Prolog DCG Grammars

Computers Playing Jeopardy! Course
Stony Brook University
Backus–Naur Form (BNF) grammars

Grammars generate and recognise sentences and parse trees.

BNF grammar example: \(<s> ::= a \ b \mid a <s> b\)

Example sentence: “a a a b b b”
Definite clause grammars (DCG)

- A **DCG** is a way of expressing BNF grammars in a logic programming language such as Prolog.

- The definite clauses of a DCG can be considered a set of axioms where the fact that it has a parse tree can be considered theorems that follow from these axioms.
A Simple Natural Language DCG

The cat scares the mouse.

det | noun | verb | det | noun

noun_phrase | noun_phrase

verb_phrase

sentence
A Simple Natural Language DCG

sentence  -->  noun_phrase, verb_phrase.
verb_phrase  -->  verb, noun_phrase.
noun_phrase  -->  determiner, noun.
determiner  -->  [ the].
noun  -->  [ cat].
noun  -->  [ cats].
noun  -->  [ mouse].
verb  -->  [ scares].
verb  -->  [ scare].
    ?- sentence(X,[]).
    ?- trace, sentence([the,cat,scares,the,mouse],[]).
This Grammar Generates

[ the, cat, scares, the, mouse]

[ the, mouse, scares, the, mouse]

[ the, cats, scare, the, mouse]

[ the, cats, scares, the, mouse]

CONTEXT DEPENDENT!
DCG

- Not only context-free grammars.
- Context-sensitive grammars can also be expressed with DCGs, by providing extra arguments.
Number Agreement Can Be Forced By Arguments

sentence(Number) -->
    noun_phrase(Number), verb_phrase(Number).

verb_phrase(Number) -->
    verb(Number), noun_phrase(_Number2).

noun_phrase(Number) -->
    determiner(Number), noun(Number).

determiner --> [the].
noun(singular) --> [cat].
noun(plural) --> [cats].

?- sentence(A,B,C).

noun(singular) --> [mouse].
noun(plural) --> [mice].
verb(singular) --> [scares].
verb(plural) --> [scare].
Parse trees with DCGs

sentence(s(NP,VP)) --> noun_phrase(NP), verb_phrase(VP).
noun_phrase(np(D,N)) --> det(D), noun(N).
verb_phrase(vp(V,NP)) --> verb(V), noun_phrase(NP).
det(d(the)) --> [the].
det(d(a)) --> [a].
noun(n(bat)) --> [bat].
noun(n(cat)) --> [cat].
verb(v(eats)) --> [eats].

?- sentence(Parse_tree, [the,bat,eats,a,cat], []).
Parse_tree = s(np(d(the),n(bat)),vp(v(eats),np(d(a),n(cat))))
Parse tree and context sensitive

sentence(N, s(X,Y)) --> noun_phrase(N, X), verb_phrase(N, Y).
verb_phrase(N, vp(X,Y)) --> verb(N, X), noun_phrase(_, Y).
noun_phrase(N, np(X,Y)) --> determiner(N, X), noun(N, Y).
determiner(_, det(the)) --> [the].
noun(singular, noun(cat)) --> [cat].
noun(plural, noun(cats)) --> [cats].
noun(singular, noun(mouse)) --> [mouse].
verb(singular, verb(scares)) --> [scares].
verb(plural, verb(scare)) --> [scare].

?- sentence(A, B, C, D).
Complex parse tree DCG example:

\[
\begin{align*}
\text{s}(\text{NP}, \text{VP}) & \rightarrow \text{np}(\text{Num}, \text{NP}), \text{vp}(\text{Num}, \text{VP}). \\
\text{np}(\text{Num}, \text{np}(\text{PN})) & \rightarrow \text{pn}(\text{Num}, \text{PN}). \\
\text{np}(\text{Num}, \text{NP}) & \rightarrow \\
& \quad \text{d}(\text{Det}), \\
& \quad \text{n}(\text{Num}, \text{N}), \\
& \quad \text{rel}(\text{Num}, \text{Rel}), \\
& \quad \{\text{build_np}(\text{Det}, \text{N}, \text{Rel}, \text{NP})\}. /* \text{embedded Prolog goal} */ \\
\text{vp}(\text{Num}, \text{vp}(\text{TV}, \text{NP})) & \rightarrow \\
& \quad \text{tv}(\text{Num}, \text{TV}), \\
& \quad \text{np}(\_, \text{NP}). \\
\text{vp}(\text{Num}, \text{vp}(\text{IV})) & \rightarrow \text{iv}(\text{Num}, \text{IV}). \\
\text{rel}(\_, \text{rel}(\text{nil})) & \rightarrow [\]. \\
\text{rel}(\text{Num}, \text{rel}(\text{RP}, \text{VP})) & \rightarrow \\
& \quad \text{rpn}(\text{RP}), \text{vp}(\text{Num}, \text{VP}).
\end{align*}
\]

?- \text{s(}\text{Parse}\_\text{form}', 'The boy who sits reads the book', []).

