

PHỤ LỤC A

Một số chương trình Prolog

- Yêu cầu đọc hiểu, dừng lại thuật toán và đánh giá độ phức tạp các chương trình Prolog dưới đây.
- Chạy chương trình SWI-Prolog ra kết quả.

% Thao tác trên cây

tree(X) :-

 b_tree(5, b_tree(2, b_tree(8, et, et), b_tree(2, et, et)), b_tree(3, et,
 b_tree(9, b_tree(2, et, et), et))).

is_tree(et).

is_tree(X) :-

 X=tree(_, Y, Z), is_tree(Z), is_tree(Z).

in_ordre(et).

in_ordre(tree(R, G, D)) :-

 in_ordre(G), write(-), write(R), write(-), in_ordre(D).

pre_ordre(et).

pre_ordre(tree(R, G, D)) :-

 write(-), write(R), write(-), pre_ordre(G), pre_ordre(D).

post_ordre(et).

post_ordre(tree(R, G, D)) :-

 post_ordre(G), post_ordre(D), write(-), write(R), write(-).

root(et).

root(tree(R, G, D), X) :-

 X is R.

sort(et).

sort(tree(R, G, D)) :-

 RG is root(G, X), RD is root(D, X), R>RG, R<RD, sort(G), sort(D).

is_tree(emptytree).

is_tree(X) :-

 X=tree(A, B, C), number(A), is_tree(B), is_tree(C).

```

treetest(X) :-
    X=(tree(5, tree(2, tree(8, emptytree, emptytree), tree(2, emptytree,
        emptytree)), tree(3, emptytree, tree(9, tree(2, emptytree, emptytree),
        emptytree)))).

treesort(X) :-
    X=(tree(5, tree(3, tree(2, emptytree, emptytree), tree(25, emptytree,
        emptytree)), tree(11, tree(8, emptytree, tree(9, emptytree, emptytree)),
        tree(13, emptytree, emptytree)))).

affpre(X) :-
    X=emptytree.

affpre(X) :-
    X=tree(A, B, C), write(A), affpre(B), affpre(C).

affin(X) :-
    X=emptytree.

affin(X) :-
    X=tree(A, B, C), affin(B), write(A), affin(C).

affpost(X) :-
    X=emptytree.

affpost(X) :-
    X=tree(A, B, C), affpost(B), affpost(C), write(A).

tree_sort(X) :-
    X=emptytree.

tree_sort(X) :-
    X=tree(A, B, C), B=emptytree, C=emptytree.

tree_sort(X) :-
    X=tree(A, B, C), B=tree(Y, _, _), A>Y, tree_sort(B), C=emptytree.

tree_sort(X) :-
    X=tree(A, B, C), B=emptytree, C=tree(Y, _, _), Y>A, tree_sort(C).

tree_sort(X) :-
    X=tree(A, B, C), B=tree(Y, _, _), A>Y, tree_sort(B), C=tree(Z, _, _), Z>A,
    tree_sort(C).

% Bài toán Tháp Hà Nội (Hanoi towers).

towers :-
    repeat,
    write('Number of rings (or ctrl-c to end): '),
    read(X),
    hanoi(X),
    fail.

hanoi(N) :- move(N, left, center, right), !.

move(0, _, _, _) :- !.

```

```

move(N,A,B,C) :-
    M is N-1,
    move(M,A,C,B),
    inform(A,B),
    move(M,C,B,A).

inform(A,B) :-
    write([move,disk,from,A,to,B]), nl.

% Tạo bảng (Table).
table :-
    create_popup( _,_,_,_ ),
    for(0,15,Row),
    for(0,15,Colidx),
    Attr is Row*16+Colidx,
    Col is 5*Colidx,
    tmove(Row,Col),
    write(Attr),
    tmove(Row,Col),
    wa(5, Attr),
    fail.

table :-
    nl,nl,write( $Press any key to continue$ ),
    get0_noecho(_),
    exit_popup.

for(X, X, X) :- !.
for(Y, X, Y) :- true.
for(X, Y, Z) :- inc(X, X1), for(X1, Y, Z).

:- public misscann/0.
misscann :- solve(start-state(3,3,left), X), pathwrite(X).

solve(Init-state(0,0,right), Init-state(0,0,right)-finish) :- !.
solve(Init-S1, Final) :-
    new_state(S1, S2),
    not member(S2, Init),
    solve(Init-S1-S2, Final).

new_state(state(M1,C1,left), state(M2,C2,right)) :-
    move(M, C),
    M =< M1,
    C =< C1,
    M2 is M1-M,
    C2 is C1-C,
    balanced(M2, C2).

new_state(state(M1,C1,right), state(M2,C2,left)) :-

```

```

    move(M, C),
    M2 is M1+M,
    C2 is C1+C,
    M2 <= 3,
    C2 <= 3,
    balanced(M2, C2).

