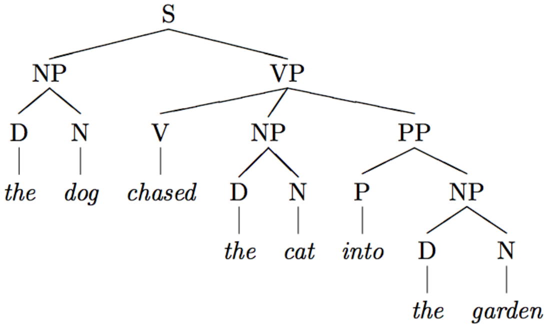
Definite-Clause Grammars [Covington, 1994]

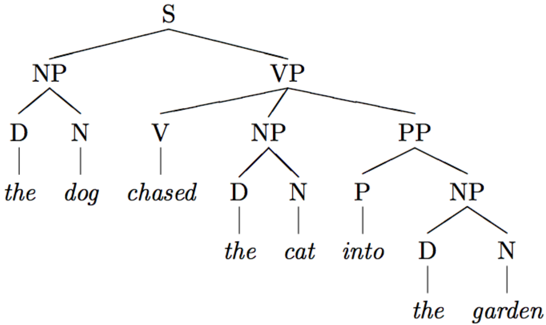
Desislava Zhekova

desi@linuxusers.de

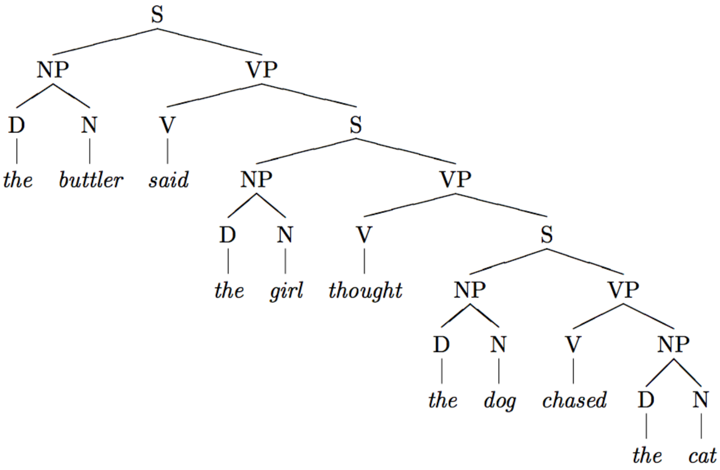
December 10, 2007



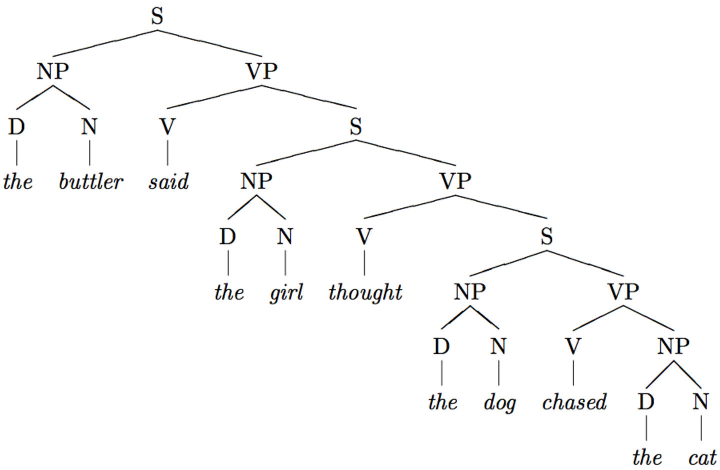
S	$\longrightarrow$	NP VP
NP	$\longrightarrow$	D N
VP	$\longrightarrow$	V NP
VP	$\longrightarrow$	V NP PP
PP	$\longrightarrow$	P NP
D	$\longrightarrow$	the
D	$\longrightarrow$	a
N	$\longrightarrow$	dog
N	$\longrightarrow$	cat
N	$\longrightarrow$	garden
V	$\longrightarrow$	chased
V	$\longrightarrow$	saw
P	$\longrightarrow$	into



S	→	NP VP
NP	→	D N
VP	→	V NP
VP	→	V NP PP
PP	→	P NP
D	→	the
D	→	a
N	→	dog
N	→	cat
N	→	garden
V	→	chased
V	→	saw
P	→	into


S
NP
VP
D
N
V
S
the
buttler
said
NP
VP
D
N
V
S
the
girl
thought
NP
VP
D
N
V
NP
the
dog
chased
D
N
the
cat

BOTTOM-UP vs. TOP-DOWN



S → NP VP

```
s(L1, L) :- np(L1, L2), vp(L2, L).
```

S → NP VP

```
s(L1, L) :- np(L1, L2), vp(L2, L).
```

`L1-[the, dog, saw, the, cat]` - *the input string (IS)*

`L2-[saw, the, cat]` - *the IS without the initial NP*

`L-[ ]` - *the IS without the NP or the VP*

```
s(L1,L) :- np(L1,L2), vp(L2,L).
np(L1,L) :- d(L1,L2), n(L2,L).
vp(L1,L) :- v(L1,L2), np(L2,L).
d([the|L],L).
d([a|L],L).
n([dog|L],L).
n([cat|L],L).
n([gardener|L],L).
n([policeman|L],L).
n([butler|L],L).
v([chased|L],L).
v([saw|L],L).
```

What are DCG Rules?

