Ivan Bratko

PROLOG PROGRAMMING FOR ARTIFICIAL INTELLIGENCE, 4TH EDN. ADDITIONAL EXERCISES IN DCG GRAMMARS

Note: These exercises supplementary to those in the book: I. Bratko, Prolog Programming for Artificial Intelligence, 4th edn., Pearson Education 2011.

Problem

Consider the DCG grammar:

```
s --> sa, sb.
sa --> [a].
sa --> [a, a], sa.
sb --> [b].
sb --> [b], sb.
```

(a) Give all sentences of length up to three symbols, generated by this grammar.

(b) Describe by a regular expression the set of sentences generated by this grammar .

(c) Add appropriate arguments to non-terminal symbols s, sa and sb, and modify the grammar rules so that the grammar will also compute the difference between the numbers of occurrences of symbols “a” and symbols “b” in the sentence. Also, the modified grammar should only allow sentences with the number of a’s greater than the number of b’s. For example, the modified grammar should produce the following answers:

```
?-  s( D, [a,a,a,a,a,b,b], [ ]).
D = 3
?-  s( D, [a,a,a,b,b,b], [ ]).
no
```

Answers

(a) [a,b], [a,b,b]

(b) $(aa)^{*}ab^{+}$

This means: 0 or more occurrences of “aa” followed by “a” followed by 1 or more occurrences of “ab”

(c)

```
s(D) --> sa(Na), sb(Nb), {Na > Nb, D is Na - Nb}.
sa(1) --> [a].
sa(N) --> [a, a], sa(N0), {N is 2 + N0}.
```

```
sb(1) --> [b].
sb(N) --> [b], sb(N0), {N is 1 + N0}.
```

Problem
Consider the following DCG grammar:

```
s --> [stop].
s --> [a], s.
s --> [b], s.
```

(a) Describe the language generated by this grammar.
(b) Modify this grammar by adding an argument N to non-terminal s in this grammar, so that N will be the number of occurrences of symbol 'a' in the sentence.

Problem
Consider the following DCG grammar:

```
s(N)  -->  n(N).
s(S)  -->  n(N), s(S0), { S is N + S0}.
n(1)  -->  [one].
n(2)  -->  [two].
n(3)  -->  [three].
```

What are Prolog's answers to the following questions (for each question give all Prolog's answers in the order in which the answers are generated by Prolog):

(a) ?- s(Sa, [three], []).

(b) ?- s(Sb, [one, two, three], []).

(c) ?- s(3, L, []).

Answers
(a) Sa = 3
(b) Sb = 6

(c)
  L = [three];
  L = [one, two];
  L = [one, one, one];
Next answer: Prolog gets in infinite loop!

Problem
Consider the following DCG grammar:

```
s( N, Dir) --> num(N).
s( N1, inc) --> num(N1), s(N2, inc), {N1 =< N2}.
s( N1, dec) --> num(N1), s(N2, dec), {N1 >= N2}.
num(N) --> [N], { N=1; N=2; N=3}.
```

What are Prolog's answers to the queries below, using this grammar?

```
(a)  ?-  s( N, Da, [2,3], [ ]).
(b)  ?-  s( N, Db, [2, 3, 4], [ ]).
(c)  ?-  s( N, Dc, [ 1, 3, 2], [ ]).
(d)  ?-  s( N, Dd, [2, 2, 2, 2]).     % Give all Prolog's answers
(e)  ?-  s( 2, De, [ N1, N2], [ ]).  % Give all Prolog's answers
```

Questions
Define the meaning in logic of the following sentences:

(a) Jimmy likes every girl.
(b) John knows all famous artists.
(c) John knows a pretty musician that plays Piazzolla.
(d) Each farmer has a dog and a cat that like each other.
(e) Every tourist from Japan that travels to Europe visits Venice and Vienna.

Answers
(a) all(X, girl(X) ==> likes(jimmy, X))
(b) all(X, artist(X) and famous(X) ==> knows(john, X))
(c) exist(X, knows(john, X) and musician(X) and pretty(X) and plays(X, piazzolla))
(d) all(X, farmer(X) ==>

exists(Y, dog(Y) and exists(Z, cat(Z) and likes(Y, Z) and likes(Z, Y))))
(e) all(X, tourist(X) and from (X, japan) and travels_to(X, europe) ==>
visits(X, venice) and visits(X, vienna))

Questions
Define the meaning in logic of the following sentences:

(a) A skier from Austria won a downhill race at Wengen.
(b) John and Mary like every artist who plays Mozart.
(c) A smart farmer sold all the horses he owned to an American turist.
(d) A black and a white horse participated in the Randwick race, and the black one won the race.