```

```

balanced(X, X) :- !.
balanced(3, X) :- !.
balanced(0, X).
move(2,0).
move(1,0).
move(1,1).
move(0,1).
move(0,2).
member(X, Y - X) :- !.
member(X, Y - Z) :- member(X,Y).
pathwrite(A-B) :- !,
    pathwrite(A),
    nl,
    !,
    pathwrite(B).
pathwrite(X) :- write(X).

```

% solve/1

% PROLOG interpreter written in PROLOG (so-called vanilla meta-interpreter)

% usage: ?-solve(PrologGoal).

```
solve(true).
```

```
solve((A, B)) :-
```

```
    solve(A),
```

```
    solve(B).
```

```
solve(A) :-
```

```
    clause(A,B),
```

```
    solve(B).
```

% Ví dụ một hệ chuyên gia

```

animal(su_tu_bien, [1, 2, 8, 18]).
animal(ca_voi, [1, 2, 4, 9]).
animal(su_tu, [1, 2, 5, 11]).
animal(báo, [1, 2, 5, 10, 17]).
animal(gau, [1, 2, 6, 8, 15, 16, 17]).
animal(huou_cao_co, [1, 5, 7, 10]).
animal(voi, [1, 5, 6, 12, 13, 18]).
animal(con_diec, [3, 7, 8, 15, 18]).
animal(chim_mòng_bien, [3, 8, 14, 15, 18]).
animal(dà_dieu_châu_phi, [7]).
animal(chim_canh_cut, [4, 8, 19]).

```

```
question(1) :-  
    write('Con vật này có lông không ?'),  
    nl, !, answer(1).  
question(2) :-  
    write('Con vậ.t này ăn thịt không ?'),  
    nl, !, answer(2).  
question(3) :-  
    write('Con vậ.t này biết bay không ?'),  
    nl, !, answer(3).  
question(4) :-  
    write('Con vậ.t này có sống trong nước không ?'),  
    nl, !, answer(4).  
question(5) :-  
    write('Con vậ.t này có sống ở Châu Phi không ?'),  
    nl, !, answer(5).  
question(6) :-  
    write('Con vậ.t này có sống ở trong rừng không ?'),  
    nl, !, answer(6).  
question(7) :-  
    write('Con vậ.t này có chân dài không ?'),  
    nl, !, answer(7).  
question(8):-  
    write('Con vậ.t này có ăn cá không ?'),  
    nl, !, answer(8).  
question(9):-  
    write('Con vậ.t này có ăn phù du sinh vật nổi không ?'),  
    nl, !, answer(9).  
question(10):-  
    write('Con vậ.t này có da lông lốm đốm không ?'),  
    nl, !, answer(10).  
question(11):-  
    write('Con vậ.t này có bờm không ?'),  
    nl, !, answer(11).  
question(12):-  
    write('Con vậ.t này có răng nanh hay ngà không ?'),  
    nl, !, answer(12).  
question(13):-  
    write('Con vậ.t này có sừng không ?'),  
    nl, !, answer(13).  
question(14):-  
    write('Con vậ.t này có làm tổ trong núi đá không ?'),  
    nl, !, answer(14).  
question(15):-  
    write('Con vậ.t này có một phần da (lông) màu trắng không ?'),  
    nl, !, answer(15).  
question(16):-  
    write('Con vậ.t này có một phần da (lông) màu nâu không ?'),  
    nl, !, answer(16).  
question(17):-
```

```

        write('Con vậ.t này có leo trèo trên cây không ?'),
        nl, !, answer(17).
question(18):-
        write('Con vậ.t này có da (lông) màu xám không ?'),
        nl, !, answer(18).
question(19):-
        write('Con vậ.t này có da (lông) màu trắng và đen không ?'),
        nl, !, answer(19).

% Nhận diện con vật
identifier :-
        animal(X, Y),
        put_questions(Y),
        explaining(X, Y),
        abolish(discovery, 2).
identifier(X) :-
        write('Con vật chưa biết.'), nl.

% Hệ chuyên gia đặt câu hỏi ngược lại người sử dụng
put_questions([]).
put_questions([ X|Y ]) :-
        questioner(X),
        put_questions(Y).
questioner(X) :-
        discovery(X, yes), !.
questioner(X) :-
        discovery(X, no), !, fail.
questioner(X) :-
        question(X).

% Phân tích câu trả lời
answer(X) :-
        read(R),
        assert(discovery(X, R)),
        R = yes.

explaining(X, Y) :-
        write('Đây là con :'),
        write(X), nl,
        write('Bởi vì :'),
        nl, explications(Y).

explications([ ]).
explications([ X1|X2 ]) :-
        expl(X1), nl,
        explications(X2).

expl(1) :-
        write('Con vậ.t này là loài động vật có vú.').

