ĐẠI HỌC QUỐC GIA TP.HCM
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

CƠ SỞ TRÍ TUỆ NHÂN TẠO

BÁO CÁO ĐỒ ÁN 2 – LOGIC BẬC NHẤT

LÀM QUEN VỚI CÔNG CỤ PROLOG

Thành viên nhóm:

Đào Thành Đạt –18120014

Trần Quốc Bảo – 18120111

Lê Minh Khoa – 18120415

Mục Lục

PHẦN 1: TÌM HIỂU NGÔN NGỮ PROLOG

Tổng quan về ngôn ngữ Prolog

Prolog là ngôn ngữ được sử dụng phổ biến trong lĩnh vực trí tuệ nhân tạo. Nguyên lý lập trình logic dựa trên các mệnh đề Horn (Horn logic). Là một ngôn ngữ cấp cao, có đặc điểm gần với ngôn ngữ tự nhiên, từ những người mới học đến những lập trình viên chuyên nghiệp đều có thể tiếp cận một cách nhanh chóng, viết ra một chương trình ứng dụng hữu ích. Khác với những ngôn ngữ cấu trúc như Pascal, hay C mà ta đã làm quen, Prolog là một ngôn ngữ mô tả, với một số sự kiện và quy luật suy diễn đã mô tả, Prolog sẽ suy luận cho ta các kết quả. Pro

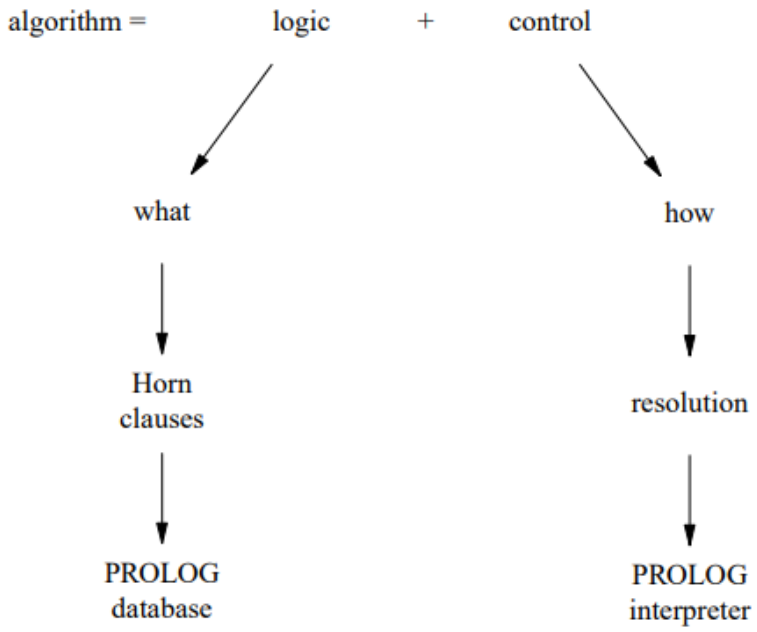


Figure 1: The relationship between PROLOG and logic programming.

Khái niệm: Prolog là một ngôn ngữ lập trình. Tên gọi Prolog được xuất phát từ cụm từ tiếng Pháp Programmation en logique, nghĩa là "lập trình theo lô gíc". Xuất hiện từ năm 1972 (do Alain Colmerauer và Robert Kowalski thiết kế), mục tiêu của Prolog là giúp người dùng mô tả lại bài toán trên ngôn ngữ của logic, dựa trên đó, máy tính sẽ tiến hành suy diễn tự động dựa vào những cơ chế suy diễn có sẵn (hợp nhất, quay lui và tìm kiếm theo chiều sâu) để tìm câu trả lời cho người dùng. Prolog được sử dụng nhiều trong các ứng dụng của trí tuệ nhân tạo và ngôn ngữ học trong khoa học máy tính (đặc biệt là trong ngành xử lý ngôn ngữ tự nhiên vì đây là mục tiêu thiết kế ban đầu của nó). Cú pháp và ngữ nghĩa của Prolog đơn giản và sáng sủa, nó được người Nhật coi là một trong những nền tảng để xây dựng máy tính thế hệ thứ năm mà ở đó, thay vì phải mô tả cách giải quyết một bài toán trên máy tính, con người chỉ cần mô tả bài toán và máy tính sẽ hỗ trợ họ nốt phần còn lại.

1.1.Các đặc điểm cơ bản của Prolog

1.1.1. Đối tượng

Các đối tượng trong Prolog có thể là sự việc, sự vật, con người,… được biểu thị bởi các hằng và biến trong chương trình. Hằng là giá trị không thể thay đổi, còn các biến có giá trị thay đổi được gán trong khi chạy chương trình.

Ví dụ: con mèo, Bảo, xe hơi, giá vé tàu….

1.1.2. Quan hệ giữa các đối tượng

Được dùng dưới hình thức vị từ, bao gồm tên của các vị từ và các đối số của nó, các đối số được đặt trong dấu ngoặc đơn, cách nhau bởi dấu phẩy.

