Marlo Manuel Carreon
02/20/2018
Linguistics 185A
Assignment 6

1.

$$
\begin{aligned}
S &\rightarrow NP\ VP \\
S &\rightarrow WHILE\ S\ S \\
NP &\rightarrow NP\ POSS\ N \\
NP &\rightarrow (D)\ N\ (PP)\ (SRC)\ (ORC) \\
VP &\rightarrow V\ (NP)\ (PP) \\
PP &\rightarrow P\ NP \\
SRC &\rightarrow THAT\ VP \\
ORC &\rightarrow NP\ V
\end{aligned}
\qquad
\begin{aligned}
N &\rightarrow \text{dog, cat, rat, wife, brother} \\
NP &\rightarrow \text{John, Mary} \\
V &\rightarrow \text{barked, chased, bit, ate, fled} \\
D &\rightarrow \text{the} \\
P &\rightarrow \text{on, in, with} \\
THAT &\rightarrow \text{that} \\
POSS &\rightarrow \text{'s} \\
WHILE &\rightarrow \text{while}
\end{aligned}
$$

	left-branching	right-branching	center-embedding
Human difficulty	no increase	no increase	increase
Bottom-up parser	no increase	increase	increase
Top-down parser	increase	no increase	**increase**
Left-corner parser	**no increase**	**no increase**	**increase**

Left-corner parser ; left-branching

1b. John ‘s brother ‘s dog barked - largest memory load 4

	Type of step	Rule used	Configuration
0	-	-	($\overline{S}$, John ‘s brother ‘s dog barked)
1	shift	NP -> John	(NP $\overline{S}$, ‘s brother ‘s dog barked)
2	lc-predict	NP -> NP POSS N	($\overline{POSS}$ $\overline{N}$ NP $\overline{S}$, ‘s brother ‘s dog barked)
3	match	POSS -> ‘s	($\overline{N}$ NP $\overline{S}$, brother ‘s dog barked)
4	match	N -> brother	(NP $\overline{S}$, ‘s dog barked)
5	lc-predict	NP -> NP POSS N	($\overline{POSS}$ $\overline{N}$ NP $\overline{S}$, ‘s dog barked)
6	match	POSS -> ‘s	($\overline{N}$ NP $\overline{S}$, dog barked)
7	match	N -> dog	(NP $\overline{S}$, barked)
8	lc-connect	S -> NP VP	($\overline{VP}$, barked)
9	shift	V -> barked	(ε , ε)

1c. John ‘s brother ‘s wife ‘s dog barked - largest memory load 4

	Type of step	Rule used	Configuration
0	-	-	($\overline{S}$, John ‘s brother ‘s wife ‘s dog barked)
1	shift	NP -> John	(NP $\overline{S}$, ‘s brother ‘s wife ‘s dog barked)
2	lc-predict	NP -> NP POSS N	($\overline{POSS}$ $\overline{N}$ NP $\overline{S}$, ‘s brother ‘s wife ‘s dog barked)
3	match	POSS -> ‘s	($\overline{N}$ NP $\overline{S}$, brother ‘s wife ‘s dog barked)
4	match	N -> brother	(NP $\overline{S}$, ‘s wife ‘s dog barked)
5	lc-predict	NP -> NP POSS N	($\overline{POSS}$ $\overline{N}$ NP $\overline{S}$, ‘s wife ‘s dog barked)
6	match	POSS -> ‘s	($\overline{N}$ NP $\overline{S}$, wife ‘s dog barked)
7	match	N -> wife	(NP $\overline{S}$, ‘s dog barked)
8	lc-predict	NP -> NP POSS N	($\overline{POSS}$ $\overline{N}$ NP $\overline{S}$, ‘s dog barked)
9	match	POSS -> ‘s	($\overline{N}$ NP $\overline{S}$, dog barked)
10	match	N -> dog	(NP $\overline{S}$, barked)
11	lc-connect	S -> NP VP	($\overline{VP}$, barked)
12	shift	V -> barked	(ε , ε)

