czym ich nazwy pisze się z wielkiej litery.

Predykat `=` wyraża równość (dokładniej ich unifikowalność).

Wpisz cel

```
?- X = 1.
```

Wyjaśnienie

System odpowiada `X = 1` ponieważ liczba 1 jest jedyną wartością spełniającą warunek `X = 1`.

Polecenie 7

Wpisz cel

```
?- X = 1, X = 2.
```

Wyjaśnienie

System odpowiada **false** ponieważ żadna wartość nie jest jednocześnie równa dwóm różnym liczbom 1 i 2.

Polecenie 8

Wpisz cel

```
?- X = 1; X = 2.
```

Wyjaśnienie

System drukuje odpowiedź `X = 1` i czeka na to co zrobi użytkownik. Jeśli naciśniesz klawisz **[Enter]**, to pojawi się zachęta do wpisania następnego pytania. Jednak wartość 1 nie jest jedyną spełniającą alternatywę dwóch warunków $X = 1 \vee X = 2$. Jeśli naciśniemy średnik, to system poszuka drugiej odpowiedzi i pojawi się `X = 2`.

Polecenie 9

Użyte predykaty

Do obliczania wartości wyrażenia arytmetycznego służy predykat `is`. W wyrażeniu arytmetycznym mogą wystąpić operacje `+, -, *, /, **` (podnoszenie do potęgi), stałe liczbowe oraz zmienne, którym nadano już wartości będące liczbami.

Wpisz cel

```
?- X is 1, Y is X+1.
```

Wyjaśnienie

System udziela odpowiedzi `X = 1, Y = 2`. Najpierw pod zmienną `X` została podstawiona wartość 1 (zmienna `X` została zunifikowana z wartością 1), a następnie obliczona wartość wyrażenia `1+1` została zunifikowana ze zmienną `Y`.

Polecenie 10

Użyte predykaty

Dwuargumentowe predykaty `=:=`, `=\=`, `<`, `>`, `=<`, `>=` służą do porównywania wartości dwóch wyrażeń arytmetycznych (umieszcza się je między wyrażeniami).

Wpisz cel

```
?- (X is 1; X is 2), 2*X > 3.
```

Wyjaśnienie

System poszukuje takiej wartości `X`, która podwojona jest większa od 3. Do wyboru ma dwie możliwe wartości 1 lub 2 (alternatywa `X is 1; X is 2`). Pierwsza z wartości nie spełnia zadanego warunku, natomiast druga go spełnia. Stąd odpowiedź `X=2`.

Polecenie 11

Użyte predykaty

Predykat `between(I0, I1, I)` jest spełniony gdy liczba całkowita `I` leży w zakresie `I0 .. I1`, gdzie `I0` i `I1` są liczbami całkowitymi. Można go również używać do generowania kolejnych liczb z zakresu. W tym celu należy jako trzeci jego argument podać zmienną, pod którą będą podstawiane kolejne liczby z zakresu (w kolejnych nawrotach).

Wpisz cel

```
?- between(7, 11, K), K =\= 9.
```

Wyjaśnienie

System udziela czterech odpowiedzi (naciskaj średnik po każdej z nich), gdyż w zakresie od 7 do 11 są cztery liczby całkowite różne od 9.

Czasami system sam zauważa, że nie ma więcej odpowiedzi i nie czeka na naciśnięcie średnika po ostatniej z nich.

Polecenie 12

Użyte predykaty

Predykat `halt` służy do zakończenia pracy z Prologiem.

Wpisz cel

```
?- halt.
```