```

```
expl(2) :-  
    write('Con vậ.t này là loài động vật ăn thịt.').  
expl(3) :-  
    write('Con vậ.t này biết bay.').  
expl(4) :-  
    write('Con vậ.t này sống trong nước.').  
expl(5) :-  
    write('Con vậ.t này sống ở Châu Phi.').  
expl(6) :-  
    write('Con vậ.t này sống trong rừng.').  
expl(7) :-  
    write('Con vậ.t này có chân dài.').  
expl(8) :-  
    write('Con vậ.t này ăn cá.').  
expl(9) :-  
    write('Con vậ.t này ăn ăn phù du sinh vật nổi.').  
expl(10) :-  
    write('Con vậ.t này có da lông lốm đốm.').  
expl(11) :-  
    write('Con vậ.t này có bờm.').
```

Hướng dẫn sử dụng SWI-Prolog

I. Giới thiệu SWI-Prolog

SWI-Prolog thuộc họ Prolog Edinburgh do Giáo sư Jan Wielemaker xây dựng từ năm 1983 tại Khoa Khoa học Thông tin Xã hội, trường Đại học Amsterdam, Hà Lan (Dept. of Social Science Informatics, SWI, University of Amsterdam, Netherlands). Hiện nay, có thể tải miễn phí phần mềm SWI-Prolog phiên bản 5.2.11 for Windows từ địa chỉ Web :

<http://www.swi-prolog.org/>.

SWI-Prolog có một thư viện vị từ, và tài liệu hướng dẫn phong phú. SWI-Prolog hoạt động theo hệ thống đơn thể, có giao diện trao đổi hai chiều linh hoạt với ngôn ngữ C. SWI-Prolog là một ngôn ngữ sơ phạm, do vậy bị hạn chế ít nhiều về tốc độ biên dịch chương trình.

SWI-Prolog hoạt động trong môi trường đồ hoạ XPCE định hướng đối tượng (GUI) X-window. SWI-Prolog tương đối dễ sử dụng nhờ khai thác các đặc trưng tương tác đồ hoạ. SWI-Prolog có các phiên bản cho các máy chạy Linux/Unix, Macintosh. Khi tải SWI-Prolog, NSD có thể tải thêm tài liệu hướng dẫn sử dụng.