Ví dụ:

Ngôn ngữ tự nhiên	Prolog
Jonh là cha của susan. Bảo mệt. John ăn pizza.	father(john,susan). tired(Bao) eats(john,pizza)

1.1.3. Sự kiện (Facts), luật (Terms), câu hỏi(queries)

- Sự kiện: là một vị từ diễn tả một đặc điểm của một đối tượng, một sự vật nào đó.
  Ví dụ: 5 là số tự nhiên.
- Luật: được trình bày dưới dạng mệnh đề, biểu thị quy tắc liên quan đến đối tượng đó.
  Ví dụ: "Scott là ông nội của Susan" là đúng nếu "Scott là cha của John" đúng và "John là cha của Susan" đúng.
- Câu hỏi: dùng để hỏi máy tính về các dữ kiện đã được đưa ra dưới dạng một mệnh đề.
  Ví dụ: (sử dụng dữ liệu ở bảng 2.2)

Câu hỏi	Prolog (dạng prompt ?-)	Prolog phản hồi
Is John the father of Susan?	father(john,susan).	yes (or(true))
Is Bao tired?	tired(Bao).	yes
Is Dung tired?	tired(Dung).	no (of false)
Who is tired?	Tired(X)	X = Bao
Who eats pizza?	eats(X, pizza).	X = John

1.2.Cấu trúc của một chương trình Prolog

1.2.1. Cú pháp của các thành phần cơ bản:

Sự kiện	`<clause>`
Luật	`<clause> :- <clause>`
Câu hỏi	`<?-> + <clause>` Ví dụ: `?- eats(X, pizza).`
Biến	Một ký tự hoa hoặc một chuỗi ký tự, bắt đầu bằng một ký tự hoa. Có thể dùng _ (dấu gạch dưới) để đặt cho biến tự do. Ví dụ: X, Abc, _,….
Hằng	Gồm các số, bắt đầu bằng chữ thường và đặt trong dấu ngoặc kép.

1.2.2. Phần Domains

Đây là phần khai báo kiểu các đối tượng mà ta sử dụng, ta có thể định nghĩa kiểu dữ liệu mới dựa trên những kiểu dữ liệu đã biết.

Các kiểu dữ liệu này sử dụng cho các đối số của vị từ. Ta có thể sử dụng những kiểu dữ liệu có sẵn như để chương trình mạch lạc, ta thường định nghĩa lại các kiểu cơ bản.

Cú pháp: `<danh sách kiểu mới> = <kiểu đã biết>`

Hoặc: `<danh sách kiểu mới> = <danh sách kiểu đã biết>`

Trong đó các kiểu mới phân cách nhau bởi dấu phẩy, còn các kiểu đã biết phân cách nhau bởi dấu chấm phẩy.

Ví dụ:

```
Domains
ten, tacgia, diachi = string
namsx = integer
dientich = real
dovat = sach(ten, tacgia, namsx);nha(dia_chi, dien_tich)
```

Trong ví dụ trên, ta đã định nghĩa các kiểu mới, trong đó có:

- ten, tacgia, diachi: dựa vào kiểu dữ liệu đã biết string.
- Namsx, dientich: dựa vào kiểu dữ liệu đã biết integer, real.
- Dovat: dựa vào kiểu dữ liệu mới sach, nha.

1.2.3. Phần Predicates

Đoạn khai báo kiểu cho vị từ sẽ dùng trong các đoạn sau

1.2.4. Phần Clause

Đoạn liệt kê sự kiện và định nghĩa qui tắc đã khai báo trong phần Predicates

1.2.5. Phần Goal

Đoạn đưa vào các vị từ câu hỏi, có thể đặt trước phần Clauses

1.3.Các nguyên tắc của ngôn ngữ Prolog

1.3.1. Đồng nhất

Một quan hệ bất kì đồng nhất với một quan hệ cùng tên với cùng số lượng tham số, với các đại lượng này đồng nhất theo từng cặp.

Một biến cũng có thể đồng nhất với biến, hằng khác.

Ví dụ:

Ta có thể thay thế tired(Bao), tired(Dat), tired(Khoa) bằng tired(X).

Khi đó thay vì trả về kết quả yes, tired(X) sẽ trả về 3 kết quả: Bao, Dat, Khoa.

1.3.2. Hợp giải, quay lui

Ví dụ: Ta có các Facts:

All men are mortal

Socrates is a man.

Thì từ đó ta có:

Socrates is mortal.

Nghĩa là, theo tam đoạn luận, ta có "Tất cả những người đàn ông đều phải chết", "Socrates là một người đàn ông" nên từ đó suy ra "Socrates chết".
Prolog sử dụng **hợp giải** để xâu chuỗi các mệnh đề một cách máy móc và chứng minh một truy vấn. Thuật toán này thường bắt nguồn từ nguyên tắc phân giải của Robinson (1965), được gọi là SLD resolution.

Giả sử hệ thống đang chứng minh goal g, trong đó g được mô tả như sau:
g :- g_1, g_2, …, g_{j-1}, g_j, …, g_n. Khi các g_i kiểm chứng từ trái sang phải, đến g_j là sai thì hệ thống sẽ **quay lui** lại g_{j-1} để tìm lời giải khác.