Left-corner parser ; Right branching

2b. Mary chased the cat that bit the rat - largest memory load 2

	Type of step	Rule used	Configuration
0	-	-	($\overline{S}$, Mary chased the cat that bit the rat)
1	shift	NP -> Mary	(NP $\overline{S}$, chased the cat that bit the rat)
2	lc-connect	S -> NP VP	($\overline{VP}$, chased the cat that bit the rat)
3	shift	V -> chased	(V $\overline{VP}$, the cat that bit the rat)
4	lc-connect	VP -> V NP	($\overline{NP}$, the cat that bit the rat)
5	shift	D -> the	(D $\overline{NP}$, cat that bit the rat)
6	lc-connect	NP -> D N SRC	(D $\overline{NP}$, cat that bit the rat)
7	match	N -> cat	($\overline{N}$ $\overline{SRC}$, cat that bit the rat)
8	match	THAT -> that	($\overline{SRC}$, that bit the rat)

9	lc-connect	SRC -> THAT VP	(THAT $\overline{SRC}$, bit the rat)
10	shift	V -> bit	($\overline{VP}$, bit the rat
11	lc-connect	VP -> V NP	($\overline{NP}$, the rat)
12	shift	D -> the	(D $\overline{NP}$, rat)
13	lc-connect	NP -> D N	($\overline{N}$, rat)
14	match	N -> rat	(ε , ε)

2c. Mary chased the cat that bit the rat that ate the cheese - memory load 2

	Type of step	Rule used	Configuration
0	-	-	($\overline{S}$, Mary chased the cat that bit the rat that ate the cheese)
1	shift	NP -> Mary	(NP $\overline{S}$, chased the cat that bit the rat that ate the cheese)
2	lc-connect	S -> NP VP	($\overline{VP}$, chased the cat that bit the rat that ate the cheese)
3	shift	V -> chased	(V $\overline{VP}$, the cat that bit the rat that ate the cheese)
4	lc-connect	VP -> V NP	($\overline{NP}$, the cat that bit the rat that ate the cheese)
5	shift	D -> the	(D $\overline{NP}$, cat that bit the rat that ate the cheese)
6	lc-connect	NP -> D N SRC	(D $\overline{NP}$, cat that bit the rat that ate the cheese)
7	match	N -> cat	($\overline{N}$ $\overline{SRC}$, cat that bit the rat that ate the cheese)
8	match	THAT -> that	($\overline{SRC}$, that bit the rat that ate the cheese)
9	lc-connect	SRC -> THAT VP	(THAT $\overline{SRC}$, bit the rat that ate the cheese)
10	shift	V -> bit	($\overline{VP}$, bit the rat that ate the cheese)
11	lc-connect	VP -> V NP	($\overline{NP}$, the rat that ate the cheese)
12	shift	D -> the	(D $\overline{NP}$, rat that ate the cheese)
13	lc-connect	NP -> D N SRC	($\overline{N}$ $\overline{SRC}$, rat that ate the cheese)
14	match	N -> rat	($\overline{SRC}$, that ate the cheese)
15	match	THAT -> that	(THAT $\overline{SRC}$, ate the cheese)

16	lc-connect	SRC -> THAT VP	($\overline{VP}$, ate the cheese)
17	shift	V -> ate	(V $\overline{VP}$, the cheese)
18	lc-connect	VP -> V NP	($\overline{NP}$, the cheese)
19	shift	D -> the	(D $\overline{NP}$, cheese)
20	lc-connect	NP -> D N	($\overline{N}$, cheese)
21	match	N -> cheese	(ε , ε)

Top-down parser ; Center embedding

3b. The rat the cat chased fled - memory load 4

	Type of step	Rule used	Configuration
0	-	-	(S , The rat the cat chased fled)
1	predict	S -> NP VP	(NP VP, the rat the cat chased fled)
2	predict	NP -> D N ORC	(D N ORC VP, the rat the cat chased fled)
3	match	D -> the	(N ORC VP, rat the cat chased fled)
4	match	N -> rat	(ORC VP, the cat chased fled)
5	predict	ORC -> NP V	(NP V VP, the cat chased fled)
6	predict	NP -> D N	(D N V VP, the cat chased fled)
7	match	D -> the	(N V VP, cat chased fled)
8	match	N -> cat	(V VP, chased fled)
9	match	V -> chased	(VP, fled)
10	predict	VP -> V	(V, fled)
11	match	V -> fled	(ε , ε)