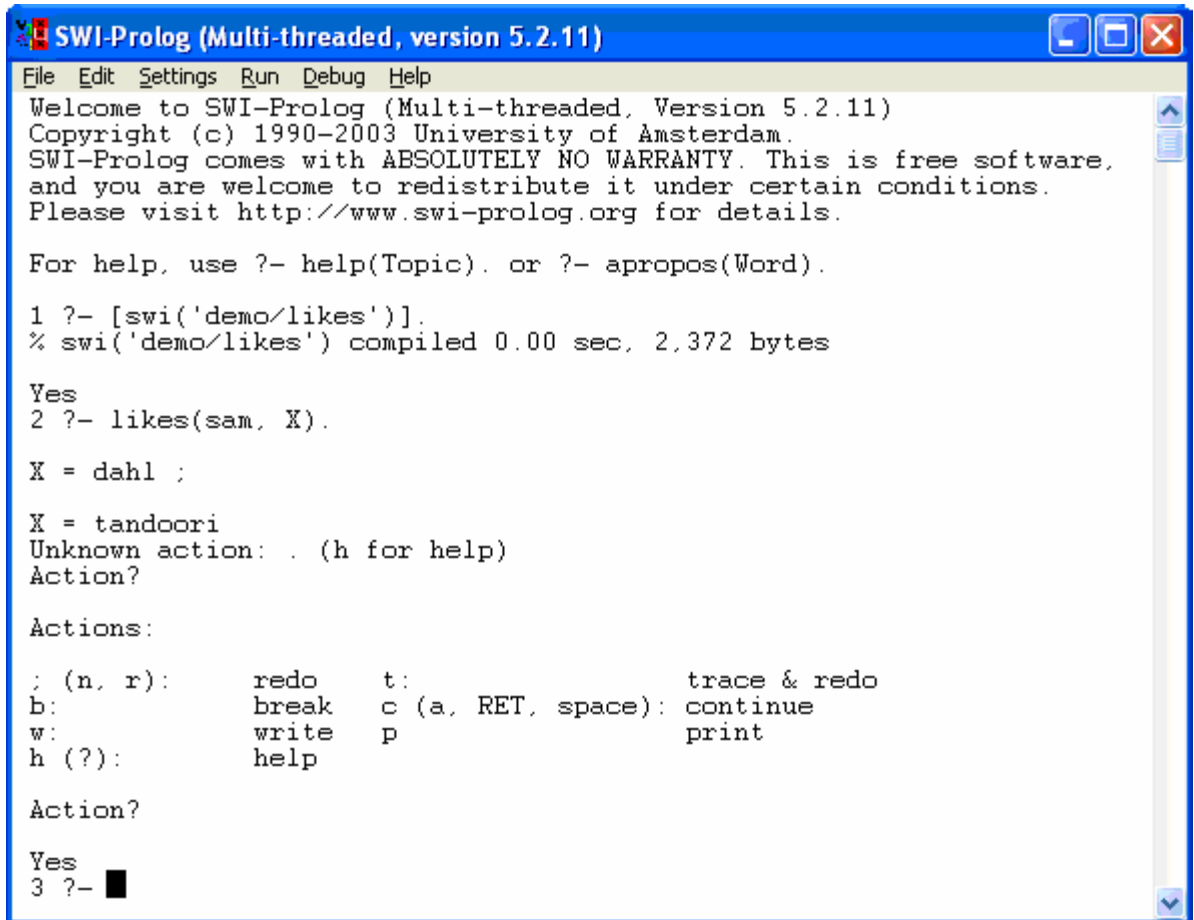
Sau khi cài đặt phiên bản SWI-Prolog 5.2.11 for Windows, rồi khởi động trình plwin.exe, cửa sổ làm việc SWI-Prolog hiện ra như hình dưới đây.

SWI-Prolog làm việc theo chế độ tương tác (interpreter mode) (xem hình). Dấu nhắc lệnh SWI-Prolog là một cặp dấu chấm hỏi và dấu gạch ngang (dấu trừ), được liệt kê thứ tự 1, 2, 3... để theo dõi quá trình làm việc của NSD, theo sau là con trỏ màn hình (dấu chèn) nhấp nháy :

1 ?- ■

Để ra khỏi SWI-Prolog, dùng lệnh File/ Exit, hoặc dùng vị từ halt :

?- halt.



```

SWI-Prolog (Multi-threaded, version 5.2.11)
File Edit Settings Run Debug Help
Welcome to SWI-Prolog (Multi-threaded, Version 5.2.11)
Copyright (c) 1990-2003 University of 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).

1 ?- [swi('demo/likes')].
% swi('demo/likes') compiled 0.00 sec, 2,372 bytes

Yes
2 ?- likes(sam, X).

X = dahl ;

X = tandoori
Unknown action: . (h for help)
Action?

Actions:

; (n, r):      redo      t:      trace & redo
b:            break    c (a, RET, space): continue
w:            write    p      print
h (?):        help

Action?

Yes
3 ?- █

```

Hình I.1. Cửa sổ làm việc của SWI-Prolog.

II. Làm việc với SWI-Prolog

II.1. Đặt câu hỏi

Sau khi một chương trình Prolog được biên dịch hoặc được tải vào bộ nhớ, NSD có thể đặt câu hỏi truy vấn (kết thúc bởi một dấu chấm). Tùy theo câu hỏi (đích phải xoá), Prolog trả lời đúng (Yes) hoặc sai (No) và kèm theo kết quả $X = \langle \text{value} \rangle$ nếu trong đích có chứa biến X nào đó. Trong trường hợp có nhiều câu trả lời, ngay sau kết quả trả lời đầu tiên, NSD có thể đặt một dấu chấm phẩy ; (semi-colon) nếu muốn tiếp tục yêu cầu Prolog đưa ra các câu trả lời khác. Tiếp tục quá trình này, Prolog lần lượt đưa ra các kết quả khác nhau cho đến khi, hoặc Prolog trả lời No, thì có nghĩa là không còn câu trả lời nào nữa, hoặc Yes, nếu NSD muốn dừng lại bằng cách gõ Enter (↵). NSD có thể nhận được thông báo lỗi nếu câu hỏi có vấn đề :

ERROR: Undefined procedure <... /...> % NSD đã gõ sai tên thủ tục
 hoặc :
 ERROR: Syntax error: Operator expected % NSD đã gõ sai biểu thức, v.v...