1.4.Các kiểu dữ liệu

Prolog cung cấp các kiểu dữ liệu chuẩn là: char, integer, real, string và symbol.
Char: Kiểu kí tự. Nằm trong dấu nháy đơn. Ví dụ: 'a', '+'.
Integer: Là kiểu số nguyên, tập giá trị từ -32768 đến 32767. Ví dụ: 123, 21.
Real: Kiểu số thực. Ví dụ: 1.1.
String: Kiểu chuỗi kí tự, nằm trong dấu ngoặc kép. Ví dụ: "Bao".
Symbol: Kiểu dữ liệu sơ cấp.

1.5.Phép toán số học

Toán tử số học trong Prolog bao gồm một tập hợp các toán tử tiền tố, tiền tố và hậu tố.
Ký hiệu số học: "+", "-", "*" và "/".
Trình dịch Prolog coi toán tử như các hàm và biến đổi các biểu thức thành các terms.
Ví dụ: `2 * 3 + 4 * 2` sẽ tương đương với `+(*(2, 3), *(4, 2))`.
Bảng độ ưu tiên thứ tự toán tử(từ 1-1200).

Priority	Specifier	Operators
1,200	xfx	:- -->
1,200	fx	:- ?-
1,100	xfy	;
1,050	xfy	->
1,000	xfy	','
900	fy	\+
700	xfx	= \=
700	xfx	== \== @< @=< @> @>=
700	xfx	=..
700	xfx	is =:= =\= < =< > >=
550	xfy	:
500	yfx	+ - # /\ \/
400	yfx	* / // rem mod << >>
200	xfx	**
200	xfy	^
200	fy	+ - \

1.6.Phép toán so sánh

Các phép toán so sánh trong Prolog xử lí biểu thức số học và chữ. Sau khi tính toán biểu thức ở cả 2 vế thì Prolog mới đưa ra kết quả cuối cùng.

Ví dụ:

```
?- 1 + 2 < 3 + 4
True
```

Kí tự được so sánh bằng mã ASCII.

Ví dụ:

```
?- a @< b.
true.
```

Ngoài ra còn một số phép só sánh khác:

	Arithmetic comparison	Literal term comparison
Equality operator	=:=	==
Inequality operator	=\=	\==
Less than	<	@<
Less than or equal	=<	@=<
Greater than	>	@>
Greater than or equal	>=	@>=

1.7.Các hàm nhập xuất

1.7.1. Nhập xuất kí tự:

Nhập xuất mặc địnhsẽ được thực hiện dưới hai luồng chuẩn(standard I/O stream) thường là nhập vào từ bàn phím và xuất ra màn hình.

get_char(?Char)	Đọc kí tự tiếp theo của input stream hiện tại.
put_char(+Char)	Ghi kí tự vào output stream hiện tại.
nl/0	Ghi một dòng mới vào output stream.

1.7.2. Nhập xuất luật(Terms) và tệp(files):

read(?Term)	Đọc một term từ input stream hiện tại.
write(?Term)	Ghi một term vào output stream hiện tại.
open(+SourceSink, +Mode, -Stream)	Mở tệp SourceSink ở Mode đầu vào hoặc đầu ra. Luồng (Stream) hợp nhất với luồng đã mở và sử dụng cho các hoạt động tiếp theo.
close(+Stream)	Đóng luồng Stream.

1.8.Kỹ thuật đệ qui trong Prolog

Đệ qui theo cách đã biết tốn rất nhiều tài nghiên về bộ nhớ, bộ nhớ tang theo cấp số mũ tuỳ thuộc vào số lần gọi đệ qui. Tuy nhiên có một số trường hợp Prolog có thể biến đổi đệ qui về phép lặp gọi là đệ qui đuôi.

Ví dụ:

Bài toán Tháp hà nội: Có 3 cọc A, B, C. Trên cọc A có n đĩa tròn chồng lên nhau với đường kính khác nhau, sao cho đĩa bé luôn nằm trên đĩa lớn. Ta có thêm 2 cọc B, C trống. Tìm cách chuyển n đĩa từ A đến C với số lần di chuyển đĩa ít nhất (2^n-1), sao cho: mỗi lần chỉ di chuyển 1 đĩa bé nhất trên cùng của một cọc tùy ý sang một cọc khác để đĩa bé luôn nằm trên đĩa lớn.

