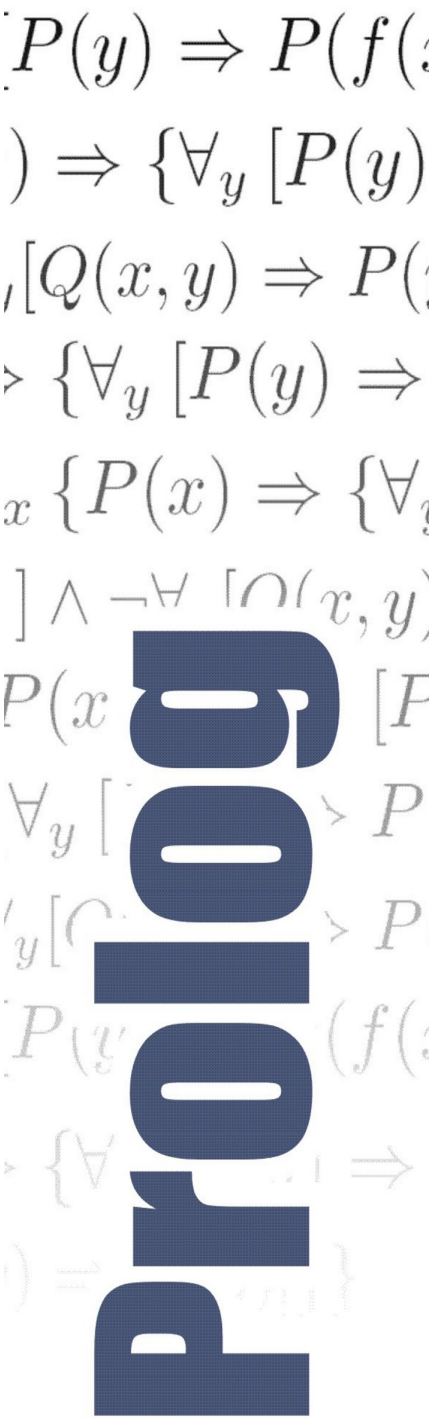
Katedra Wytrzymałości Materiałów
i Metod Komputerowych Mechaniki
www.kwmimkm.polsl.pl

Wydział Mechaniczny Technologiczny
Politechnika Śląska



Inżynieria wiedzy

Instrukcja do zajęć laboratoryjnych

5. Struktury, listy i rekurencja w języku Prolog

Opracował: mgr inż. Jacek Ptaszny
jacek.ptaszny@polsl.pl

Gliwice 2008

1 Cel ćwiczenia

Wykonując ćwiczenie zapoznasz się z możliwościami wykorzystania struktur i list, oraz z metodą definiowania reguł rekurencyjnych w języku Prolog.

2 Zanim przejdziemy dalej

Przypomnij sobie wszystkie informacje dotyczące programowania w języku Prolog, które poznałeś(aś) do tej pory. Znajdziesz je w instrukcji do zajęć laboratoryjnych nr 4.

3 Zaczynamy!

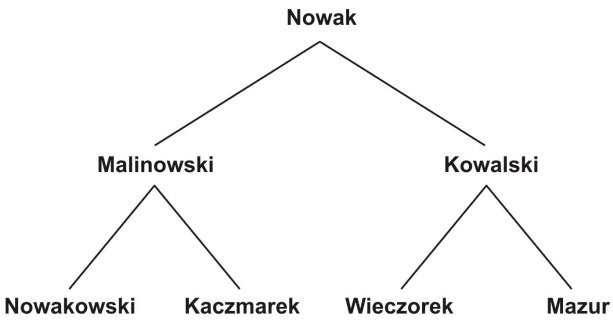
3.1 Struktury

3.1.1 Uruchom aplikację PIE i utwórz nowy plik programu

 Zapisz plik nadając mu nazwę *struktury.pro*. Pamiętaj o częstym zapisywaniu pliku.

3.1.2 Sformułuj struktury odwzorowujące schemat organizacyjny

Należy sformułować struktury odwzorowujące schemat organizacyjny, przedstawiający hierarchię stanowisk zajmowanych przez pracowników firmy:



Wpisz następujący kod:

```
schemat_org(nowak, podwladni(malinowski,kowalski)).
schemat_org(malinowski, podwladni(nowakowski,kaczmarek)).
schemat_org(kowalski, podwladni(wieczorek,mazur)).
schemat_org(nowakowski, podwladni(nikt,nikt)).
schemat_org(kaczmarek, podwladni(nikt,nikt)).
schemat_org(wieczorek, podwladni(nikt,nikt)).
schemat_org(mazur, podwladni(nikt,nikt)).
```

 Struktura to obiekt składający się ze zbioru innych obiektów.

