TAG Parsing the Earley Way

Eric Kow
LORIA

11 January 2004

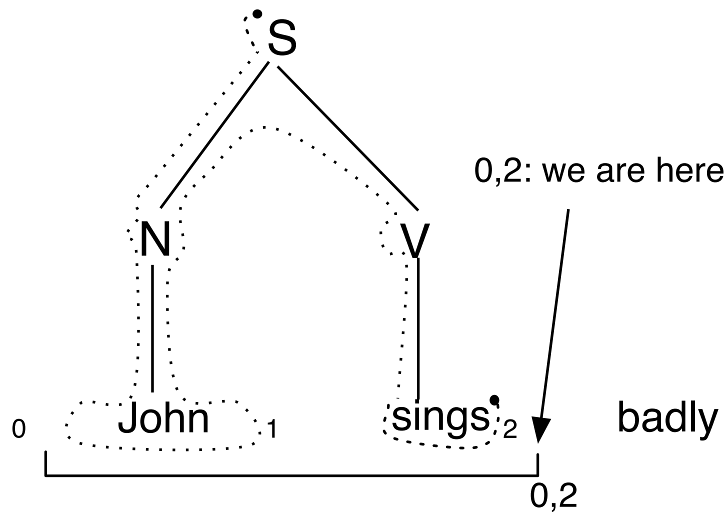
Introduction

The Earley algorithm is an (almost) descendant chart-parser for CFG.

1. We present a variant proposed by Schabes (1988) for Tree Adjoining Grammar (TAG).
2. We assume a basic familiarity with TAG, but no knowledge of Earley or chart-parsing.
3. We do not explain chart-parsing in general, but this is not crucial.

Basic Concept: Counters

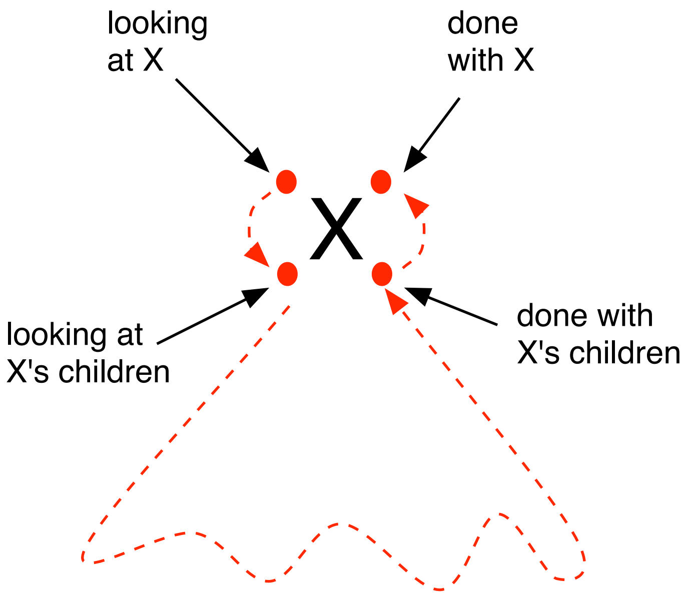
We use a pair of numbers to track what part of the input string we are looking at.

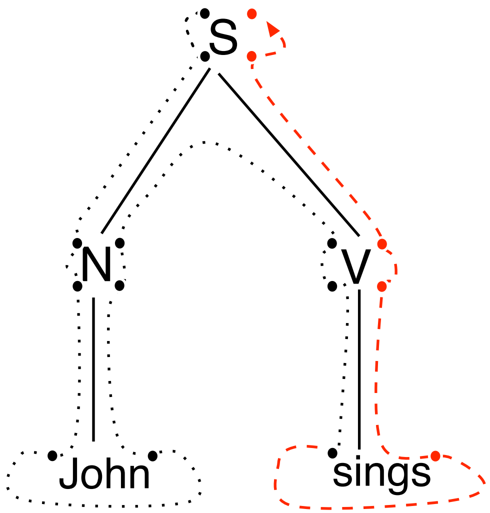


We initialise the counters to $0, 0$

Basic Concept: Dots

We use some dots to remember what we have seen in the parse.



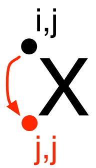


We always *copy* dots around; we never destroy them.

Simple Rules

We use some simple rules to move (copy) these dots around. Notice carefully how the counters advance in all of these rules!