Domain	`i = integer` `s = symbol`
Predicates	`HaNoi(i, i)` `ChuyenDia(i, i, s, s, s, i)`

Clauses	`HaNoi(N, SoLanDiChuyen) :- ChuyenDia(N, N, a, c, b, SoLanDiChuyen).`
	`ChuyenDia(1, Nhan, Tu, Den, _, 1) :- !, nl,`
	`write("Chuyen dia: ", Nhan, " tu ", Tu, " den ", Den).`
	`ChuyenDia(N, Nhan, Tu, Den, Tgian, Tong):-`
	`N1=N-1, NhanPhu=Nhan-1,`
	`ChuyenDia(N1, NhanPhu, Tu, TGian, Den, Tong1),`
	`ChuyenDia(1, Nhan, Tu, Den, Tgian, 1),`
	`ChuyenDia(N1, NhanPhu, TGian, Den, Tu, Tong2),`
	`Tong = Tong1 + Tong2 + 1.`

Goal	`makewindow(1, 3, 7, "Thap Ha Noi", 0, 0, 25, 80),`
	`write("Nhap so dia n (1<=n<=12): "), readint(N),`
	`HaNoi(N, Tong),`
	`write("Tong so lan di chuyen dia: ", Tong)..`

Kết quả với số đĩa là 2	`Chuyen dia: 1 tu a den b`
	`Chuyen dia: 2 tu a den c`
	`Chuyen dia: 1 tu b den c`
	`Tong so lan di chuyen dia: 3`

Kết quả của chương trình với n=2 là:

```
Chuyen dia: 1 tu a den b
Chuyen dia: 2 tu a den c
Chuyen dia: 1 tu b den c
Tong so lan di chuyen dia: 3
```

PHẦN 2: TÌM HIỂU MÔI TRƯỜNG SWI-PROLOG

2.1. Giới thiệu:

SWI-Prolog là một môi trường lập trình Prolog miễn phí, thường được sử dụng cho mục đích giáo dục và các ứng dụng mạng ngữ nghĩa. Prolog có ưu điểm có nhiều tính năng, thư viện cho lập trình logic ràng buộc, đa luồng, GUI, interface cho Java, ODBC, và các công cụ cho lập trình viên (IDE với debug bằng GUI). SWI-Prolog chạy được trên nhiều

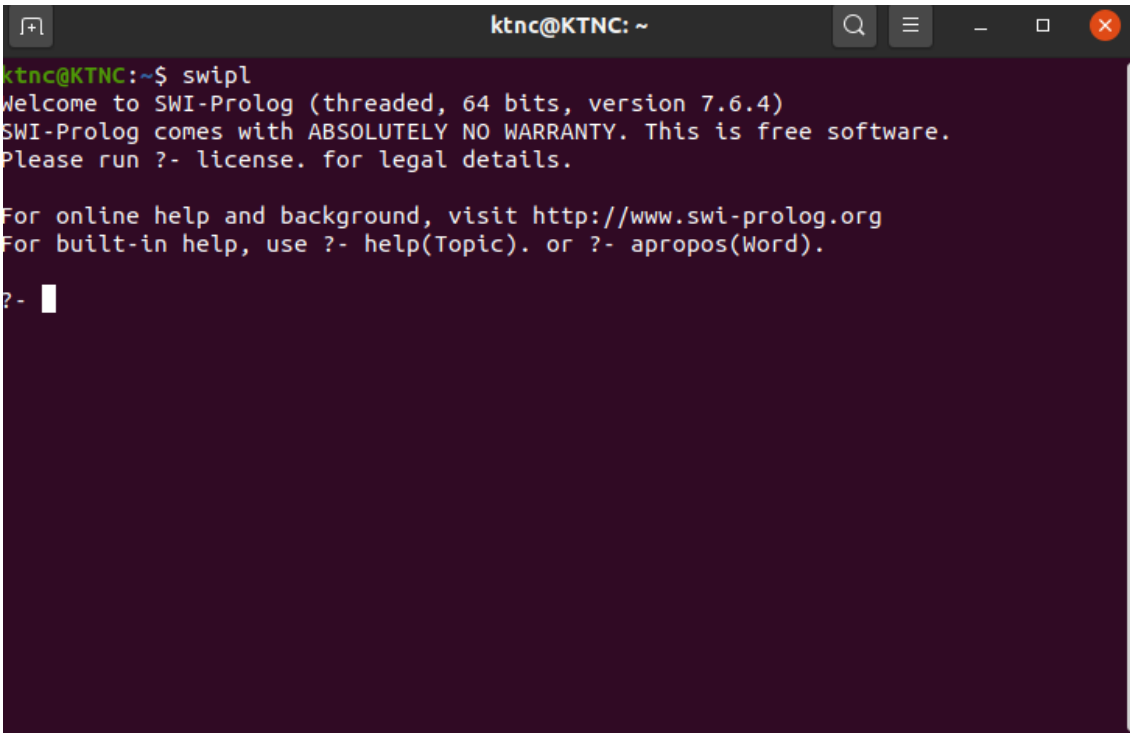
nền tảng như Windows, Unix, Macintosh, Linux. SWI là viết tắt của Sociaal-Wetenschappelijke Infomatica (“Social Science Infomatics").

2.2. Cách sử dụng SWI-Prolog:

2.2.1. Cài đặt và khởi động SWI-prolog

Để cài đặt SWI-Prolog, ta truy cập trang chủ, tải về bản cài đặt. Ngoài ra, ta có thể cài đặt trên nền Docker.

Trên môi trường Windows, SWI-Prolog cung cấp cho người dùng giao diện sử dụng. Trên Linux, ta chỉ có thể dùng SWI-Prolog thông qua dòng lệnh. Để mở SWI-Prolog bằng dòng lệnh, ta sử dụng câu lệnh **swipl**. Khi đó, giao diện SWI-Prolog hiện ra với các thông tin giới thiệu. Ngoài ra ta có thể mở một tập tin trong lúc khởi động bằng cách thêm tên vào argument.



