Cut in prolog

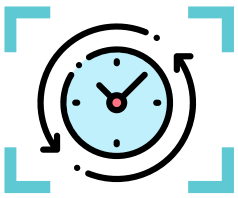
عملي مشترك

14/5/2018

RB Informatics;

مبادئ الذكاء الصنعي

محتويات المحاضرة



Backtracking

Cut in prolog

Fail

تمارين

مفهوم القطع في البرولوغ (cut)

يقوم برنامج prolog بتجربة جميع الحلول سواء كانت صحيحة أم خاطئة ويقوم برد أول حل صحيح مع احتفاظه ببقية الحلول، فإذا قام بطباعة الحل وبعده نقطة يكون الحل وحيد أما إذا طبع الحل بدون نقطة عندها يكون لديه عدد من الحلول نستطيع رؤيتها بالضغط على الفاصلة المنقوطة (;).

تعني الـ cut أنه إذا وجد حل صحيح يتوقف البرنامج عن التجربة في باقي القواعد ويردّه.

يرمز للـ cut في prolog بالرمز (!).

مثال للتوضيح

إجرائية الـ sum:

```
sum([],0).
sum([H|T],S):- is_list(H),sum(H,S1),sum(T,S2),S is S1+ S2;number(H),sum(T,S3), S is
S3+H.
```

لنجرّب المثال التالي:

```
?- sum([[1,2],3,4,5],A).
A = 15 ;
false.
```

نلاحظ أن الخرج كان بدون نقطة لأن الحل ليس وحيد وعندما جرّبنا استخدام الفاصلة المنقوطة يعطي false.

الحل ليس وحيد أي أن H ممكن أن يكون سلسة وممكن أن يكون رقم.

استخدام الـ cut في هذه الإجرائية:

```
sum([],0).
sum([H|T],S):- is_list(H),!,sum(H,S1),sum(T,S2),S is S1+ S2;number(H),sum(T,S3), S i
s S3+H.
```

إشارة التعجب تعني cut أي إذا وجد حل صحيح يتوقف عن التجربة في باقي القواعد ويردّه.

نجرّب المثال السابق بعد إضافة الـ cut للقاعدة:

```
?- sum([[1,2],3,4,5],A).
A = 15.
```

نلاحظ أن الجواب مع نقطة وهذا لأن البرنامج وجد حل صحيح فردّ القيمة 15 وتوقف عن تجربة باقي القواعد.

مثال آخر

لدينا القاعدة التالية بدون استخدام الـ cut:

```
a(X,Y):- b(X),c(Y).
b(1).
b(2).
b(3).

c(1).
c(2).
c(3).
```

الخرج:

```
?- a(X,Y).
X = Y, Y = 1;
X = 1,
Y = 2;
X = 1,
Y = 3;
X = 2,
Y = 1;
X = Y, Y = 2;
X = 2,
Y = 3;
X = 3,
Y = 1;
X = 3,
Y = 2;
X = Y, Y = 3.
```

نستخدم cut بعد الحقيقة الأولى:

```
a(X,Y):-b(X),!,c(Y).
```

الخرج:

```
?- a(X,Y).
X = Y, Y = 1;
X = 1,
Y = 2;
X = 1,
Y = 3.
```

نضيف cut بعد الحقيقة الثانية:

```
a(X,Y):-b(X),c(Y),!.
```

الخرج:

```
?- a(X,Y).
X = Y, Y = 1.
```

تمارين

1 إيجاد عدد مرات تكرار عنصر في مجموعة عميقة (مجموعة قد تحتوي على عناصر و مجموعات):

```
count(_,[],0).
count(X,[H|T],C):-is_list(H),!,count(X,H,C1),count(X,T,C2),C is C1+C2.
count(X,[H|T],C):-X=H,!,count(X,T,C1),C is C1+1;count(X,T,C).
```

التنفيذ:

```
?- count(1,[1,1,[1,2],3,4],L).
L = 3.
```

فكرة الحل:

السطر الأول شرط التوقف إذا كانت السلسلة فارغة يرد صفر.

السطر الثاني إذا كان رأس السلسلة هو سلسلة أيضاً نستدعي الإجرائية عنده وعند باقي السلسلة ويكون عدد مرات تكرار العنصر X يساوي مجموع خرجي الاستدعائين السابقين.

السطر الثالث إذا كان العنصر مساوياً لرأس السلسلة نستدعي الإجرائية عند باقي السلسلة ويكون الخرج مساوياً خرج الاستدعاء السابق +1 وإلا يستدعي الإجرائية عند باقي السلسلة ولا يزيد للخرج 1.

2 إيجاد مجموع عناصر مجموعة عميقة بحيث نرد المجموع بعد ضرب كل عنصرين متجاورين مع بعضهما:

```
sum([],0):-!.
sum([H|[]],H):-!.
sum([H|[H1|T]],R):-sum(T,R1), R is (H*H1)+R1.
```

التنفيذ:

```
?- sum([1,2,3,4],L).
L = 14.

?- sum([1,2,3,4,5],L).
L = 19.
```

فكرة الحل:

شرطي التوقف: السطر الأول شرط التوقف إذا كانت السلسلة فارغة يرد صفر، السطر الثاني إذا كانت السلسة مكونة من عنصر وحيد يرد رأس السلسة.