Answers
(a) exists(X, skier(X) and from(X,austria) and (exists(Y, downhill_race(Y) and at(Y,wengen) and won(X,Y)))
(b) all(X, artist(X) and plays(X,mozart) => likes(john,X) and likes(mary,X))
(c) exists(F, farmer(F) and smart(F) and exists(T, turist(T) and from(T,america) and all(H, horse(H) and owned(F,H) => sold(F,H,T))))
(d) exists(X, horse(X) and black(X) and participated_in(X,randwick_race) and
won(X) and exists(Y, horse(Y) and white(Y) and participated_in(Y,randwick_race)))

Questions
State the meaning in logic for the following sentences:

(a) Fritz went on a fishing trip and met a nice lady who teaches French.
(b) Every skier from Austria likes downhill.
(c) Mary fears all her neighbours.
(d) John has two students that play tennis.
(e) Every teacher that plays tennis drinks beer.
(f) Write an English sentence that has the following formal meaning:
all(X, dog(X) ==> all(Y, cat(Y) and black(Y) ==> chases(X, Y)))

Problem
Here is a very simple grammar:

```
sentence --> noun, verb.
noun --> [john].
noun --> [mary].
verb --> [sings].
verb --> [paints].
```

(a) Give two sentences generated by this grammar.

(b) Now we add the meaning of nouns and verbs into this grammar as follows:

```
noun( john) --> [john].
noun( mary) --> [mary].
verb( X, sings(X)) --> [sings].
verb( X, paints(X)) --> [paints].
```

Add the appropriately modified rule for sentence, so that the grammar will answer as illustrated by the example:

```
?-  sentence( M, [ mary, sings], [ ]).
M = sings( mary)
```

Answers
(a) [john, sings]; [mary, paints]
(b) sentence(VP) --> noun(X), verb(X, VP).

Problem
Consider the following DCG grammar:

```
s --> w.
s --> w, [and], s.
w --> [one].
w --> [two].
w --> [three].
```

What are Prolog's answers to the following queries (give all Prolog's answers):

(a) ?- s([one, and, three], []).
(b) ?- s([one, two, three], []).
(c) ?- s([X, Y, three], []).

Now the following rules are added to our grammar:

```
p --> s.
p --> s, [times], p.
```

(d) What will be all Prolog's answers to the query:

```
?- p([ one, P, two, Q, three], []).
```

(e) What is the intuitive meaning of this grammar?

Answers
(a) yes
(b) no
(c) X=one, Y=and;
X=two, Y=and;
X=three, Y=and
(d) P=and, Q=and;
P=and, Q=times;
P=times, Q=and;
P=times, Q=times
(e) Grammar of a kind of arithmetic expressions where numbers are written as 'one', 'two', 'three', and operators as 'and' and 'times'.

Problem
Consider English sentences like:

Ann likes music.
John likes tennis and music.
Mary likes football, tennis, music and ballet.
John and Mary enjoy tennis and music.
Ann, John and Mary like music.

(a) Define DCG grammar rules for **noun** and **composite_noun**, where the latter is a sequence of any number of nouns like “football, tennis, music and ballet” (nouns separated by commas, the last two nouns by “and”).

(b) Extend the grammar from (a) to handle sentences of the kind as shown above. Pay attention to singular and plural forms of verbs, and enforce number agreement between nouns, composite nouns and verbs by adding the argument Number as appropriate.

Problem

Consider the DCG grammar below:

```
s(M)  -->  w(M).
s(M1)  -->  w(M1), s(M2), {M1 =< M2}.
w(1)  -->  [one].
w(2)  -->  [two].
w(3)  -->  [three].
```

How will Prolog answer the following questions? When there are several answers possible, give the first two answers.

(a) ?- s(Ma, [two, two, three, three], []).
(b) ?- s(Mb, [two, one, three, three], []).
(c) ?- s(2, L, []).

Problem

Consider the DCG grammar rule that attempts to define the meaning of determiner ‘every’:

```
determiner( X, Prop, Assn, exists( X, Prop and Assn)  --> [every].
```

(a) Is this rule correct? If not, suggest a corrected rule.

(b) What is the role of the arguments Prop and Assn? Which phrases of a sentence will determine the values of Prop and Assn?

Answers

(a) Not correct. Correct is:

```
determiner( X, Prop, Assn, all( X, Prop ==> Assn) ) --> [every].
```

(b) Property is determined by noun phrase, Assn is determined by verb phrase. These two arguments are “slots” that provide access to parts of the meaning so that these parts can be filled in later from the context when noun phrase and verb phrase are known.