Predict

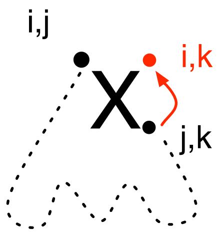


pred $\dfrac{<{}^{\bullet}X,i,j>}{<{}_{\bullet}X,j,j>}$

Scan

scan $\dfrac{<{}^{\bullet}word,i,j>}{<word^{\bullet},i,j+1>}$

Complete

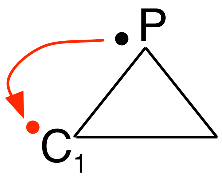


comp $\dfrac{<{}^{\bullet}X,i,j><X_{\bullet},j,k>}{<X^{\bullet},i,k>}$

Equivalences

Dots will also be copied automatically from

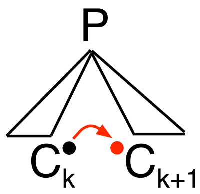
Parent $\longrightarrow$ First Child



$(= a)$ $<_{\bullet} P, i, j> \Leftrightarrow <^{\bullet} C_1, i, j>$

C_1 is the first child of P

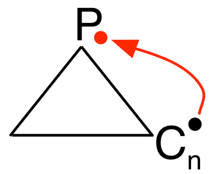
Sibling$\longrightarrow$ Sibling



$(= b)$ $< C_k^{\bullet}, i, j> \Leftrightarrow <^{\bullet} C_{k+1}, i, j>$

C_k is a child of P and C_{k+1} is the next

Last Child $\longrightarrow$ Parent



$(= c)$ $< C_n^{\bullet}, i, j> \Leftrightarrow < P_{\bullet}, i, j>$

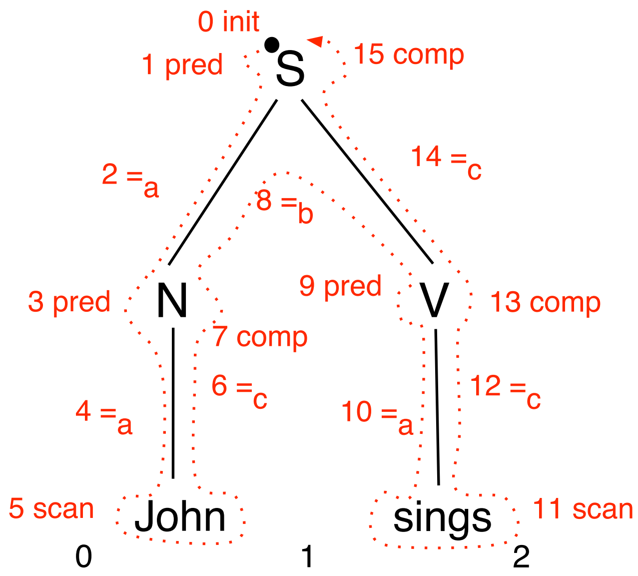
C_n is the last child of P

We call these movements **equivalences** and give them names like $= a$.

Exercise: Figure out why these do not need to be rules

Simple Parsing

Let us parse a sentence using a silly one-tree grammar.



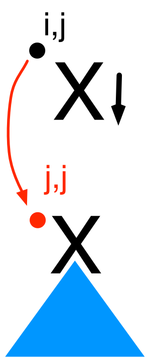
0	init	${}^{\bullet}S$	$0, 0$
1	pred	${}_{\bullet}S$	
2	$=_a$	${}^{\bullet}N$	
3	pred	${}_{\bullet}N$	
4	$=_a$	${}^{\bullet}John$	
5	scan	$John^{\bullet}$	$0, 1$
6	$=_c$	$N_{\bullet}$	
7	comp	$N^{\bullet}$	
8	$=_b$	${}^{\bullet}V$	
9	pred	${}_{\bullet}V$	$1, 1$
10	$=_a$	${}^{\bullet}sings$	
11	scan	$sings^{\bullet}$	$1, 2$
12	$=_c$	$V_{\bullet}$	
13	comp	$V^{\bullet}$	$0, 2$
14	$=_c$	$S_{\bullet}$	
15	comp	$S^{\bullet}$	

Exercise: Follow this trace and draw the dots on the parse tree.

Substitution Rules

We add rules to do more interesting things, such as **substitution**. Notice how similar these are to the simple versions of `predict` and `complete`.

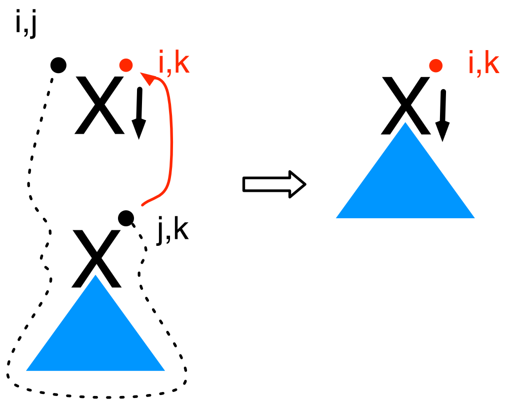
Predict substitution



pred$_S$ $\dfrac{<{}^{\bullet}X_{\downarrow},i,j>}{<{}^{\bullet}X,j,j>}$

X is the root of an elementary tree

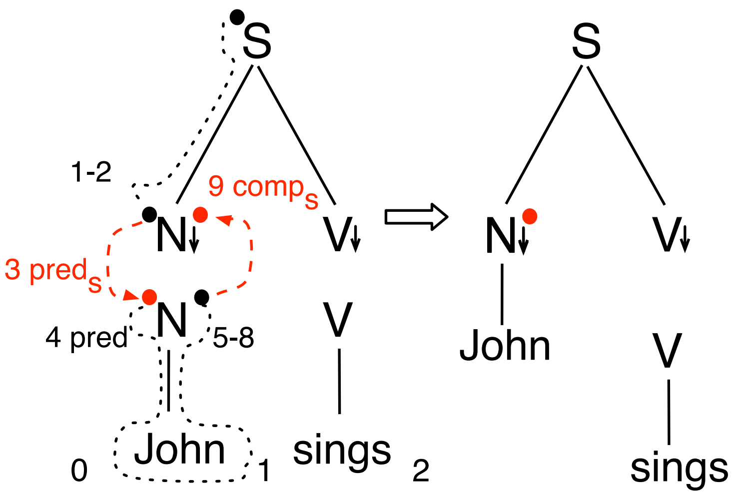
Complete substitution



comp$_S$ $\dfrac{<{}^{\bullet}X_{\downarrow},i,j><X^{\bullet},j,k>}{<X_{\downarrow}^{\bullet},i,k>}$

Substitution Example

Let us parse the sentence with a more interesting grammar.

