Computational Semantics:

A Syntax-Semantics Interface in Prolog

Lexicon sample:

- (PN ; $\lambda P.(P)(tom)$) $\rightarrow$ *tom*
- (N ; $\lambda y.cat(y)$) $\rightarrow$ *cat*
- (DT ; $\lambda P.\lambda Q.\forall x((P)(x) \Rightarrow (Q)(x))$) $\rightarrow$ *every* $|$ *each*
- (DT ; $\lambda P.\lambda Q.\exists x((P)(x) \wedge (Q)(x))$) $\rightarrow$ *a* $|$ *an* $|$ *some*
- (IV ; $\lambda x.sneeze(x)$) $\rightarrow$ *sneezed*
- (TV ; $\lambda R.\lambda x.(R)(\lambda y.like(x, y))$) $\rightarrow$ *likes*
- (P ; $\lambda P.\lambda Q.\lambda x.((Q)(x) \wedge (P)(\lambda y.in(x, y)))$) $\rightarrow$ *in*
- (ADJ ; $\lambda P.\lambda x.((P)(x) \wedge yellow(x))$) $\rightarrow$ *yellow*

Grammar rules, so far:

- (NP; ϕ) $\rightarrow$ (PN; ϕ)
- (VP; ϕ) $\rightarrow$ (IV; ϕ)
- (NP; $(\phi)(\psi)$) $\rightarrow$ (DT; ϕ) (N; ψ)
- (N ; $(\phi)(\psi)$) $\rightarrow$ (ADJ ; ϕ) (N ; ψ)
- (PP; $(\phi)(\psi)$) $\rightarrow$ (P; ϕ) (NP; ψ)
- (N; $(\psi)(\phi)$) $\rightarrow$ (N; ϕ) (PP; ψ)
- (VP; $(\phi)(\psi)$) $\rightarrow$ (TV; ϕ) (NP; ψ)
- (S; $(\phi)(\psi)$) $\rightarrow$ (NP; ϕ) (VP; ψ)

Compositionality

Lexical specifications

(N ; $\lambda y.cat(y)$) $\rightarrow$ *cat*

(IV ; $\lambda x.sneeze(x)$) $\rightarrow$ *sneezed*

(DT ; $\lambda P.\lambda Q.\forall x((P)(x) \Rightarrow (Q)(x))$) $\rightarrow$ *every*

(DT ; $\lambda P.\lambda Q.\exists x((P)(x) \wedge (Q)(x))$) $\rightarrow$ *some*

We'll take a couple of shortcuts, and avoid implementing $\beta-$reduction in full.

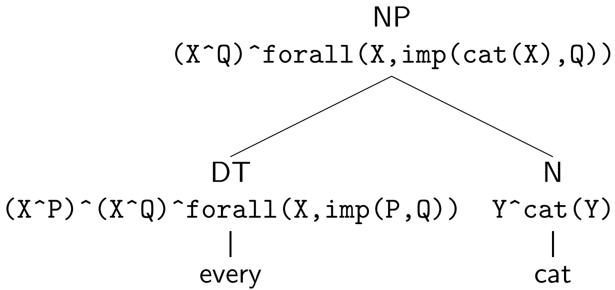
Implemented:

```
lex(n(X^cat(X)),cat).
lex(iv(X^sneeze(X)),sneezed).
lex(dt((X^P)^(X^Q)^forall(X,imp(P,Q))),every).
lex(dt((X^P)^(X^Q)^exists(X,and(P,Q))),a).
```

Phrasal rule for NPs

(NP; $(\phi)(\psi)$) $\rightarrow$ (DT; ϕ) (N; ψ)

```
rule(np(B),[dt(A^B),n(A)]).
```



`A = X^P = Y^cat(Y)`

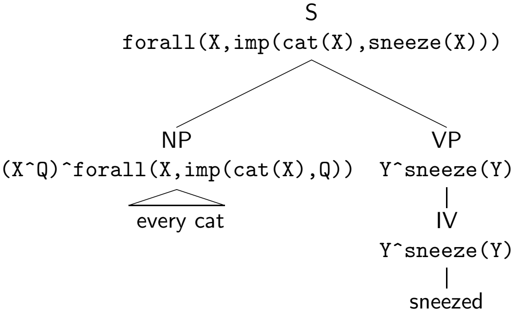
so:

`X = Y` and `P = cat(X)`

Compositionality

$$(\text{IV}\ ;\ \lambda x.sneeze(x)) \rightarrow \textit{sneezed}$$
$$(\text{VP};\ \phi) \rightarrow (\text{IV};\ \phi)$$
$$(\text{S};\ (\phi)(\psi)) \rightarrow (\text{NP};\ \phi)\ (\text{VP};\ \psi)$$

```
lex(iv(X^sneeze(X)),sneezed).
rule(vp(A),[iv(A)]).
rule(s(B),[np(A^B),vp(A)]).
```

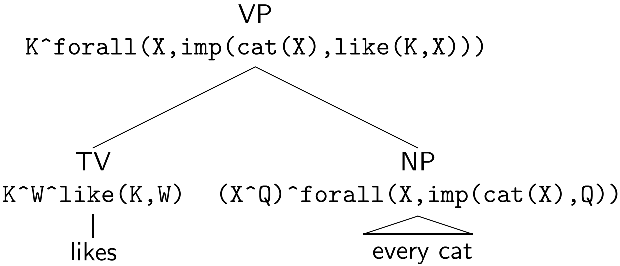


`A = X^Q = Y^sneeze(Y)`, and so `X = Y` and `Q = sneeze(X)`

Compositionality

$(\text{TV} ; \lambda R.\lambda w.(R)(\lambda k.like(w,k))) \rightarrow$ *likes*
$(\text{TV} ; \lambda k.\lambda w.like(k,w)) \rightarrow$ *likes*
$(\text{VP}; (\phi)(\psi)) \rightarrow (\text{TV}; \phi)\ (\text{NP}; \psi)$

```
lex(tv(K^W^like(K,W)),likes).
rule(vp(A^B),[tv(A^C),np(C^B)]).
```

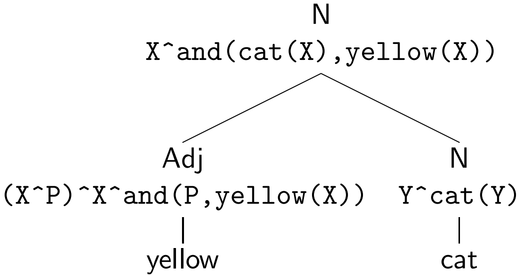



```
A = K  and  C = W^like(K,W) = X^Q
```
so:
```
B = forall(X, imp(cat(X), like(K,X)))
```

Compositionality

$$(\text{ADJ} \;;\; \lambda P.\lambda x.((P)(x) \wedge \mathit{yellow}(x))) \rightarrow \textit{yellow}$$
$$(\text{N} \;;\; (\phi)(\psi)) \rightarrow (\text{ADJ} \;;\; \phi)\ (\text{N} \;;\; \psi)$$

```
lex(adj((X^P)^X^and(P,yellow(X))),yellow).
rule(n(A),[adj(B^A),n(B)]).
```



`B = Y^cat(Y) = X^P`

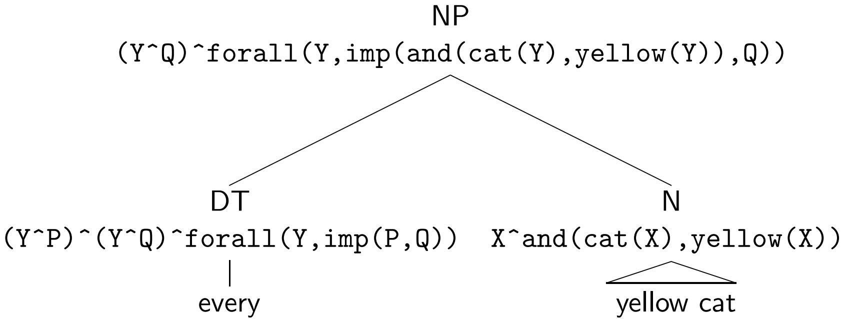
so:

`X = Y  and  cat(X) = P`

Compositionality

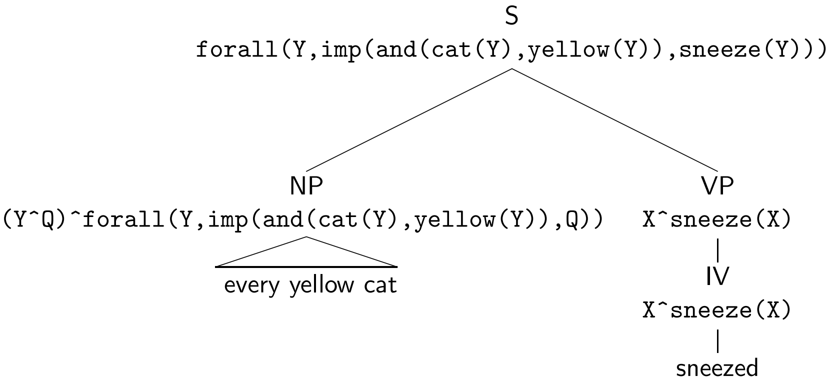
Everything works as before:

```
rule(np(B),[dt(A^B),n(A)]).
```