\text{Parse}\_\text{form} = \text{s(}\text{np(}\text{d(the)}, \text{n(boy)}, \text{rel}(\text{rpn(who)}, \text{vp(}\text{iv(sits))})\)), \text{vp(}\text{tv(reads)}, \text{np(}\text{d(a)}, \text{n(\text{book}))})\))
Command Sequences For A Robot

- DCG grammars can also be used for recognizing or generating robot moves:
  - Example: up and down robot movements:
    - “up up down up down”
  - BNF grammar:
    - `<step> ::= up | down`
    - `<move> ::= <step> | <step> <move>`
  - Prolog DCG:
    - `step --> [up].`  
      - `- move( [up,down,up], []).` 
    - `step --> [down].`  
      - `yes` 
    - `move --> step.`  
      - `- move( [up, X, up], []).` 
    - `move --> step, move.`  
      - `X = up;` 
      - `X = down`
Command Sequences For A Robot

- Determining **meaning**:
  
  \[
  \text{move( } \text{Dist}\text{) } \rightarrow \text{ step( } \text{Dist}\text{).}
  \]
  
  \[
  \text{move( } \text{Dist}\text{) } \rightarrow \text{ step( } \text{D1}\text{), move( } \text{D2}\text{), \{Dist is D1 + D2\}.}
  \]
  
  \[
  \text{step( 1) } \rightarrow \text{ [ up].}
  \]
  
  \[
  \text{step( -1) } \rightarrow \text{ [ down].}
  \]

  
  \[
  \text{?- move( } \text{D, [ up, up, down, up], [ ] }.\text{)}
  \]
  
  \[
  \text{D = 2}
  \]
Prolog Direct Clause Grammars for parsing (using efficient tabling):

- Context sensitive,
- With number agreement,
- Using Wordnet KB.

```prolog
:- [wn_s].

sentence(N,s(X,Y)) --> noun_phrase(N,X), verb_phrase(N,Y).

noun_phrase(N,np(X,Y)) --> determiner(N,X), noun(N,Y).

verb_phrase(N,vp(X,Y)) --> verb(N,X), noun_phrase(_,Y).
verb_phrase(N,vp(X,Y)) --> verb(N,X), prepositional_phrase(_,Y).

noun(singular,noun(N)) --> [N], { s(_Synset,_,N,n,_,_,) }.

verb(singular,verb(V)) --> [V], { s(_Synset,_,V,v,_,_,) }.

determiner(singular, det(a)) --> [a].
determiner(_,det(the)) --> [the].
```

?- sentence(singular,Parse,[the, conference, is, a, success],[[]]).
Parse=s(np(det(the), noun(conference)), vp(verb(is), np(det(a), noun(success))))
Wordnet grammar

Adding general rules for plural cases:

noun(singular,noun(N))  -->  [N],
    { s(_Synset,_,N,n,_,_)}.

noun(plural,noun(N))  -->  [N],
    { s(_Synset,_,N2,n,_,_),
      atom_concat(N2, s, N) }.

verb(singular,verb(V))  -->  [V],
    { s(_Synset,_,V2,v,_,_),
      atom_concat(V2, s, V) }.

verb(plural,verb(V))  -->  [V],
    { s(_Synset,_,V,v,_,_) }.

?- sentence(singular, Parse, [the, team, wins, the, game],[]).

?- sentence(plural, Parse, [the, teams, win, the, games],[]).

Note: this does not include special rules for constructing plurals
E.g. plural of “entity” is “entities”.

(c) Paul Fodor (CS Stony Brook)
NLP meanings in Prolog

- Sentence → Parse tree → Formalised meaning

  “John paints” → paints( john)
  “John likes Annie” → likes( john, annie)

- DCG meaning:

  \%
  % “paints” means “paints(X)”
  %
  intrans_verb( X, paints(X) ) --> [ paints].
  % “john” means “john”
  %
  proper_noun( john ) --> [ john].
  %
  sentence(Y) --> proper_noun(X), intrans_verb( X, Y ).