Hình 2.2.1.a. Mở giao diễn prolog từ terminal.

Các tập tin chương trình Prolog có phần mở rộng “.pl" hoặc “.pro" khi có xung đột về tên mở rộng tập tin giữa các phần mềm.

Ta cũng có thể sử dụng lênh pwd và cd để kiểm tra vùng làm việc hiện tại và thay đổi thư mục làm việc.

Để mở một tập tin vào SWI-Prolog, ta sử dụng lệnh **[‘path/to/ten-tap-tin.pl']**. Có một vài tập tin đặc biệt trong một project SWI-Prolog mà ta cần lưu ý:

- Load.pl: được dùng để cài đặt môi trường và mã nguồn. Ngoài ra nó còn được dùng để parse lệnh command line và chạy chương trình.
- Run.pl: được dùng để thực thi chương trình. Thông thường, tập tin này sẽ gọi load.pl và thực thi chương trình từ load.pl.

- Save.pl: dùng để tạo và lưu trạng thái hiện tại của chương trình cùng với lệnh qsave_program/2 với các tùy chọn khác nhau.
- Debug.pl: dùng cho việc debug. Ngoài ra ta có thể tùy biến khả năng in ra màn hình các thông tin trong lúc debug.

Mặc dù ta nên cài đặt các rule trong file nhưng trong một vài trường hợp cần thiết, ta có thể thêm các luật trực tiếp trong chương trình bằng cách sử lệnh **[user].**, sau đó nhập lần lượt từng luật vào và khi kết thúc ta nhấn Ctrl + D.

```
ktnc@KTNC:~/Prolog$ swipl
Welcome to SWI-Prolog (threaded, 64 bits, version 7.6.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit http://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- [user].
|: a.
|: b.
|: c.
|: ^D% user://1 compiled 0.00 sec, 3 clauses
true.

?-
```

Hình 2.2.1.b. Nhập các luật(rule) vào prolog.

Để thực hiện truy vấn trong chương trình, ta sử dụng các biến trong Prolog. Hệ thống sẽ thực hiện truy vấn và trả về kết quả X = <giá trị> nếu có. Khi đó người dùng có thể nhấn nút ";" hoặc dấu cách để tìm kết quả khác, ngược lại nhấn Enter để trở về. Một câu truy vấn kết thúc bằng dấu chấm khi không còn hoặc trả về false. khi không có kết quả.

```
ktnc@KTNC:~/Prolog$ swipl
Welcome to SWI-Prolog (threaded, 64 bits, version 7.6.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit http://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- ['first.pl'].
true.

?- giaovien(A).
A = a .

?- giaovien(x).
false.

?-
```

Hình 2.2.1.c. Thực hiện các truy vấn trong Prolog.

2.2.2.Chu trình test-edit-reload

SWI-Prolog không yêu cầu người dùng sử dụng một trình editor cụ thể. Ta có thể xác định trình editor cho Prolog bằng Prolog flag editor. Tuy nhiên ta vẫn có thể sử dụng trình Editor có sẵn của SWI-Prolog là PceEmac.

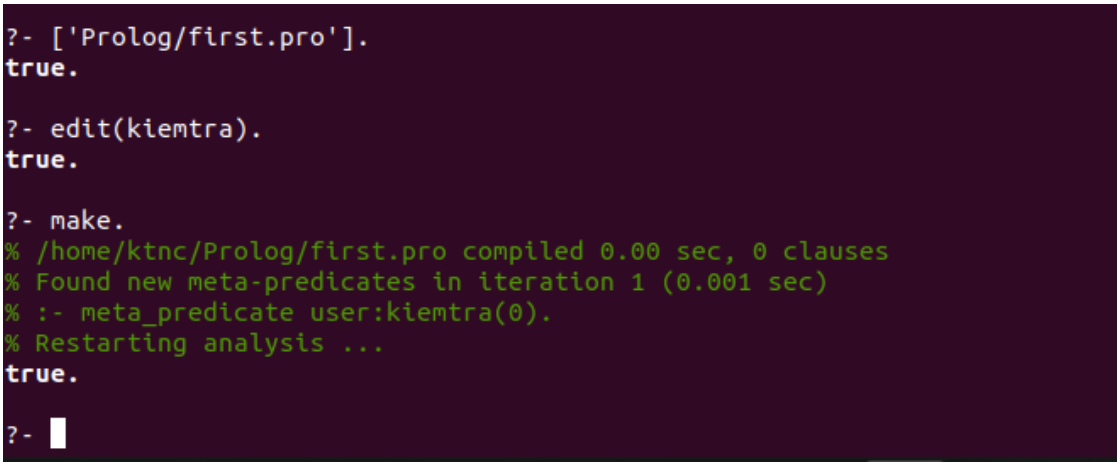
Hình 2.2.2.a. Demo Sử dụng Editor PceEmac – Editor có sẵn của Prolog

Một điểm đặc biệt của Prolog là việc cho phép người dùng chỉnh sửa mã nguồn ngay trong lúc chạy. Để xác định vị trí cần chỉnh sửa, ta dùng lệnh edit(X) (với X là tên file, tên module hoặc tên phương thức). Khi có nhiều hơn một kết quả, ta có thể chọn nơi cần chỉnh sửa.