 Wprowadzone struktury zawierają po dwa obiekty. Pierwszy z nich to atom określający nazwisko pracownika, natomiast drugi to struktura o nazwie "podwladni", zawierająca nazwiska podwładnych.

 Aby zachować jednolitą budowę wszystkich struktur, dla pracowników nie mających podwładnych, do struktury "podwladni" wprowadzono obiekty "nikt".

 Struktury mogą być zagnieżdżane jedna w drugiej teoretycznie nieograniczoną liczbę razy.

3.1.3 Utwórz regułę wyświetlającą nazwisko dowolnego pracownika oraz listę jego podwładnych

Do istniejącego kodu dopisz:

```
lista_podwladnych(nikt).
lista_podwladnych(Nazwisko) :- schemat_org(Nazwisko, podwladni(Nazwisko1, Nazwisko2)),
write(Nazwisko), nl, lista_podwladnych(Nazwisko1), lista_podwladnych(Nazwisko2).
```

 W części warunkowej reguły wykorzystano predykaty wbudowane. Predykat **write** powoduje wyświetlenie wartości jego argumentu. Predykat **nl** powoduje przejście do następnej linii.

 Wiele reguł formułowanych w języku Prolog ma budowę rekurencyjną (rekursyjną) - w części warunkowej definiowanej reguły występuje ona sama. Powoduje to cykliczne wywołanie reguły, aż do napotkania wywołania reguły dla predykatu "skrajnego".

 W tym przypadku reguła "lista_podwladnych" wykonywana jest dla kolejnych węzłów drzewa leżących na coraz niższych poziomach, aż do napotkania węzłów "nikt". Predykat "lista_podwladnych(nikt).", poprzedzający definicję reguły, zapewni zakończenie jej wykonywania bez wyświetlenia komunikatu "No solutions", ponieważ zostanie odnaleziony odpowiedni fakt w bazie wiedzy.

3.1.4 Zapisz plik programu i wczytaj go do obszaru roboczego

3.1.5 Uruchom regułę

Wpisz w oknie dialogowym:

```
lista_podwladnych(nowak).
```

Program powienien odpowiedzieć:

```
nowak
malinowski
nowakowski
kaczmarek
kowalski
wieczorek
mazur
True
1 Solution.
```

3.1.6 Wyświetl listy podwładnych wszystkich pracowników

Wydaj polecenia dotyczące kolejno wszystkich pracowników, analogicznie jak w punkcie poprzednim.

3.1.7 Wyświetl informacje dotyczące struktury

...będącej drugim argumentem predykatu "schemat_org" dla pracownika "nowak". W oknie poleceń wpisz:

```
schemat_org(nowak,X), functor(X,Y,Z).
```

Program powinien odpowiedzieć:

```
X = podwladni(malinowski,kowalski), Y = podwladni, Z = 2
1 Solution
```

 Argumenatmi predykatu wbudowanego **functor** są kolejno: struktura (X), nazwa struktury (Y) oraz liczba jej argumentów (Z).

3.1.8 Wyświetl drugi argument struktury

...będącej z kolei drugim argumentem predykatu "schemat_org" dla pracownika "nowak". W oknie dialogowym wpisz:

```
schemat_org(nowak,X), arg(2,X,Y).
```

Program powinien odpowiedzieć:

```
X = podwladni(malinowski,kowalski), Y = kowalski
1 Solution
```

 Argumenatmi predykatu wbudowanego **arg** są kolejno: numer argumentu (2), struktura (X) oraz wybrany argument struktury (Y).

3.2 Listy

3.2.1 Utwórz nowy plik programu

Zapisz plik pod nazwą "listy.pro".

3.2.2 Zdefiniuj listę dowolnych liczb

Wprowadź kod:

```
liczby([4, 20, 19, 11, 2, 5, 10]).
```

 Argumentem powyższego predykatu jest lista.