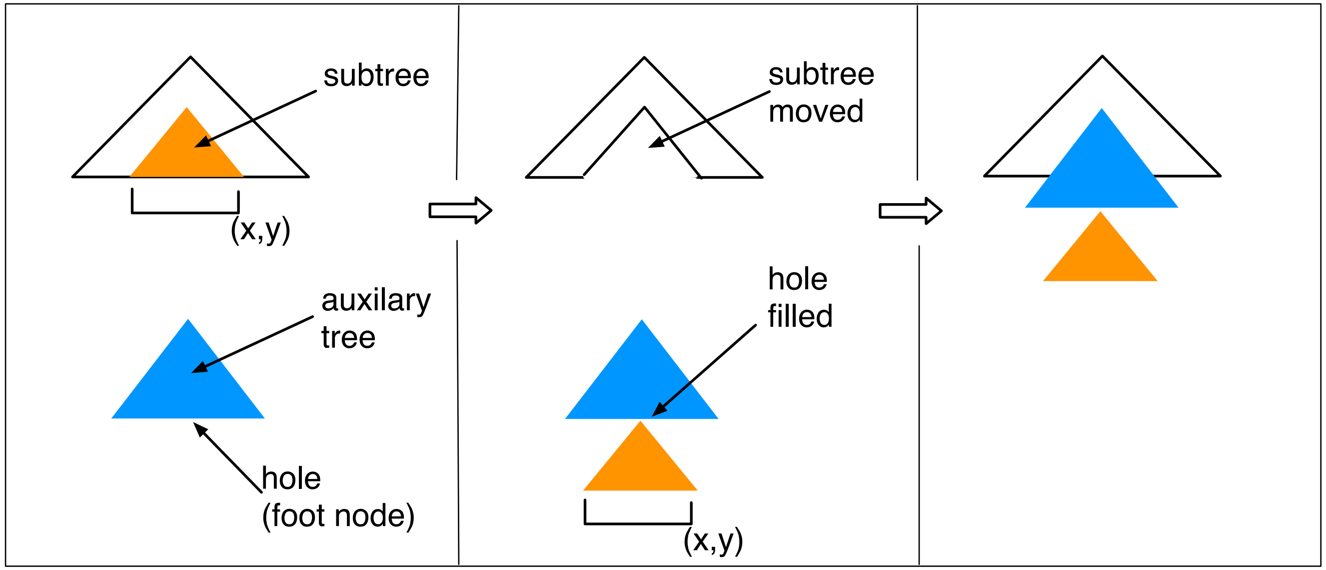


0	init	$^\bullet S$	$0, 0$
1	pred	$_\bullet S$	
2	$=a$	$^\bullet N \downarrow$	
3	pred$_S$	$^\bullet N$	
4	pred	$_\bullet N$	
5	$=a$	$^\bullet John$	
6	scan	$John^\bullet$	$0, 1$
7	$=c$	$N_\bullet$	
8	comp	$N^\bullet$	
9	comp$_S$	$N \downarrow^\bullet$	

Exercise: Complete this parse.

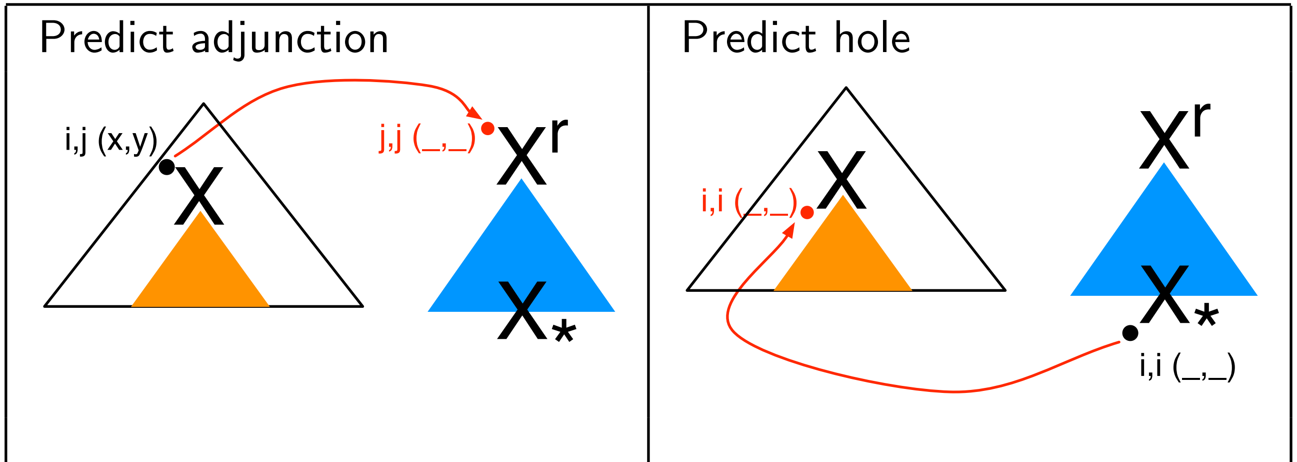
Advanced Concept: Holes

In order to do TAG **adjunction**, we need to account for the behaviour of **holes** (foot nodes) by introducing a second pair of counters. We write these in parentheses, for example $(3, 4)$



Adjunction Rules (Predict)

Now we need to add four more rules to make **adjunction** work. The `predict` rules create potential holes $(\_,\_)$.