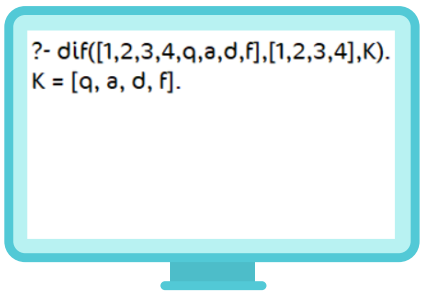
السطر الثالث نقوم بتجزئة السلسلة إلى رأس و بقية ثم نجزء باقي السلسة إلى رأس وبقية أيضاً وبذلك نكون قد حصلنا على أول عددين في السلسلة حيث نستدعي الإجرائية عند بقية العناصر ويكون الخرج مساوية لخرج بقية العناصر مضافاً إليه جداء أول عنصرين فيها وفي حال كان عدد العناصر زوجي سنتوقف عند الشرط الأول أما إذا كان فردي سيتوقف عند الشرط الثاني حيث يبقى آخر عنصر وحيد ولا يوجد عنصر آخر لضربه معه فنرده كما هو.

3 إيجاد الفرق بين سلسلتين باستخدام ال cut:

المقصود بالفرق بين سلسلتين أي طباعة العناصر الموجودة في سلسة وغير موجودة في الأخرى.

```
dif([],_,[]).
dif(L,[],L).
dif([H|T],M,R):-not(member(H,M)),R=[H|R1],dif(T,M,R1);dif(T,M,R),!.
```

التنفيذ:



فكرة الحل:

شرط التوقف الأول إذا كانت السلسلة الأولى فارغة والثانية مهما كانت سيقوم برد سلسلة فارغة.

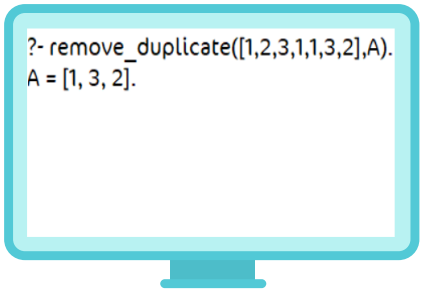
شرط التوقف الثاني إذا كانت السلسلة الثانية فارغة سيقوم برد ما تحويه السلسلة الأولى.

السطر الثالث إذا كان رأس السلسلة الأولى غير (not) موجود (ليس عنصر من عناصر السلسلة الثانية) سيقوم بإضافته إلى سلسلة الخرج واستدعاء الإجرائية عودياً عند باقي السلسة وصولاً إلى أحد شرطي التوقف وإلا: سيقوم باستدعاء الإجرائية فقط دون إضافته إلى سلسلة الخرج.

4 حذف العناصر المكررة في سلسلة باستخدام cut:

```
remove_duplicate([],[ ]).
remove_duplicate([H|T],R):- member(H,T),!,remove_duplicate(T,R);
R=[H|R1],remove_duplicate(T,R1).
```

التنفيذ:



فكرة الحل:

السطر الأول: شرط التوقف إذا كانت السلسلة فارغة سيرد سلسلة فارغة.

السطر الثاني إذا كان رأس السلسلة عضو في بقية السلسلة سيقوم باستدعاء الإجرائية عودياً عند بقية السلسة دون إضافة رأس السلسة إلى سلسلة الخرج وإلا سيستدعي الإجرائية عودياً ويقوم بإضافته إلى سلسلة الخرج.

Backtracking

التراجعية هي خاصية ملازمة لبرنامج prolog فعند تنفيذ أي برنامج عليه يقوم بتجربة جميع الحلول الممكنة.

مثال

لتكن لدينا القاعدة التالية:

```
Mutual_preference (X) :- likes( mary,X ) , likes( john,X ).
```

ومجموعة الحقائق التالية:

```
likes(mary,food).
likes(mary,tea).
likes(john,tea).
likes(john,mary).
```

تنفيذ البرنامج السابق سيكون بالشكل التالي:

أولاً:

```
Mutual_preference (X) :- likes( mary,X ),likes( john,X ).
```

likes(mary,food).
likes(mary,tea).
likes(john,tea).
likes(john,mary).