II.2. Chạy trình demo

Sau đây là ví dụ chạy chương trình demo likes.pl của SWI-Prolog. Nội dung tệp likes.pl như sau :

```
likes(sam, A) :-
    indian(A),
    mild(A).
likes(sam, A) :-
    chinese(A).
likes(sam, A) :-
    italian(A).
likes(sam, chips).
italian(pizza).
italian(spaghetti).
chinese(chow_mein).
chinese(chop_suey).
chinese(sweet_and_sour).
mild(dahl).
mild(tandoori).
mild(kurma).
indian(curry).
indian(dahl).
indian(tandoori).
indian(kurma).
```

NSD nạp chương trình vào bộ nhớ bởi lệnh sau :

```
?- [swi('demo/likes')].
% swi('demo/likes') compiled 0.00 sec, 2, 340 bytes
Yes
```

Sau khi tệp likes.pl được tải vào bộ nhớ, NSD đặt các câu hỏi và nhận được kết quả như sau :

```
?- like(X, Y).
Correct to: likes(X, Y)? Yes
X = sam
Y = dahl ;
X = sam
Y = tandoori ;
```

...

Yes % NSD đã gõ Enter và vẫn còn câu trả lời.

?-

II.3. Chạy trình demo XPCE

SWI-Prolog được trang bị chức năng lập trình hướng đối tượng tạo giao diện đồ hoạ, được gọi là XPCE, có mã nguồn tương thích với Unix/X11 và Windows.

NSD có thể thực hiện các chương trình demo XPCE bằng cách gõ :

```
?- [swi('xpce/prolog/demo/pce_demo')].
```

```
% contrib(contrib) compiled into pce_contrib 0.02 sec, 1,260 bytes
```

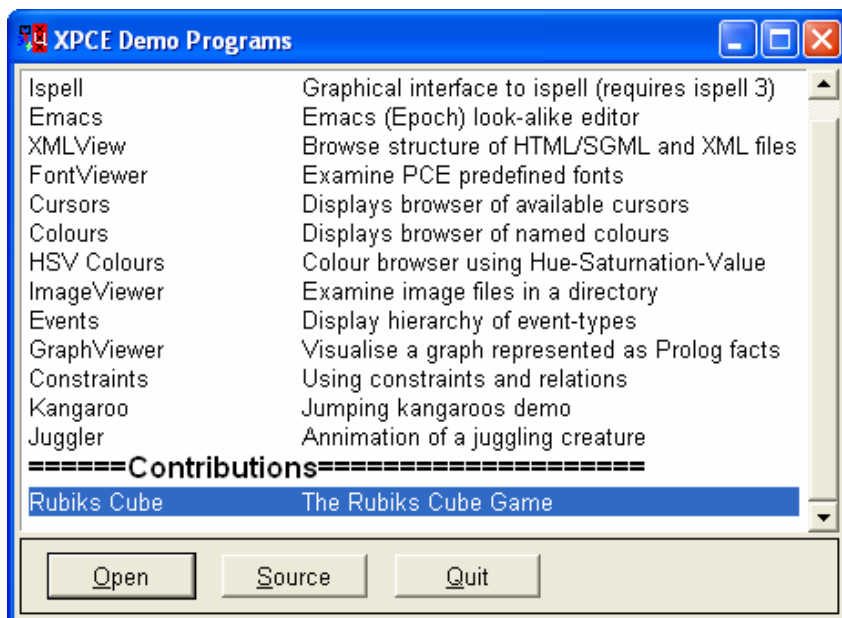
```
% swi('xpce/prolog/demo/pce_demo') compiled into pce_demo 0.02 sec, 11,408 bytes
```