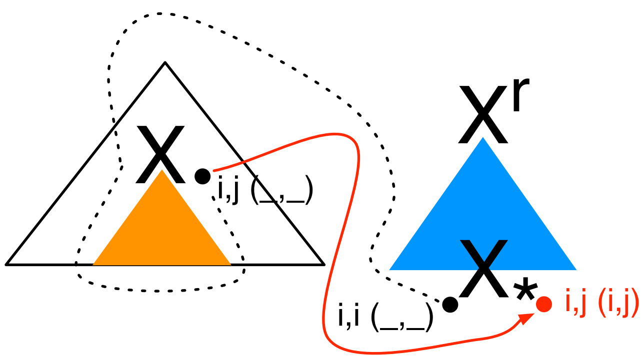
$\text{pred}_a \quad \dfrac{<{}^{\bullet}X,i,j,(x,y)>}{<{}^{\bullet}X^r,j,j,(\_,\_)>}$

$\text{pred}_h \quad \dfrac{<\bullet X_*,i,i,(\_,\_)>}{<\bullet X,i,i,(\_,\_)>}$

Adjunction Rules (Complete)

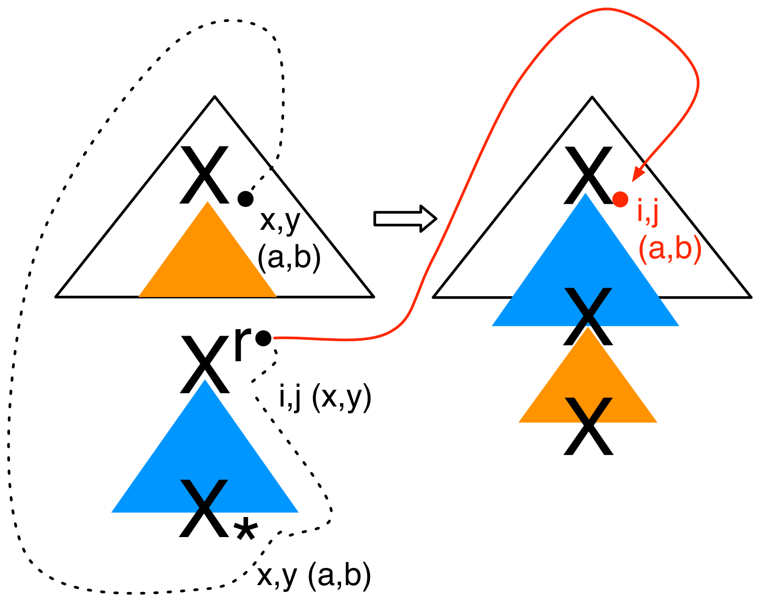
The `complete` rules fill the hole with a piece of the original tree.

Complete hole



(comp$_h$) $$\frac{<\bullet X_*,i,i,(\_,\_)><X_\bullet,i,j,(\_,\_)>}{<X_{*\bullet},i,j,(i,j)>}$$

Complete adjunction



(comp$_a$) $$\frac{<X_\bullet,x,y,(a,b)><X^{r\bullet},i,j,(x,y)>}{<X_\bullet,i,j,(a,b)>}$$

Holes (Equivalences, Predict and Scan)

We also need to modify all the other rules we have seen to account for holes.

1. `Predict` and $\texttt{predict}_s$ create non-holes $(\_, \_)$.
   Note: We can't tell the difference between non-holes and potential holes.

2. Equivalences and `scan` keep (propagate) the same holes as before.

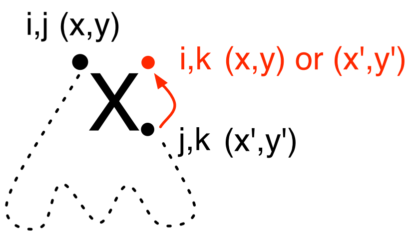
3. `Complete` and $\texttt{complete}_s$ also propagate holes, but they are tricky!

*Exercise: Draw the diagrams for the equivalences, **`scan`** and the **`predict`** rules with holes.*

Holes (Complete)

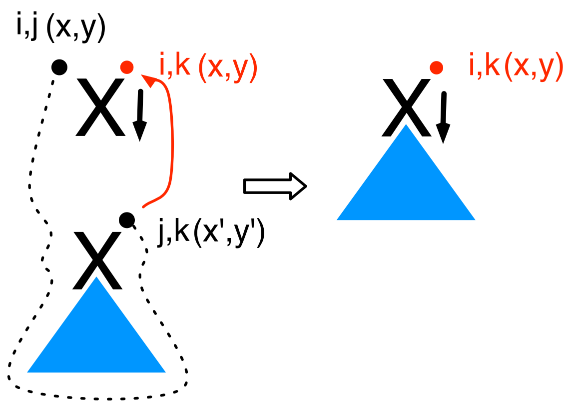
At most one hole ever exists during any `complete`, but this hole could come from any subtree! If there is a hole, we propagate it.