What are DCG Rules?

nonterminal symbol - -> *expansion*

What are DCG Rules?

nonterminal symbol --> *expansion*, **where expansion is:**

- A nonterminal symbol such as np
- A list of terminal symbols
- A null constituent represented by [ ]
- A plain Prolog goal enclosed in braces

```
{write ('Found NP')}
```

- A series of any of these expansions joined by commas

```
s  --> np, vp.
np --> d,  n.
vp --> v,  np.
d  --> [the]; [a].
n  --> [dog]; [cat]; [gardner]; [policeman]; [butler].
v  --> [chased]; [saw].
```

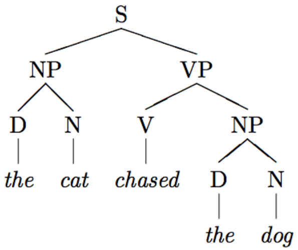
$$A \longrightarrow A\ B$$

$$\text{A} \rightarrow \text{A B}$$

$$\text{NP} \rightarrow \text{NP } \textit{Conj} \text{ NP}$$
$$\text{NP} \rightarrow \text{D N}$$

$$A \rightarrow A\ B$$

NP → NP *Conj* NP	NP → NPX *Conj* NP
NP → D N	NP → NPX
	NPX → D N



`s(np(d(the), n(cat)), vp(v(chased), np(d(the), n(dog))))` before translation

```
s(a, b) --> np(c, d), vp(e, f).
```

after translation

```
s(a, b, L1, L2) --> np(c, d, L1, L2), vp(e, f, L2, L).
```

```
s(s(NP, VP))  --> np(NP), vp(VP).
np(np(D, N))  --> d(D),  n(N).
vp(vp(V, NP)) --> v(V),  np(NP).
d(d(the))     --> [the].
n(n(dog))     --> [dog].
n(n(cat))     --> [cat].
v(v(chased))  --> [chased].
v(v(saw))     --> [saw].
```

The dog chases the cats.	(Singular subject, singular verb)
The dogs chase the cats.	(Plural subject, plural verb)
**The dog chase the cats.*	(Singular subject, plural verb)
**The dogs chases the cats.*	(Plural subject, singular verb)

```
n(singular) --> [dog]; [cat]; [mouse].
n(plural)   --> [dogs]; [cats]; [mice].
```

```
v(singular) --> [chases]; [sees].
v(plural)   --> [chase]; [see].
```

```
np(Number) --> d, n(Number).
vp(Number) --> v(Number), np(_).
```

```
s --> np(Number), vp(Number).
```

He sees him.	**Him sees he.*
She sees her.	**Her sees she.*
They see them.	**Them see they.*

```
pronoun(singular, nominative) --> [he]; [she].
pronoun(singular, accusative) --> [him]; [her].
pronoun(plural, nominative) --> [they].
pronoun(plural, accusative) --> [them].
```

```
np(Number, Case) --> pronoun(Number, Case).
np(Number, _)    --> d, n(Number).
```

```
s --> np(Number, nominative), vp(Number).
vp(Number) --> v(Number), np(_, accusative).
```

VERB	COMPLEMENT	EXAMPLE
sleep, bark	None	*(The cat) slept.*
chase, see	One NP	*(The dog) chased <u>the cat</u>.*
give, sell	Two NPs	*(Max) sold <u>Bill</u> <u>his car</u>.*
say, claim	Sentence	*(Max) claimed <u>the cat barked</u>.*

$$VP \rightarrow V$$
$$VP \rightarrow V\ NP$$
$$VP \rightarrow V\ NP\ NP$$
$$VP \rightarrow V\ S$$

$$
\begin{aligned}
&VP \rightarrow V1 \\
&VP \rightarrow V2\ NP \\
&VP \rightarrow V3\ NP\ NP \\
&VP \rightarrow V4\ S
\end{aligned}
$$

```
vp --> v(1).
vp --> v(2), np.
vp --> v(3), np, np.
vp --> v(4), s.
```

```
v(1) --> [barked]; [slept].
v(2) --> [chased]; [saw].
v(3) --> [gave]; [sold].
v(4) --> [said]; [thought].
```

Max said Bill thought Joe believed Fido barked.

Who said Bill thought Joe believed Fido barked? (Max.)
Who did Max say ␣ thought Joe believed Fido barked? (Bill.)
Who did Max say Bill thought ␣ believed Fido barked? (Joe.)
Who did Max say Bill thought Joe believed ␣ barked? (Fido.)

```
s(In,Out)-->[who,did],np([who|In]),Out1),vp(Out1,Out).
```

```
np([who|Out], Out) --> [ ].
```

```
s(In, Out) --> np(In, Out1), vp(Out1, Out).
```

```
np(X, X) --> [max]; [joe]; [bill]; [frido].
```

```
vp(X, X)     --> v.
vp(In, Out) --> v, np(In, Out).
vp(In, Out) --> v, s(In, Out).
```

```
v --> [saw]; [said]; [thought]; [believed]; [barked].
v --> [see]; [say]; [think]; [believe]; [bark].
```

```
n --> [X], (noun (X)).

noun(dog).
noun(cat).
noun(gardener).
.
.
.
```

Covington, M. A. (1994).
Natural Language Processing for Prolog Programmers.
Prentice Hall, Englewood Cliffs.