Yes

```
?- pcedemo.
```

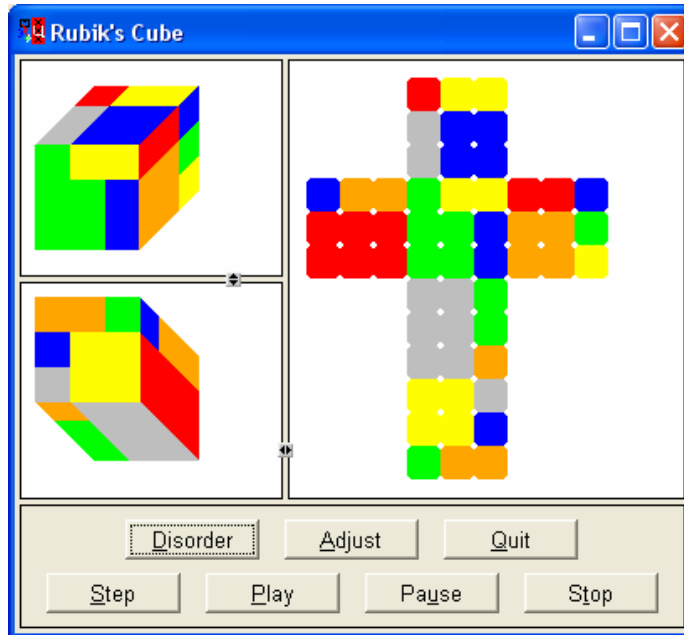
Yes

Một cửa sổ hiện ra như sau :



Hình II.1. Các chương trình demo XPCE của SWI-Prolog.

NSD chọn xem một trò chơi bằng cách lựa tên trò chơi rồi Open. Sau đó, nhấp chuột tại nút Quit để kết thúc chương trình. Ví dụ trò chơi Rubiks Cube như hình dưới đây :



Hình II.2. Trình demo XPCE Rubiks Cube của SWI-Prolog.

II.4. Các lệnh đơn (Menu commands)

Cửa sổ làm việc SWI-Prolog gồm một số lệnh đơn chủ yếu như sau :

File/consult...

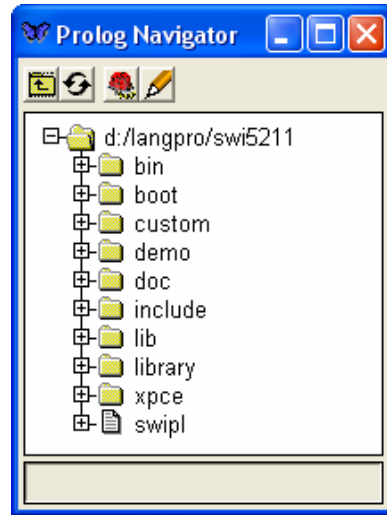
Tương tự lệnh `consult(File)` dùng để nạp tệp chương trình File vào bộ nhớ.

File/Reload modified files

Dùng để nạp lại (reloads) một tệp chương trình đã bị thay đổi vào bộ nhớ. Thường được sử dụng sau khi đã soạn thảo trên chương trình. có thể sử dụng lệnh **make**.

File/Navigator ...

Mở cửa sổ tương tự explorer của Windows để tìm kiếm tệp hoặc vị trí Prolog trong các thư mục, ổ đĩa. ..



Hình II. Cửa sổ Navigator.

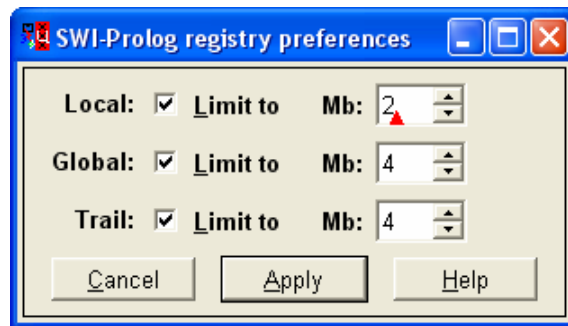
Settings/Font ... Dùng để thay đổi phông chữ trong giao diện SWI-Prolog.

Settings/User init file ...

Soạn thảo tệp cấu hình pl.ini chứa các thông tin cài đặt hệ thống và chú thích.

Settings/Stack sizes ...

Cho phép định nghĩa lại kích thước không gian làm việc của các danh sách đẩy xuống (stack sizes).