Complete



comp $\dfrac{<{}^{\bullet}X,i,j,(x,y)><X_{\bullet},j,k,(x',y')>}{<X^{\bullet},i,k,(x,y)\oplus(x',y')>}$

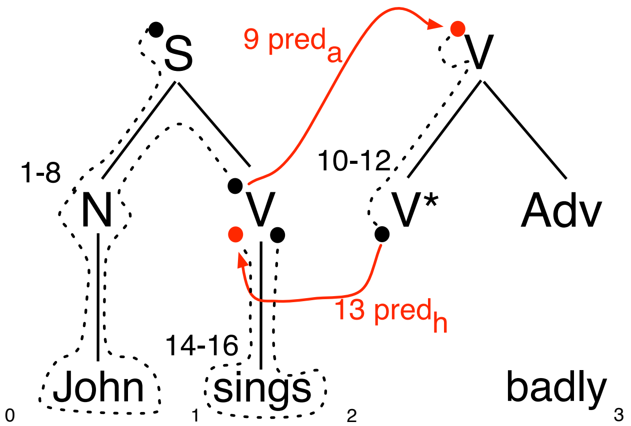
Complete substitution



comp$_s$ $\dfrac{<{}^{\bullet}X_{\downarrow},i,j,(x,y)><X^{\bullet},j,k,(x',y')>}{<X^{\bullet}_{\downarrow},i,k,(x,y)>}$

Adjunction Example

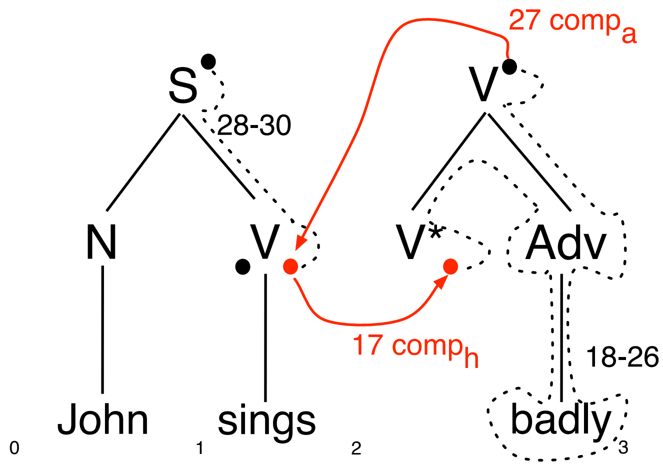
Let's try a very simple adjunction. **Before** parsing "sings", we `predict` adjunction on the V node.



0	init	${}^{\bullet}S$	$0, 0$
1	pred	${}_{\bullet}S$	$0, 0\ (\_, \_)$
2	$=a$	${}^{\bullet}N$	
3	pred	${}_{\bullet}N$	$0, 0\ (\_, \_)$
4	$=a$	${}^{\bullet}John$	
5	scan	$John^{\bullet}$	$0, 1\ (\_, \_)$
6	$=c$	$N_{\bullet}$	
7	comp	$N^{\bullet}$	$0, 1\ (\_, \_)$
8	$=b$	${}^{\bullet}V$	
9	pred$_a$	${}^{\bullet}V$	$1, 1\ (\_, \_)$
10	pred	${}_{\bullet}V$	$1, 1\ (\_, \_)$
11	$=a$	${}^{\bullet}V^*$	
12	pred	${}_{\bullet}V^*$	$1, 1\ (\_, \_)$
13	pred$_h$	${}_{\bullet}V$	$1, 1\ (\_, \_)$
14	$=a$	${}^{\bullet}sings$	$1, 1\ (\_, \_)$
15	scan	$sings^{\bullet}$	$1, 2\ (\_, \_)$
16	$=c$	$V_{\bullet}$	

Adjunction Example (2)

After parsing "sings" and "badly", we `complete` the adjunction.