Finally:

```
lex(iv(X^sneeze(X)),sneezed).
rule(vp(A),[iv(A)]).
rule(s(B),[np(A^B),vp(A)]).
```



Proper names

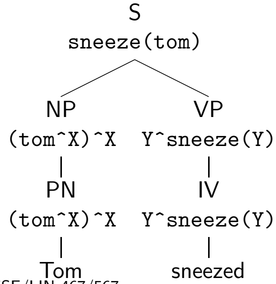
We'll avoid the complexity of PN representations:

(PN ; $\lambda P.(P)(tom)$) $\rightarrow$ *tom*

(NP; ϕ) $\rightarrow$ (PN; ϕ)

... by implementing it as follows (note: this is no longer a λ-term!):

```
lex(pn((tom^X)^X,tom).
rule(np(A),[pn(A)]).
rule(s(B),[np(A^B),vp(A)]).
```



`A = (tom^X) = Y^sneeze(Y)`

Compositionality

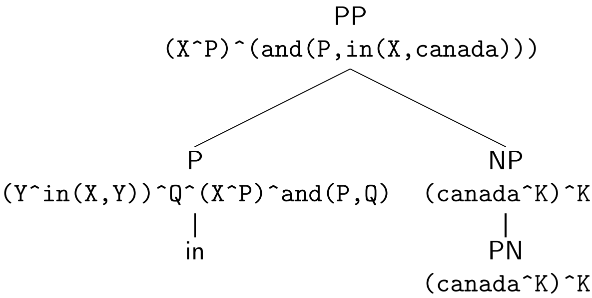
The semantic composition in adnominal PPs:

(PP; $(\phi)(\psi)$) $\rightarrow$ (P; ϕ) (NP; ψ)

(P ; $\lambda P.\lambda Q.\lambda x.((Q)(x) \wedge (P)(\lambda y.in(x, y)))$)

Since we are avoiding β-reduction, we get:

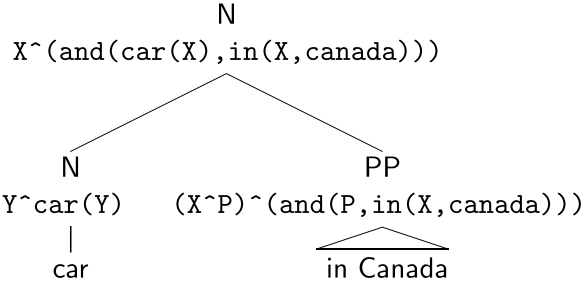
```
lex(p((Y^in(X,Y))^Q^(X^P)^and(P,Q)),in).
rule(pp(C),[p(A^B^C),np(A^B)]).
```



PPs continued

(N; $(\psi)(\phi)$) $\rightarrow$ (N; ϕ) (PP; ψ)

```
rule(n(A^C),[n(A^B),pp((A^B)^C)]).
```



The syntax-semantics interface, implemented:

```
lex(n(X^cat(X)),cat).
lex(pn((tom^X)^X,tom).
lex(dt((X^P)^(X^Q)^forall(X,imp(P,Q))),every).
lex(dt((X^P)^(X^Q)^exists(X,and(P,Q))),a).
lex(iv(X^sneeze(X)),sneezed).
lex(tv(K^W^like(K,W)),likes).
lex(adj((X^P)^X^and(P,yellow(X))),yellow).
lex(p((Y^in(X,Y))^Q^(X^P)^and(P,Q)),in).

rule(np(A),[pn(A)]).
rule(np(B),[dt(A^B),n(A)]).
rule(n(A^C),[n(A^B),pp((A^B)^C)]).
rule(n(A),[adj(B^A),n(B)]).
rule(pp(C),[p(A^B^C),np(A^B)]).
rule(vp(A),[iv(A)]).
rule(vp(A^B),[tv(A^C),np(C^B)]).
rule(s(B),[np(A^B),vp(A)]).
```