Hình II.3 Định nghĩa kích thước danh sách đẩy xuống

Run/Interrupt

Ngắt một tiến trình Prolog (Prolog process) có thể sử dụng tổ hợp phím Control-C có cùng hiệu quả tương tự. Lúc này, xuất hiện đối thoại :

Action (h for help) ? Options:

a:	abort	b:	break
c:	continue	e:	exit

```

g:          goals          t:          trace
h (?):      help
Action (h for help) ? abort
ERROR: Execution Aborted
% Execution Aborted
?-

```

Run/New thread

Tạo một cửa sổ làm việc mới để thực hiện chương trình. Trong cửa sổ mới này có thể dùng chung chương trình và dữ liệu đã có mặt trong cửa sổ làm việc chính.

Debug/Edit spy points ...

Soạn thảo các điểm ngắt trong các vị từ.

Help Tìm đọc các nội dung hướng dẫn sử dụng SWI-Prolog khác nhau.

II.5. Soạn thảo chương trình

NSD có thể sử dụng một hệ soạn thảo văn bản bất kỳ của Windows như NotePad, NCeditor... để soạn thảo chương trình Prolog, lưu cất lên đĩa, sau đó tải vào môi trường SWI-Prolog để chạy nhờ lệnh :

```
?- consult( <file_name> ).
```

hoặc :

```
?- [ <file_name> ].
```

Với <file_name> là tên tệp chương trình (không cần ghi phần mở rộng .pl. Có thể chỉ định đường dẫn thư mục đến tên tệp. Chú ý dấu chấm hỏi ?- đặt trước mỗi câu lệnh là dấu nhắc của SWI-Prolog.

Ví dụ nạp chương trình bài toán 8 Quân Hậu (xem phần trình bày lý thuyết) :

```
?- [myex5].
```

```
% myex5 compiled 0.00 sec, 2,020 bytes
```

```
Yes
```

```
?- solution( 8, S ).           % Tìm một lời giải cho bài toán 8 Quân Hậu
```

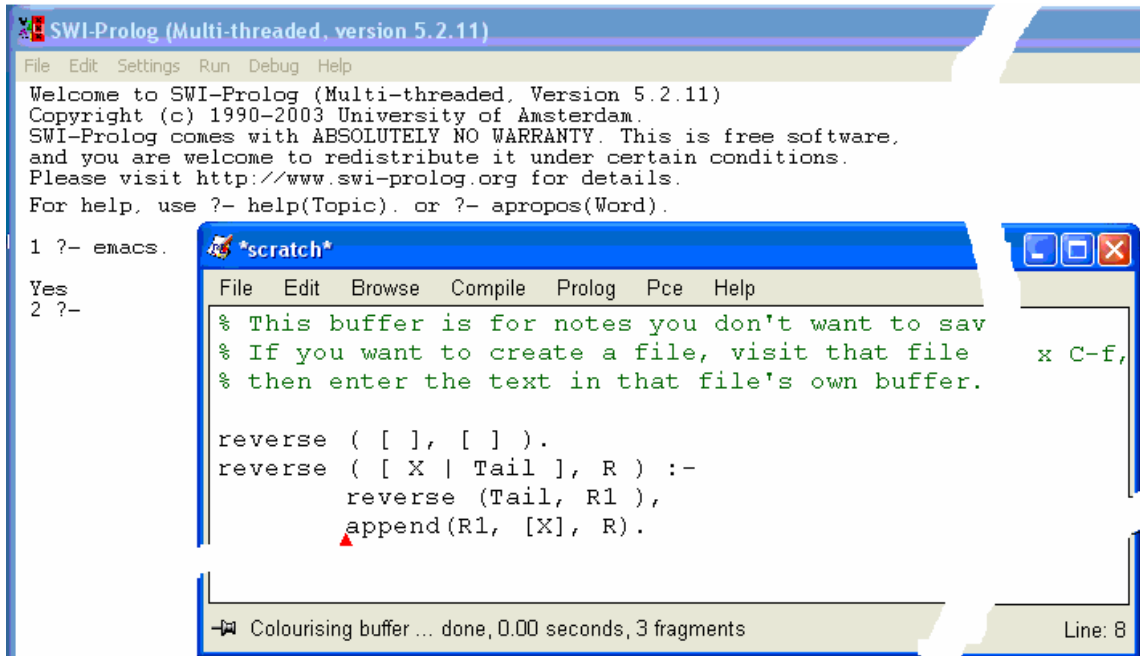
```
S = [1, 7, 4, 6, 8, 2, 5, 3]    % Vị trí Quân Hậu trên các cột.
```

```
Yes
```

Khi đã nạp chương trình Prolog vào bộ nhớ, dùng lệnh :

```
?- Listing.
```

để liệt kê các dòng lệnh trong cửa sổ làm việc chính của Prolog.