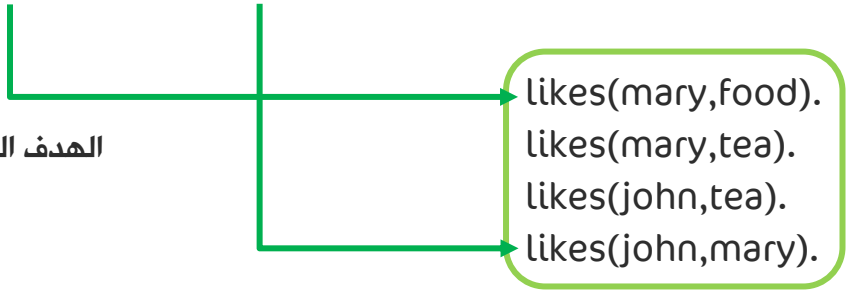
أول هدف محقق x=food

نجرب الحل من أجل الهدف الثاني

likes(john,food)

ثانياً:

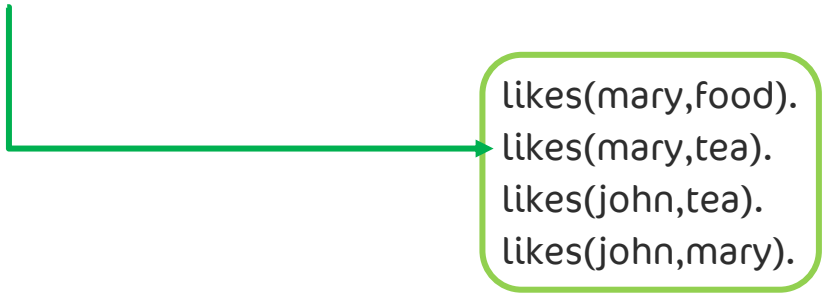
```
Mutual_preference (X) :- likes( mary,X ),likes( john,X ).
```



الهدف الثاني فشل لذلك نتراجع لتجربة
حل آخر

ثالثاً:

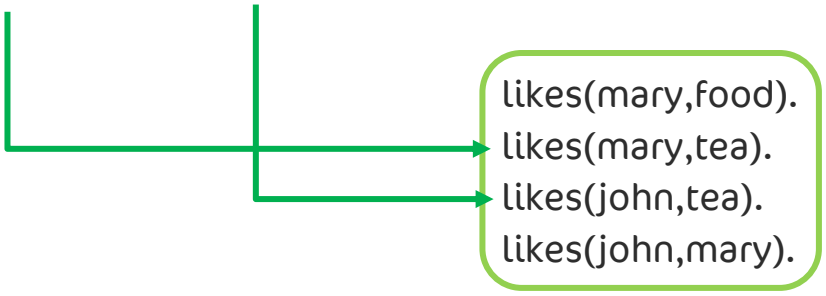
```
Mutual_preference (X) :- likes( mary,X ),likes( john,X ).
```



أول هدف محقق مرة ثانية x=tea
نجرب الحل من أجل الهدف الثاني

رابعاً:

```
Mutual_preference (X) :- likes( mary,X ),likes( john,X ).
```



ثاني هدف محقق أيضاً وبالتالي
وصلنا إلى حل ناجح.

في حال الفشل يتراجع إلى الهدف السابق لتجربة خيار جديد.

Fail

- رمز خاص يفشل على الفور عندما يصادف Prolog هدفه.
- عندما يفشل الـ prolog يجرّب الـ backtracking ولهذا تعتبر تعليمة fail لإجبار prolog على عمل backtracking.
- عند استخدام fail مع cut تسمح لنا بتعريف استثناءات للقواعد العامة.

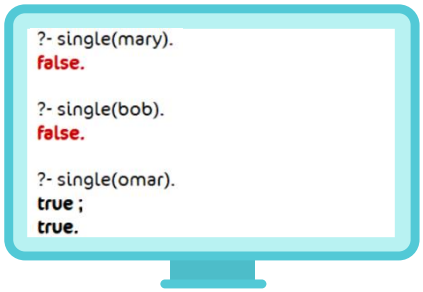
مثال

استخدام cut:

```
married(peter,lucy).
married(paul,mary).
married(bob,juliet).
married(harry,geraldine).
married(Person):- married(Person, _),! ; married(_, Person).

single(Person) :- \+ married(Person, _),\+ married(_, Person).
single(Person):- not( married(Person)).
```

التنفيذ:



استخدام cut و fail:

```
married(peter,lucy).
married(paul,mary).
married(bob,juliet).
married(harry,geraldine).
married(Person):- married(Person, _),! ; married(_, Person).
single(P) :- married(P), ! ,fail.
single(_).
```

التنفيذ:

ملاحظة

cut تجبر البرنامج على عدم عمل backtrack بينما fail تجبره على عمل backtrack

تمارين

❖ اكتب برنامج يقوم بفحص سلسلة إن كانت سلسلة جزئية من سلسلة أخرى:

```
is_sub(_,[]).
is_sub(L,[H|T]):-member(H,L),is_sub(L,T).
```

يقوم البرنامج السابق بفحص إن كانت السلسلة [H|T] هي سلسلة جزئية من السلسلة L وذلك بالمرور على عناصر سلسلة كلها وفي كل مرة فحص انتماء العنصر H إلى السلسة L باستخدام التابع member(,)

❖ اكتب برنامج يقوم بحذف عنصر معين من سلسلة:

```
remove([],_,[]).
remove([H|T],X,R):- X=:=H,remove(T,X,R1),R=R1,!;remove(T,X,R1),R=[H|R1].
```