3.2.3 Zapisz plik programu i wczytaj go do obszaru roboczego

3.2.4 Wyświetl pierwszy element listy

W oknie dialogowym wpisz:

```
liczby([X|_]).
```

Program powinien odpowiedzieć:

```
X = 4
1 Solution
```

 Dowolną listę L można zapisać w postaci [L]=[H|T], gdzie H jest pierwszym elementem listy (ang. head) natomiast T jest pozostałą jej częścią (ang. tail).

 Listy mogą zawierać oprócz liczb również atomy, ciągi znaków i struktury.

3.2.5 Wyświetl wszystkie elementy listy oprócz pierwszego

W oknie dialogowym wpisz:

```
liczby([_|X]).
```

Program powinien odpowiedzieć:

```
X = [20, 19, 11, 2, 5, 10]
1 Solution
```

3.2.6 Wyświetl dwa pierwsze elementy listy

W oknie dialogowym wpisz:

```
liczby([X,Y|_]).
```

Program powinien odpowiedzieć:

```
X = 4,Y = 20
1 Solution
```

3.2.7 Napisz regułę wyświetlającą n-ty element listy

Dopisz kod:

```
nty_element([X|_], 1) :- write(X), nl.
nty_element([X|Y], N) :- N > 1, N1 is N - 1, nty_element(Y, N1).
```

 Reguła działa w sposób rekurencyjny "obcinając" pierwszy element listy N-krotnie. Jeśli na liście pozostaje już tylko jeden element, jest on wyświetlany i działanie reguły zostaje zakończone.

 Operator **is** oznacza przypisanie zmiennej zadanej wartości.

3.2.8 Wyświetl piąty element listy

W oknie dialogowym wpisz:

```
liczby(X),nty_element(X,5).
```

Program powinien odpowiedzieć:

```
2
X = [4,20,19,11,2,5,10]
1 Solution
```

3.3 Zdefiniuj regułę łączącą dwie listy

Do programu dopisz treść reguły:

```
dolacz([], L, L).
dolacz([H|T], L, [H|U]) :- dolacz(T, L, U).
```

 Przeanalizuj budowę reguły oraz wypróbuj jej działanie.

 Reguły rekurencyjne wykonujące operacje na listach formułuj według następującego schematu:

1. wyraź wszystkie listy będące argumentami konkluzji reguły, w formie [H|T],
2. nadaj takie same nazwy pierwszym elementom (H) list, których odpowiednie elementy muszą być jednakowe,
3. nadaj takie same nazwy pozostałym elementom (T) list, których odpowiednie elementy muszą być jednakowe,
4. sformułuj wymagane relacje pomiędzy pozostałymi elementami rozpatrywanych list,
5. dla list, dla których nie jest konieczne przedstawienie w formie [H|T], uprość ich zapis stosując pojedynczą zmienną,
6. sformułuj predykat dla "skrajnego" wywołania reguły.

3.4 Wykonaj zadania przedstawione przez prowadzącego zajęcia

4 Podsumowanie

Wykonując ćwiczenie nauczyłeś(aś) się jak w języku Prolog:

- definiować obiekty i wykonywać operacje na obiektach,
- definiować listy i wykonywać operacje na listach,
- definiować reguły rekurencyjne.

Literatura

[1] Bramer M., *Logic programming with Prolog*. Springer Science + Business Media, 2005.

[2] Cholewa W., Pedrycz W., *Systemy doradcze*. Wydawnictwo Politechniki Śląskiej, Gliwice 1987.

[3] Costa E., *Visual Prolog for Tyros*. http://www.visual-prolog.com/vip/tutorial.

[4] Kendal S., Creen M., *An Introduction to Knowledge Engineering*. Springer-Verlag, London 2007.

[5] Lucas R. J., *Mastering Prolog*. Taylor & Francis Books Ltd., 1996.

[6] Michalik K., *Prolog*. Aitech, Katowice, www.aitech.pl.

[7] Mulawka J. J., *Systemy ekspertowe*. WNT, Warszawa 1996.

[8] Nilsson J., Małuszyński J., *Logic, programming and Prolog (2ED)*. http://www.ida.liu.se/ ulfni/lpp, 2000.

[9] Russel S., Norvig P., *Artificial intelligence: A Modern Approach*. Prentice Hall, 2002.

[10] Rutkowski L., *Metody i techniki sztucznej inteligencji*. WNT, Warszawa 2005.