CS 603: Programming Languages

Lecture 27

Spring 2004

Department of Computer Science
University of Alabama
Joel Jones

Overview

- Tracing append
- Introduction to Definite Clause Grammars

Tracing Append

- See Handout

Definite Clause Grammars

- Taken from:
  - `http://www.coli.uni-sb.de/~kris/prolog-course/html/node54.html`
- Prolog was originally intended for use in computational linguistics
- Definite Clause Grammars (DCGs) are syntactic sugar for implementing Context Free Grammars (CFGs) and other grammars

Introduction to CFGs

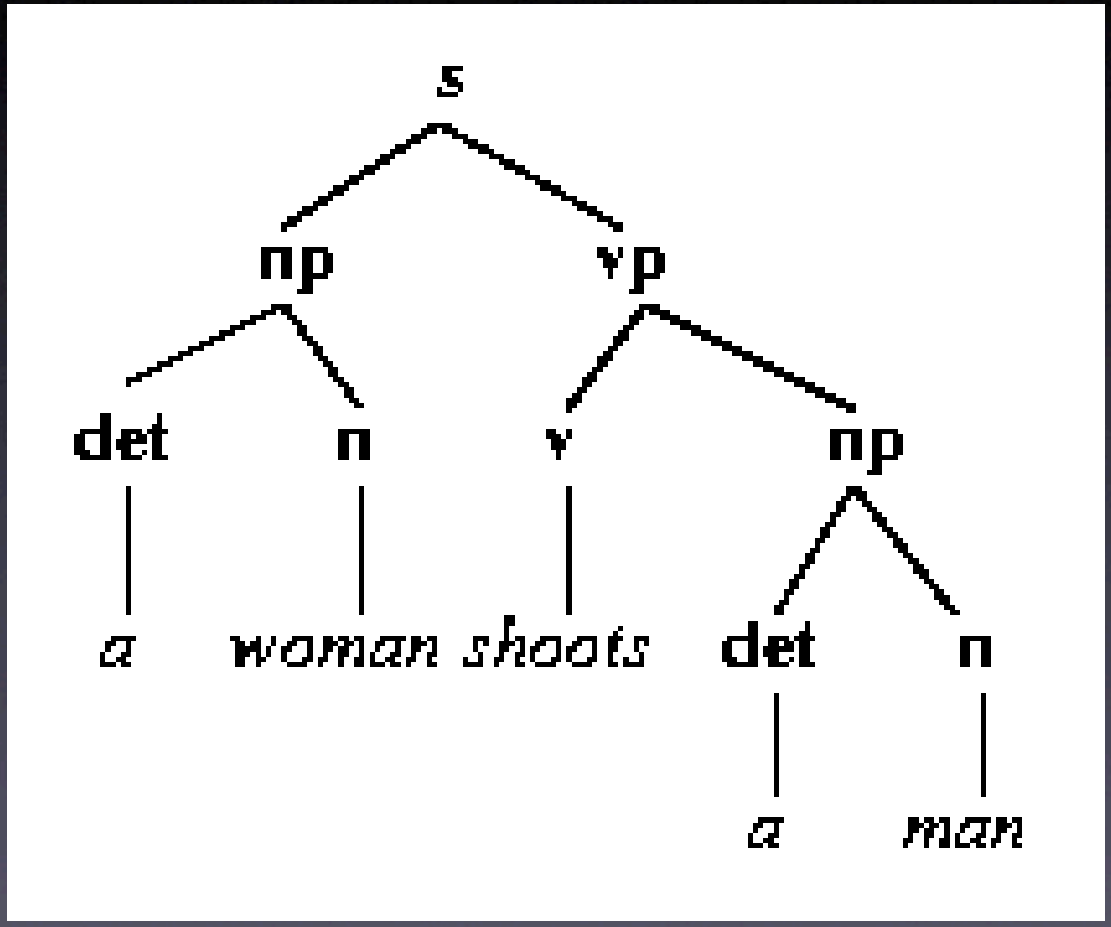
Nonterminals

CFGs have only a single non-terminal on the left-hand side (LHS)

```
s   -> np vp
np  -> det n
vp  -> v np
vp  -> v
det -> a
det -> the
n   -> woman
n   -> man
v   -> shoots
```

Terminals

Example Parse



CFG Recognition Using append

```
s(Z) :- np(X), vp(Y), append(X,Y,Z).
np(Z) :- det(X), n(Y), append(X,Y,Z).
vp(Z) :-  v(X), np(Y), append(X,Y,Z).
vp(Z) :-  v(Z).
det([the]).
det([a]).
n([woman]).
n([man]).
v([shoots]).
```

This is nice—see all sentences `s(X).`

But this is inefficient—trace `s([a,woman,shoots,a,man]).`

CFG Recognition Using Difference Lists

```
s(X,Z) :- np(X,Y), vp(Y,Z).
np(X,Z) :- det(X,Y), n(Y,Z).
vp(X,Z) :- v(X,Y), np(Y,Z).
vp(X,Z) :- v(X,Z).
det([the|W],W).
det([a|W],W).
n([woman|W],W).
n([man|W],W).
v([shoots|W],W).
```

The `s` rule says: *I know that the pair of lists* `X` *and* `Z` *represents a sentence if (1) I can consume* `X` *and leave behind a* `Y`*,and the pair* `X` *and* `Y` *represents a noun phrase, and (2) I can then go on to consume* `Y` *leaving Z behind ,and the pair* `Y` `Z` *represents a verb phrase* .

This is efficient—trace `s([a,woman,shoots,a,man],[]).`

Definite Clause Grammars

- DFGs allow us to write grammars without the ugliness of writing all those extra variables

```
s --> np,vp.
np --> det,n.
vp --> v,np.
vp --> v.
det --> [the].
det --> [a].
n --> [woman].
n --> [man].
v --> [shoots].
```

This is syntactic sugar for difference lists— `listing(s).`

Recursive Grammars

- Our original language only generated a finite number of sentences
- Add rules for conjunctions

```
s --> np,vp.
np --> det,n.
vp --> v,np.
vp --> v.
det --> [the].
det --> [a].
n --> [woman].
n --> [man].
v --> [shoots].
```

What happens if we add here?

```
s --> s,conj,s.
conj --> [and].
conj --> [or].
conj --> [but].
```

What happens if we add here and do `s([woman,shoot])`?

We need to eliminate left recursion

Left Recursion Eliminated

```
s --> simple_s.
s --> simple_s,conj,s.
simple_s --> np,vp.
np --> det,n.
vp --> v,np.
vp --> v.
det --> [the].
det --> [a].
n --> [woman].
n --> [man].
v --> [shoots].
conj --> [and].
conj --> [or].
conj --> [but].
```