Hình II.4. Cửa sổ soạn thảo văn bản PceEmacs của SWI-Prolog.

SWI-Prolog có sẵn một hệ soạn thảo *PceEmacs* rất thuận tiện để soạn thảo và thực hiện chương trình. Gõ vào dòng lệnh :

?- emacs.

để mở cửa sổ soạn thảo văn bản nhiều chức năng (scratch).

III. Một số lệnh SWI-Prolog thông dụng

Sau đây là một số ví từ điều khiển môi trường thông dụng của SWI-Prolog.

consult(+File)

Nạp tệp chương trình vào bộ nhớ. Chú ý sử dụng dấu phân cách thư mục là */*. Ví dụ :

% Nạp tệp likes.pl từ thư mục làm việc (xem pwd) vào bộ nhớ.

?- consult(likes).

% Nạp tệp likes.pl sử dụng đường dẫn đầy đủ - tuyệt đối (absolute path).

?- ['C:/Program Files/pl/demo/likes'].

% Sử dụng đường dẫn kiểu Windows (Windows-style path-name).

?- ['C:\\Program Files\\pl\\demo\\likes'].

pwd

Đưa ra thư mục làm việc (working directory hay folder) của SWI-Prolog.

ls

Liệt kê danh sách các tệp trong thư mục hiện hành.

edit

Nếu Prolog được khởi động bởi một tệp .pl từ trong Windows Explorer, thì chạy trình soạn thảo mặc nhiên, chẳng hạn Windows Notepad, để soạn thảo tệp này. Có thể sử dụng lệnh đơn **File/Edit...**

?- edit.

% Waiting for editor ...

...

Yes % Sau khi NSD kết thúc soạn thảo

edit(+File)

Sản thảo tệp hay đơn thể chương trình đã có mặt trên đĩa.

?- edit(myex). % Mở Notepad để soạn thảo tệp myex.pl

% Waiting for editor ... % hiện có ở thư mục hiện hành

...

Yes % Sau khi NSD kết thúc soạn thảo

make

Nạp tệp chương trình vào bộ nhớ sau khi đã thực hiện một số thay đổi.

trace

Chạy trình duyệt tìm sửa lỗi debugger.

apropos(+Keyword)

Tìm vị từ có chứa từ khoá Keyword.

help(+Spec)

Mở cửa sổ hướng dẫn về Spec, là tên một vị từ hoặc tên một hàm C.

Tài liệu tham khảo

1. I. Bratko (L. Ricard dịch), *Programmation en Prolog pour l' intelligence artificielle*, InterÉditions, Paris, 1988.
2. I. Bratko, *Programming for Artificial Intelligence*, AddisonWesley, 1987.
3. K.L. Clavle, S.A. Tarnlound. *Logic Programming*. Accademic Press 1983.
4. G. Falquet. *Prolog et programmation logique*. Notes de cours, Université de Genève, 2002.
5. H. Farrenry, M. Ghallab. *Éléments d'intelligence artificielle*, Hermes, Paris-Londre-Lausanne 1990.
6. Bạch Hưng Khang, Hoàng Kiêm. *Trí tuệ nhân tạo : các phương pháp và ứng dụng*. Nhà Xuất bản Khoa học Kỹ thuật, 1989.
7. J.W. Leoyd. *Foundations of Logic Programming*, Springer-Verlag, 1984.
8. P. Nugues. *La Programmation logique et le langage Prolog*. Notes de cours, ISMRA, 2002.
9. D.Teller. *Partiel de Prolog*. Notes de cours, École Centrale de Lyons, 2001.
10. Nguyễn Thanh Thuỷ. *Trí tuệ nhân tạo : các phương pháp giải quyết vấn đề và kỹ thuật xử lý tri thức*. Nhà Xuất bản Giáo dục, 1986.
11. A.Voronkov, *Logic Programming and automated reasoning*, Springer-Verlag, 1993.
12. Các tài liệu về Prolog và bài tập Prolog trên Internet :
ftp://ftp.umh.ac.be/pub/ftp_sgl/LangagesProgrammation/LP-exercProlog.pdf
<http://www.etse.urv.es/EngInf/assig/iai/Laboratoris/Prolog/>
http://membres.lycos.fr/epl2000/2eme/POO/SPV_Prolog/1_Prolog_leLangage/