3c. The rat the cat the dog bit chased fled - memory load 5

	Type of step	Rule used	Configuration

0	-	-	(S , The rat the cat the dog bit chased fled)
1	predict	S -> NP VP	(NP VP, the rat the cat the dog bit chased fled)
2	predict	NP -> D N ORC	(D N ORC VP, the rat the cat the dog bit chased fled)
3	match	D -> the	(N ORC VP, rat the cat the dog bit chased fled)
4	match	N -> rat	(ORC VP, the cat the dog bit chased fled)
5	predict	ORC -> NP V	(NP V VP, the cat the dog bit chased fled)
6	predict	NP -> D N ORC	(D N ORC V VP, the cat the dog bit chased fled)
7	match	D -> the	(N ORC V VP, cat the dog bit chased fled)
8	match	N -> cat	(ORC V VP, the dog bit chased fled)
9	predict	ORC -> NP V	(NP V V VP, the dog bit chased fled)
10	predict	NP -> D N	(D N V V VP, the dog bit chased fled)
11	match	D -> the	(N V V VP, dog bit chased fled)
12	match	N -> dog	(V V VP, bit chased fled)
13	match	V -> bit	(V VP, chased fled)
14	match	V -> chased	(VP, fled)
15	predict	VP -> V	(V, fled)
16	match	V -> fled	(ε , ε)

Left-corner parser ; Center embedding

3b. The rat the cat chased fled - memory load 5

	Type of step	Rule used	Configuration
0	-	-	($\overline{S}$, The rat the cat chased fled)
1	shift	D -> the	(D $\overline{S}$, rat the cat chased fled)
2	lc-predict	NP -> D N ORC	($\overline{N}$ $\overline{ORC}$ NP $\overline{S}$, rat the cat chased fled)
3	match	N -> rat	($\overline{ORC}$ NP $\overline{S}$, the cat chased fled)
4	shift	D -> the	(D $\overline{ORC}$ NP $\overline{S}$, cat chased fled)
5	lc-predict	NP -> D N	($\overline{N}$ NP $\overline{ORC}$ NP $\overline{S}$, cat chased fled)

6	match	N -> cat	(NP $\overline{ORC}$ NP $\overline{S}$, chased fled)
7	lc-connect	ORC -> NP V	($\overline{V}$ NP $\overline{S}$. chased fled)
8	match	V -> chased	(NP $\overline{S}$, fled)
9	lc-connect	S -> NP VP	($\overline{VP}$, fled)
10	shift	V -> fled	(ε , ε)

3c. The rat the cat the dog bit chased fled - memory load 7

	Type of step	Rule used	Configuration
0	-	-	($\overline{S}$, The rat the cat the dog bit chased fled)
1	shift	D -> the	(D $\overline{S}$, rat the cat the dog bit chased fled)
2	lc-predict	NP -> D N ORC	($\overline{N}$ $\overline{ORC}$ NP $\overline{S}$, rat the cat the dog bit chased fled)
3	match	N -> rat	($\overline{ORC}$ NP $\overline{S}$, the cat the dog bit chased fled)
4	shift	D -> the	(D $\overline{ORC}$ NP $\overline{S}$, cat the dog bit chased fled)
5	lc-predict	NP -> D N ORC	($\overline{N}$ $\overline{ORC}$ NP $\overline{ORC}$ NP $\overline{S}$, cat the dog bit chased fled)
6	match	N -> cat	($\overline{ORC}$ NP $\overline{ORC}$ NP $\overline{S}$, the dog bit chased fled)
7	shift	D -> the	(D $\overline{ORC}$ NP $\overline{ORC}$ NP $\overline{S}$,dog bit chased fled)
8	lc-predict	NP -> D N	($\overline{N}$ NP $\overline{ORC}$ NP $\overline{ORC}$ NP $\overline{S}$,dog bit chased fled)
9	match	N -> dog	(NP $\overline{ORC}$ NP $\overline{ORC}$ NP $\overline{S}$, bit chased fled)
10	lc-connect	ORC -> NP V	($\overline{V}$ NP $\overline{ORC}$ NP $\overline{S}$, bit chased fled)
11	match	V -> bit	(NP $\overline{ORC}$ NP $\overline{S}$, chased fled)
12	lc-connect	ORC -> NP V	($\overline{V}$ NP $\overline{S}$, chased fled)
13	match	V -> chased	(NP $\overline{S}$, fled)
14	lc-connect	S -> NP VP	($\overline{VP}$, fled)
15	shift	V -> fled	(ε , ε)