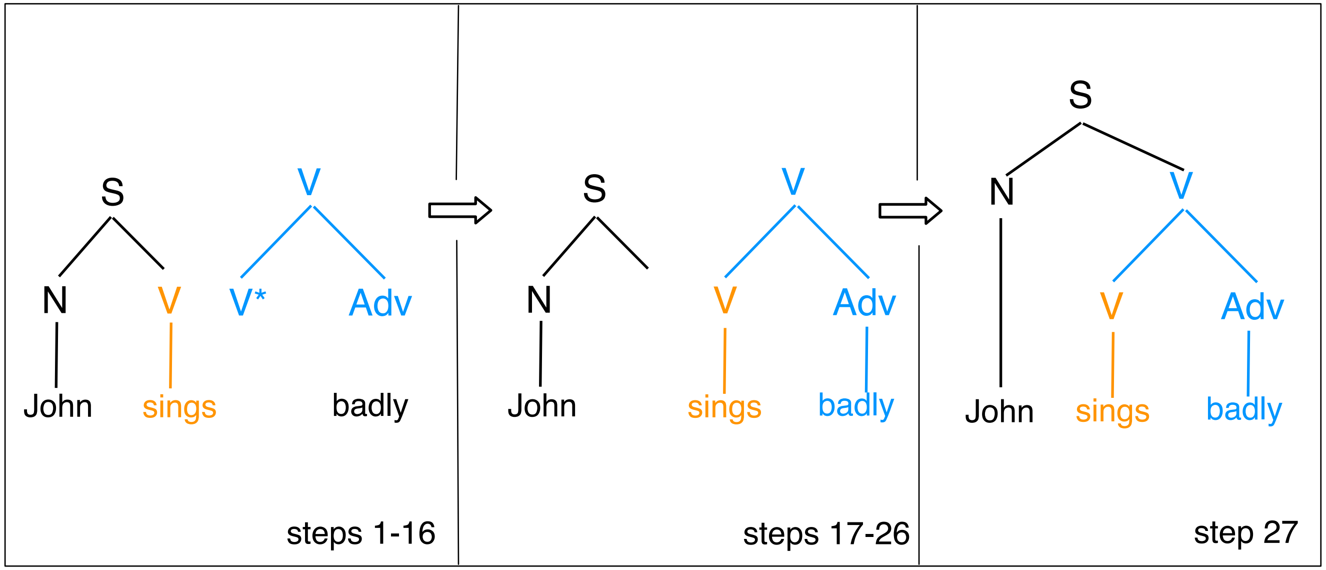


9	pred$_a$	$^\bullet V$	$1, 1\ (\_, \_)$
11	$=_a$	$^\bullet V^*$	$1, 1\ (\_, \_)$
13	pred$_h$	$_\bullet V$	$1, 1\ (\_, \_)$
16	$=_c$	$V_\bullet$	1, 2
17	comp$_h$	$V^*_\bullet$	$1, 2\ (1, 2)$
18	comp	$V^{*\bullet}$	$1, 2\ (1, 2)$
19	$=_b$	$^\bullet Adv$	
20	pred	$_\bullet Adv$	$2, 2\ (1, 2)$
21	$=_a$	$^\bullet badly$	
22	scan	$badly^\bullet$	$2, 3\ (1, 2)$
23	$=_c$	$Adv_\bullet$	
24	comp	$Adv^\bullet$	$1, 3\ (1, 2)$
25	$=_c$	$V_\bullet$	
26	comp	$V^\bullet$	
27	comp$_a$	$V_\bullet$	$1, 3\ (\_, \_)$
28	comp	$V^\bullet$	$1, 3\ (\_, \_)$
29	$=_c$	$S_\bullet$	
30	comp	$S^\bullet$	$1, 3\ (\_, \_)$

Exercise: Which steps were used use to derive steps 17 and 27?

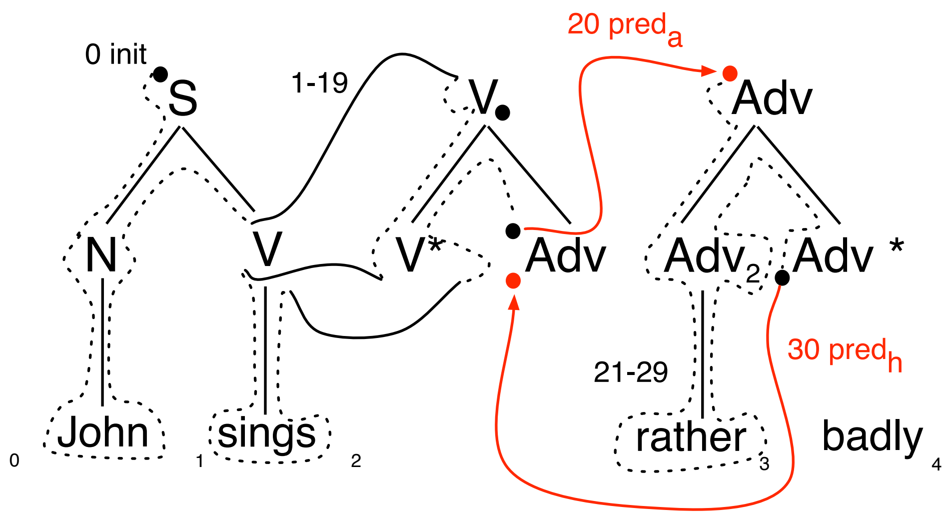
Adjunction Example (Summary)

The end result of our simple adjunction



2nd Adjunction Example

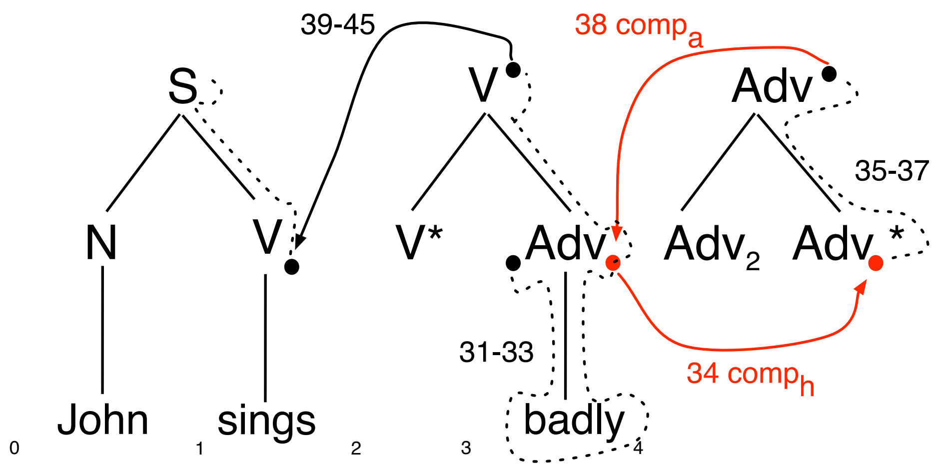
Now let's try adjunction between two auxiliary trees. We first `predict` adjunction on V and then on Adv. (See previous example for the steps skipped)