Ví dụ:

```
?- edit(country).
Please select item to edit:
1 chat:country/10'/home/jan/.config/swi-prolog/lib/chat/countr.pl':16
2 chat:country/1'/home/jan/.config/swi-prolog/lib/chat/world0.pl':72
Your choice?
```

Khi thực hiện test chương trình và gặp lỗi, ta có thể sửa trực tiếp bằng lệnh edit và reload thực hiện chạy lại chương trình. Dùng lệnh **make** ta sẽ kiểm tra được phần nào đã được chỉnh sửa và sao đó tiến hành reload lại phần đó.



Hình 2.2.2.b. Sửa chương trình trực tiếp từ terminal, dùng lệnh edit, make

2.2.3. PceEmacs

Đây là một công cụ editor có sẵn trong SWI-Prolog được xây dựng dựa trên GNU-Emacs, thích hợp cho người dùng mới.

2.2.3.1. Edit modes

Modes là trái tim của Emacs, mỗi modes khác nhau tương ứng với từng loại loại tập tin nguồn. Để chỉnh sửa Prolog, ta cần sử dụng Prolog mode.

Để sử dụng Prolog mode trong Emacs, ta có thể dùng các cách:

- Mở tập tin với tên mở rộng là .pl, .pro. Emacs sẽ tự động chuyển sang Prolog mode.
- Chèn vào dòng đầu tập tin chuỗi ký tự "-*- *Prolog* -*-".
- Vào File/Mode chọn Prolog.

2.2.3.2. Các lệnh cơ bản

Cut/Copy/Paste:

- Copy: double click chuột trái vào từ cần copy hoặc tô đen vào phần cần copy và nhấn tổ hợp Ctr + Shift + C.
- Cut: tô đen phần cần cắt và nhấn phím Del.
- Paste: dùng chuột giữa hoặc chọn Edit/Paste.

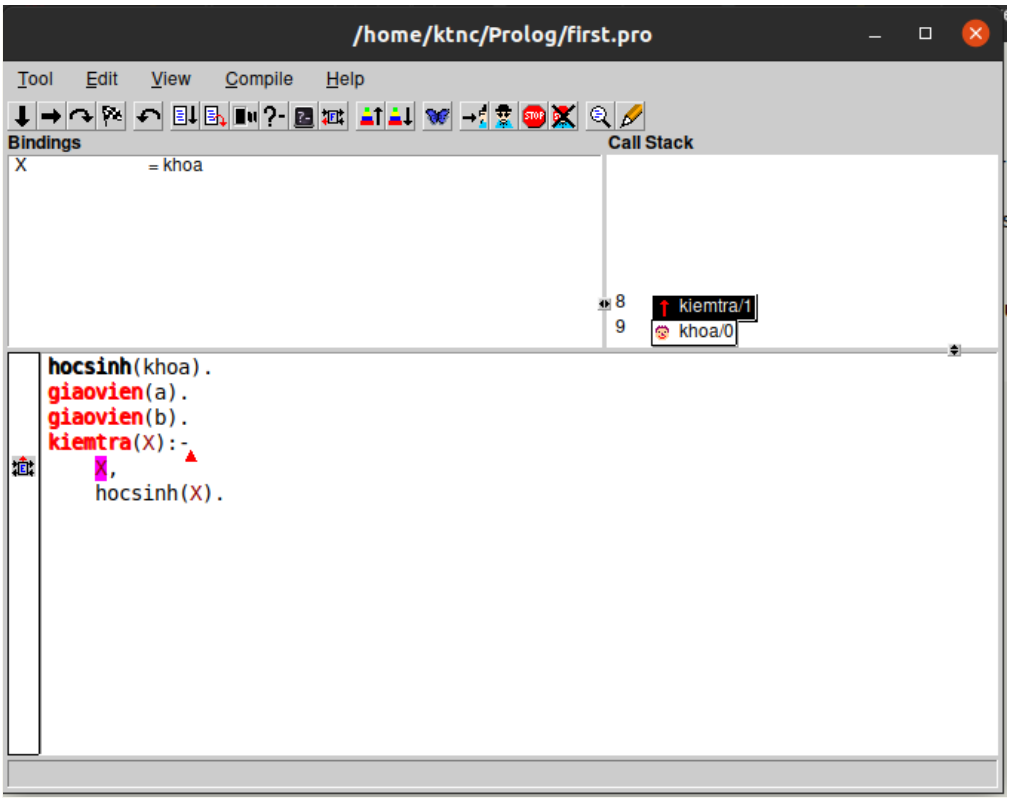
Undo: Sử dụng tổ hợp phím Ctr + Z.

Tìm kiếm: dùng tổ hợp phím Ctr + S (tìm kiếm tới) hoặc Ctr + R (tìm kiếm lui), sau đó gõ nội dung cần tìm kiếm.

