بسمه تعالی

هوش مصنوعی

پرولوگ - ۴

نیمسال دوم ۱۴۰۱-۱۴۰۰

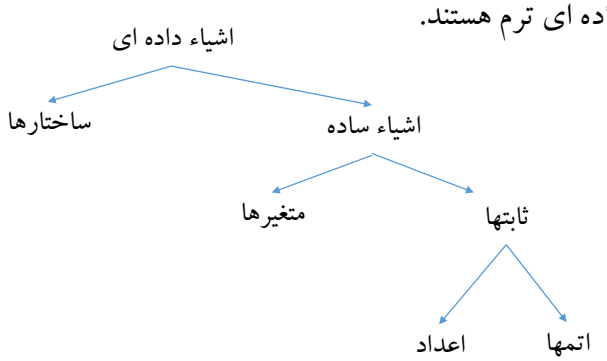
دکتر مازیار پالهنگ
آزمایشگاه هوش مصنوعی
دانشکدهٔ مهندسی برق و کامپیوتر
دانشگاه صنعتی اصفهان

یادآوری

- نصب نرم افزار
- طریقهٔ استفاده
- نوشتن واقعیات
- نوشتن قوانین ساده
- سؤال کردن
- قوانین بازگشتی
- متغیرها، متغیر گمنام
- فصل
- لیستها
- ریاضی
- کنترل عقبگرد
- نقیض به عنوان شکست

تعدادی از روالهای درونی پرولوگ

- در پرولوگ همهٔ اشیاء داده ای ترم هستند.



- گاهی مفید است که در طول برنامه چک کنیم نوع ترم چیست.
- برای مثال اگر می خواهیم دو متغیر X و Y را با هم جمع کنیم:

```
Z is X+Y
```

- قبل از اینکه این هدف انجام گیرد X و Y باید عدد صحیح یا اعشاری باشند.
- اگر مطمئن نیستیم می توانیم چک کنیم.

```
..., integer(X), integer(Y), Z is X+Y
```

- اگر X و Y هر دو عدد صحیح نباشند عمل جمع انجام نمی شود.

- برخی روالهای دیگر:
- این هدف موفق می شود اگر X فعلاً یک متغیر نمونه گذاری نشده باشد. var(X)
- این هدف موفق می شود اگر X ترمی به غیر از یک متغیر یا X هم اکنون یک متغیر نمونه گذاری شده باشد. nonvar(X)

- درست است اگر X فعلاً به یک اتم دلالت کند (در پرولوگ به ثابتها اتم گفته می شود). atom(X)
- درست است اگر X فعلا یک عدد صحیح باشد. integer(X)

- درست است اگر X فعلاً یک عدد اعشاری باشد. float(X)
- درست است اگر X فعلاً یک عدد صحیح یا اعشاری باشد. number(X)
- درست است گر X فعلاً به یک عدد یا یک اتم دلالت کند. atomic(X)
- مثال:

```
Z=2, integer(Z), nonvar(Z)?
Z=2
```

```
var(Z), Z=2?
Z=2
Z=2, var(Z)?
false
integer(Z), Z=2?
false
```

- به سه روش می توان اتم (ثابت) ساخت:
- رشته ای از حروف، ارقام و underscore که با حرف کوچک شروع شده باشد.
- رشته ای از کاراکترهای خاص همانند <-->
- رشته ای از کاراکترها که درون نقل قول یگانه قرار گرفته اند.

atom(22)?
false

atomic(22)?
true

atom(==>)?
true

atom(p(1))?
false

atom('salam')?
true

- مثال: تعداد استفاده از یک اتم در یک لیست از اشیاء

count(A,L,N)

```
count(_,[],0).
count(A,[A|T],N) :- !, count(A,T,N1), N is N1+1.
count(A,[_|T],N) :- count(A,T,N).
```

count(a,[a,b,a,a],N)?
N=3

count(a,[a,b,X,Y],Na)?
Na=3

count(b,[a,b,X,Y],Nb)?
Nb=3

L=[a,b,X,Y], count(a,L,Na),count(b,L,Nb)?
Na=3
Nb=1
X=a
Y=a

- در مثال آخر X و Y هر دو به a نمونه گذاری شدند و به همین دلیل Nb=1
- اگر علاقمند به شمارش تعداد واقعی یک اتم در یک لیست باشیم نه تعداد ترمهائی که با این اتم منطبق می شوند:

```
count(_,[],0).
count(A,[B|T],N) :- atom(B), A=B, !, count(A,T,N1), N is N1+1.
count(A,[_|T],N) :- count(A,T,N).
```

ساختن و تجزیهٔ ترمها

- .

f(a,b) =.. L?
L=[f,a,b]

T =.. [rectrangle,3,5]?
T= rectangle(3,5)

Z =.. [p,X,f(X,Y)]?
Z = p(X,f(X,Y)

ساختن و تجزیهٔ ترمها

functor(Term,F,N)

arg(N,Term,A)

functor(t(f(X),X,t), Fun, Arity)?
Fun = t
Arity = 3

arg(2,f(X,t(a), t(b)), Y)?
Y = t(a)

functor(D,date,3), arg(1,D,22),
arg(2,D,Tir),
arg(3,D,1399)?

D = date(3,Tir,1399)

ساختن و تجزیهٔ ترمها

- Name(A,L)

?- name(amin,L).
L = [97, 109, 105, 110].

کار با پایگاه دانش

- جملهٔ C به پایگاه دانش اضافه می شود. assert(C)
- جمله ای که با C منطبق شود را حذف می کند. retract(C)
- جملهٔ C را به ابتدای پایگاه دانش اضافه می کند. asserta(C)
- جملهٔ C را به انتهای پایگاه دانش اضافه می کند. assertz(C)

-happy?
false

-assert(happy)?
true

-happy?
true

-retrct(happy)?
true

-happy?
false

.•

fast(amin).
slow(amir).
slow(nasser).