19	$=_b$	${}^{\bullet}Adv$	$1,2\ (1,2)$
20	pred$_a$	${}^{\bullet}Adv$	$2,2\ (\_,\_)$
21	pred	${}_{\bullet}Adv$	$2,2\ (\_,\_)$
22	$=_a$	${}^{\bullet}Adv_2$	
23	pred	${}_{\bullet}Adv_2$	$2,2\ (\_,\_)$
24	$=_a$	${}^{\bullet}rather$	
25	scan	$rather^{\bullet}$	$2,3\ (\_,\_)$
26	$=_c$	$Adv_{2\bullet}$	
27	comp	$Adv_2^{\bullet}$	$2,3\ (\_,\_)$
28	$=_b$	${}^{\bullet}Adv^*$	
29	pred	${}_{\bullet}Adv^*$	$3,3\ (\_,\_)$
30	pred$_h$	${}_{\bullet}Adv$	$3,3\ (\_,\_)$

2nd Adjunction Example (2)

Next we `complete` adjunction on the Adv and then on V.

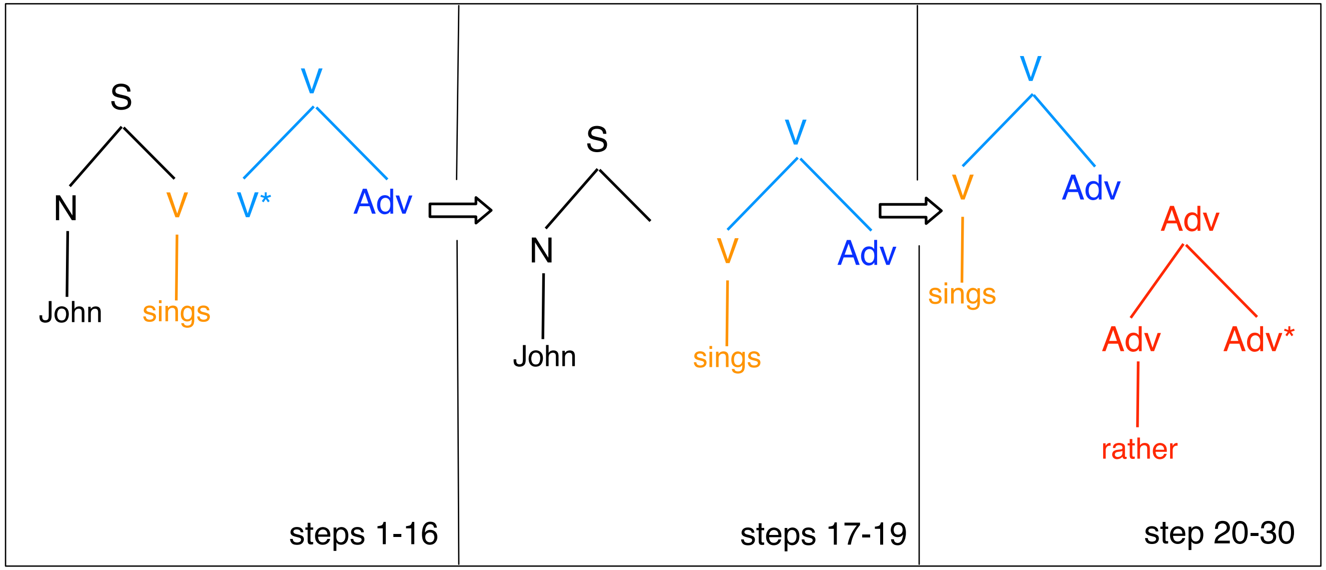


9	pred$_a$	$^\bullet V$	$1,1\ (\_,\_)$
16	$=c$	$V\bullet$	$1,2\ (\_,\_)$
20	pred$_a$	$^\bullet Adv$	$2,2\ (\_,\_)$
22	$=a$	$^\bullet Adv_2$	$2,2\ (\_,\_)$
30	pred$_h$	$\bullet Adv$	$3,3\ (\_,\_)$
31	$=a$	$^\bullet badly$	$3,3\ (\_,\_)$
32	scan	$badly^\bullet$	$3,4\ (\_,\_)$
33	$=c$	$Adv\bullet$	$3,4\ \mathbf{(\_,\_)}$
34	comp$_h$	$Adv^*_\bullet$	$3,4\ (3,4)$
35	comp	$Adv^{*\bullet}$	$3,4\ (3,4)$
36	$=c$	$Adv\bullet$	
37	comp	$Adv^\bullet$	$3,4\ (3,4)$
38	comp$_a$	$Adv\bullet$	$3,4\ \mathbf{(\_,\_)}$
39	comp	$Adv^\bullet$	$2,4\ (\_,\_)$
40	$=c$	$V\bullet$	

Exercise: Complete this parse.