2.2.4. Graphical Debugger

SWI-Prolog cung cấp cho người dùng hai trình Debugger khác nhau: console debugger và graphical debugger. Để dễ dàng hơn trong lúc debug, ta nên sử dụng graphical debugger.

Ta sử dụng lệnh **guitracer.** Để cài đặt chế độ thực thi debug, ta dùng lệnh **trace**. hoặc **spy**.

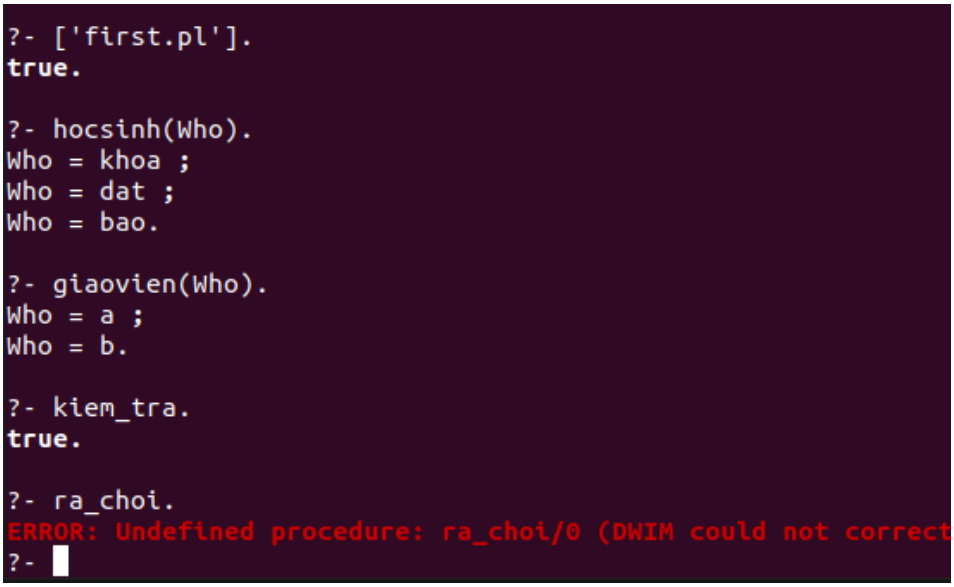


Hình 2.2.4. Debug trên môi trường SWI-Prolog

2.3.Ví dụ minh hoạ

2.3.1. Ví dụ 1:

- Tiến hành truy vấn trên cơ sở dữ liệu sau:
- hocsinh(khoa).
- hocsinh(dat).
- hocsinh(bao).
- giaovien(a).
- giaovien(b).
- kiem_tra.
- ra_ve

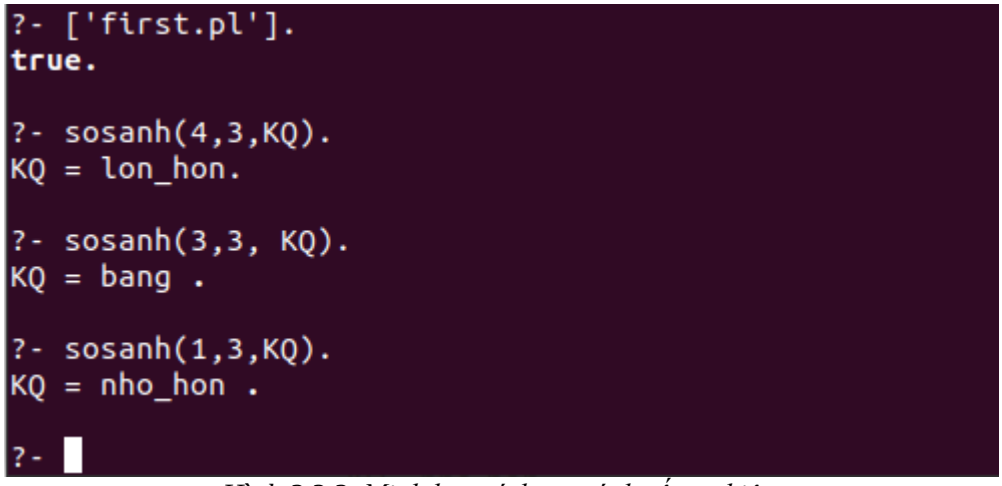


Hình 2.3.1. Minh hoạ ví dụ 1

2.3.2. Ví dụ 2:

So sánh hai số:

- sosanh(A, B, KQ):-
- A == B,
- KQ = bang.
- sosanh(A, B, KQ):-
- A < B,
- KQ = nho_hon.
- sosanh(A, B, KQ):-
- A > B,
- KQ = lon_hon.



Hình 2.3.2. Minh hoạ ví dụ so sánh số tự nhiên.

2.3.3. Ví dụ 3:

Tính a^b: ta có thể sử dụng đệ quy để tính:

- mu(A, 1, R):- R is A.
- mu(A, 0, R):- R is 1.
- mu(A, B, R):-
- B > 1,
- Nl is B - 1,
- mu(A, Nl, Rl),
- R is A*Rl.