-assert((faster(X,Y) :- fast(X), slow(Y)))?
true

-faster(A,B)?
A = amin
B = amir;
A = amin
B = nasser.

-retrct(slow(X))?
X = amir;
X = nasser;
false

-faster(amin,_)?
false

.•

-assert(p(a)), assertz(p(b)), asserta(p(c))?
ture

-p(X)?
X = c;
X = a;
X = b.

جمع آوری پاسخها

- تولید لیست L شامل همهٔ اشیاء X که هدف P را ارضاء می کنند.

bagof(X,P,L)

```
age(amir,7).
age(karim,5).
age(nasser,8).
age(amin,5).

?- bagof(Child, age(Child,5),List).
List = [karim,amin].
```

```
?- bagof(Child,age(Child,Age), List).
Age = 5,
List = [karim, amin] ;
Age = 7,
List = [amir] ;
Age = 8,
List = [nasser].
```

- اگر مقادیر Age برایمان مهم نباشد:

```
?- bagof(Child,Age^age(Child,Age), List).
List = [amir, karim, nasser, amin].
```

جمع آوری پاسخها

- اگر هیچ حلی برای P در هدف bagof(X,P,L) وجود نداشته باشد، آنگاه هدف bagof شکست می خورد.
- اگر شئ مشابه X مکرراً یافت شود، همهٔ رخدادهای آن در L ظاهر خواهند شد.
- مسند setof(X,P,L) همانند bagof می باشد فقط این بار لیست L مرتب شده خواهد بود و موارد تکراری اگر وجود داشته باشند حذف خواهند شد.

```
?- setof(Child,Age^age(Child,Age),ChildList),
|    setof(Age,Child^age(Child,Age),AgeList).
ChildList = [amin, amir, karim, nasser],
AgeList = [5, 7, 8].
```

جمع آوری پاسخها

- مسند دیگر findall(X,P,L) همانند bagof(X,P,L) می باشد، با این تفاوت که تمامی اشیاء X را جمع آوری می کند بدون توجه (احتمالاً) حلهای متفاوت برای متغیرهای درون P که با X مشترک نیستند.

```
?- findall(Child,age(Child,Age),List).
List = [amir, karim, nasser, amin].
```

- اگر پاسخی یافت نشد لیست تهی بازمی گرداند.

جمع آوری پاسخها

- در صورتی که findall در نسخه پرولوگی وجود نداشته باشد براحتی می توان آن را پیاده سازی نمود.

```
findall(X,Goal,Xlist) :- call(Goal), assertz(queue(X)), fail;
                         assertz(queue(bottom)), collect(Xlist).

collect(L) :- retract(queue(X)), !, (X == bottom, !, L=[];
                                     collect(Rest), L = [X|Rest]).
```

جستجوی عمق نخست

- حالت ساده:

```
depthFirstSimple(N, [N]):-
   goal(N).
depthFirstSimple(N, [N | Sol1) :-
   s(N,N1),
   depthFirstSimple(N1, Sol1).
```

جستجوی عمق نخست

- جستجوی گرافی:

```
% Solution is an acyclic path in reverse order between start Node and
% goal.
% Adapted from Prolog Programming for Artificial Intelligence, by
% I.Bratko
%
dfSolve(Node, Solution) :- depthFirst([],Node,Solution).

% Path is the current path found so far.
depthFirst(Path, Node, [Node|Path]) :-
   goal(Node).

depthFirst(Path, Node, Sol) :-
   s(Node, Node1),
   not(member(Node1,Path)),
   depthFirst([Node|Path],Node1,Sol).
```

جستجوی عمق محدود شده

.•

```
depthLimitedFirst(Node, [Node],_) :- goal(Node), !.

depthLimitedFirst(Node, [Node | Sol], Maxdepth) :-
   Maxdepth > 0,
   s(Node,Node1),
   Max1 is Maxdepth-1,
   depthLimitedFirst(Node1,Sol,Max1).
```

جستجوی عرض نخست

```
% Adapted from Prolog Programming for Artificial Intelligence, by
% I.Bratko
% Solution is in Reverse order.
solve(Start,Solution) :- breathFirst([[Start]], Solution).

breathFirst([[Node | Path] | _], [Node | Path]):-
   goal(Node).

breathFirst([Path | Paths], Solution):-
   extend(Path, NewPaths),
   conc(Paths, NewPaths, Paths1),
   breathFirst(Paths1, Solution).

extend([Node|Path], NewPaths) :-
   bagof([NewNode, Node | Path],
       (s(Node, NewNode), not(member(NewNode, [Node|Path]))),
        NewPaths), !.
extend(Path, []).
```

خطازدائی

- امکان استفاده از دستور spy(p) برای ردگیری دستورهائی که پس از رسیدن به مسند p انجام می شوند.

چند تمرین

- برنامه ای به زبان پرولوگ بنویسید که لیست L را گرفته و وارون آن لیست InvL را ایجاد کند.

  reverse(L,InvL)
- برنامه ای به زبان پرولوگ بنویسید که چک کند آیا لیست L به ترتیب صعودی مرتب است یا خیر.

  sorted(L)

- محیط دنیای دیو را با پرولوگ پیاده سازی نمائید و سلولهائی که عامل باید در این محیط طی نماید تا طلا را یافته و از غار خارج شود را بیابید. امتیاز عامل را نیز محاسبه نمائید.
- جستجوی A* را با پرولوگ پیاده سازی نمائید.


والسلام
م.پالهنگ