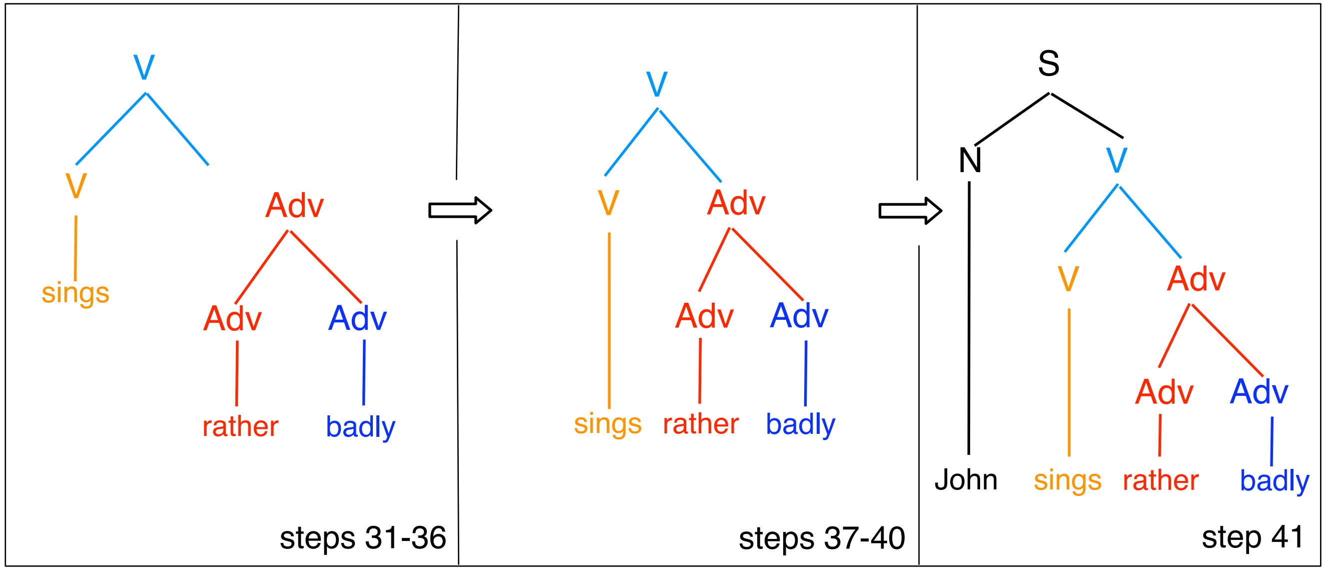
2nd Adjunction Example (Summary 1)

First we predict adjunction on V and then on Adv.



2nd Adjunction Example (Summary 2)

And finally, we complete adjunction on Adv and then V.



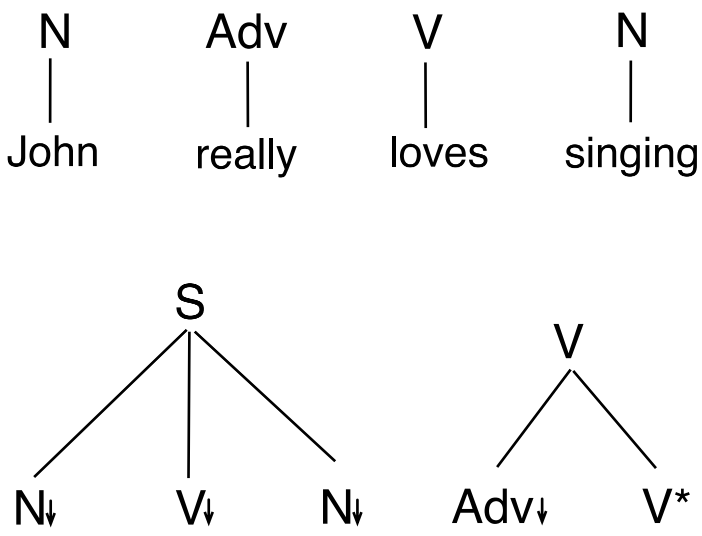
Parsing order

It might be helpful in general to think of an Earley parse as a context free grammar:

1. P $\rightarrow$ Simple $|$ Subst $|$ Adj
2. Simple $\rightarrow$ $\texttt{scan}$ $|$ $\texttt{pred} =_a$ P' $=_c$ $\texttt{comp}$
3. P' $\rightarrow$ P $|$ P' $=_b$ P'
4. Subst $\rightarrow$ $\texttt{pred}_s$ P $\texttt{comp}_s$
5. Adj $\rightarrow$ $\texttt{pred}_a$ P $\texttt{pred}_h$ P $\texttt{comp}_h$ P $\texttt{comp}_a$

Putting it all together

Exercise: Try parsing "John really loves singing" using the given grammar with adjunction and substitution.



Conclusion

Schabes' Earley-style algorithm provides a way to chart-parse TAG from the top down.

1. Consists of 3 equivalences, 3 basic Earley rules, 4 rules for adjunction and 2 rules for substitution.

2. Cost: $O(n^6)$ with adjunction

3. Parsing without adjunction $\rightarrow$ CFG

4. Substitution rules are optional but useful in practice.