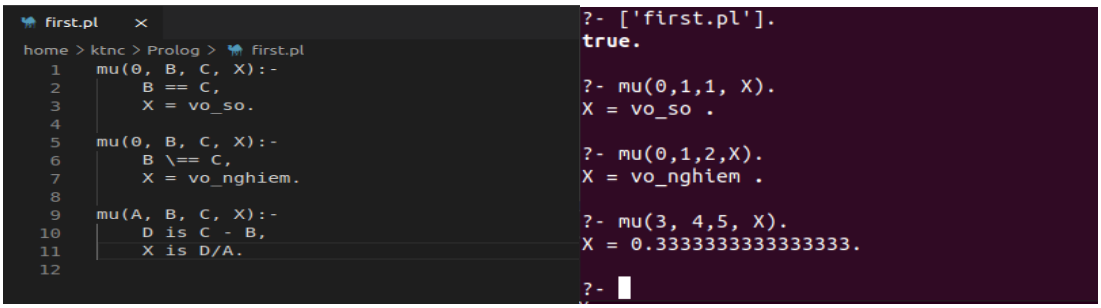
```
?- ['first.pl'].
true.

?- mu(2,3,W).
W = 8 ;
false.

?- mu(9,3, H).
H = 729
```

Hình 2.3.3. Minh hoạ ví dụ a^b.

2.3.4. Ví dụ 4:

Tìm nghiệm của phương trình ax + b = c

- mu(0, B, C, X):-
- B == C,
- X = vo_so.
- mu(0, B, C, X):-
- B \== C,
- X = vo_nghiem.
- mu(A, B, C, X):-
- D is C - B,
- X is D/A.

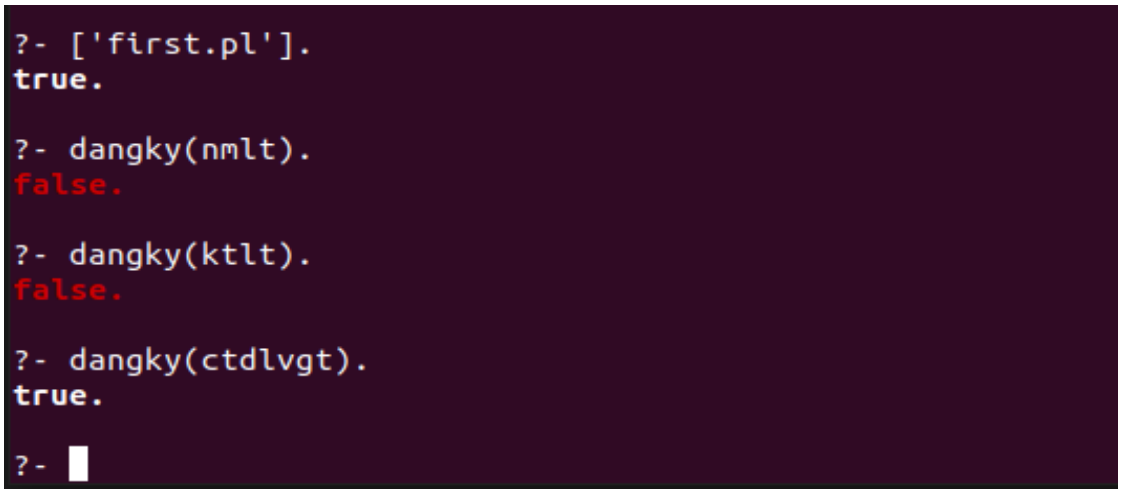


Hình 2.3.4. Minh hoạ ví dụ tìm nghiệm của phương trình bậc nhất ax + b = c

2.3.5. *Ví dụ 5:*

Cho cơ sở trí thức như sau: Câu điều kiện dangky(X) cho biết người học có được đăng ký một môn X hay không. Câu điều kiện tienquyet(A, B) cho biết môn học A là môn học tiên quyết của môn học B. Câu điều kiện quamon(X) cho biết người học đã qua môn X chưa.

- dangky(Monhoc):-
- not(quamon(Monhoc)),
- tienquyet(MHTQ, Monhoc),
- quamon(MHTQ).
- tienquyet(nmlt, ktlt).
- tienquyet(ktlt, ctdlvgt).
- quamon(nmlt).
- quamon(ktlt).



Hình 2.3.4. Minh hoạ ví dụ dùng Prolog trên cơ sở dữ liệu.

PHẦN 3: TÀI LIỆU THAM KHẢO

- https://vi.wikipedia.org/wiki/Prolog
- http://ent.htu.edu.vn/tin-tuc-su-kien/ngon-ngu-lap-trinh-prolog.html
- http://www.mind.ilstu.edu/curriculum/protothinker/prolog_intro.php
- https://www.cs.ru.nl/~peterl/teaching/CS3510/intro-prol.pdf
- Appendix A - An Introduction to Prolog.
- https://www.swi-prolog.org/
- https://www.swi-prolog.org/pldoc/doc_for?object=manual
- https://www.doc.gold.ac.uk/~mas02gw/prolog_tutorial/prologpages/