2. More on stack depth

I would choose hypothesis #2, as it incurs less of a memory load on the left-branching structure. As shown in the parses that follows, Hypothesis 1 has a memory load of 5 vs Hypothesis 2

having a memory load of 4 in the case of 6B. In the case of 6D, Hypothesis 1 has a memory load of 11, while Hypothesis 2 has a memory load of 7.

6b. Hypothesis #1 - memory load 5

	Type of step	Rule used	Configuration
0	-	-	(ε , Mary said quietly John ate)
1	shift	NP -> Mary	(NP, said quietly John ate)
2	shift	SAID -> said	(NP SAID, quietly John ate)
3	shift	ADV -> quietly	(NP SAID ADV, John ate)
4	shift	NP -> John	(NP SAID ADV NP, ate)
5	shift	V -> ate	(NP SAID ADV NP V, ε)
6	reduce	V -> VP	(NP SAID ADV NP VP, ε)
7	reduce	S -> NP VP	(NP SAID ADV S, ε)
8	reduce	VP -> SAID ADV S	(NP VP, ε)
9	reduce	S -> NP VP	(S, ε)

6b. Hypothesis #2 - memory load 4

	Type of step	Rule used	Configuration
0	-	-	(ε , Mary said quietly John ate)
1	shift	NP -> Mary	(NP, said quietly John ate)
2	shift	SAID -> said	(NP SAID, quietly John ate)
3	shift	ADV -> quietly	(NP SAID ADV, John ate)
4	reduce	X -> SAID ADV	(NP X, John ate)
5	shift	NP -> John	(NP X NP, ate)
6	shift	V -> ate	(NP X NP V, ε)
7	reduce	VP -> V	(NP X NP VP, ε)
8	reduce	S -> NP VP	(NP X S, ε)
9	reduce	VP -> X S	(NP VP, ε)
10	reduce	S -> NP VP	(S, ε)

6d. Hypothesis #1 - memory load 11

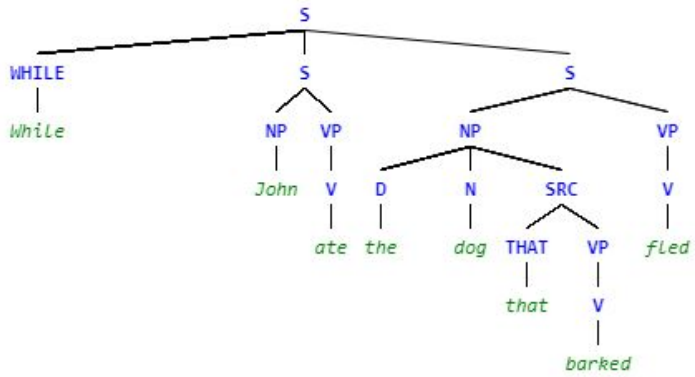
	Type of step	Rule used	Configuration
0	-	-	(ε , John said slowly John said loudly Mary said quietly John ate)
1	shift	NP -> John	(NP, said slowly John said loudly Mary said quietly John ate)
2	shift	SAID -> said	(NP SAID, slowly John said loudly Mary said quietly John ate)
3	shift	ADV -> slowly	(NP SAID ADV, John said loudly Mary said quietly John ate)
4	shift	NP -> John	(NP SAID ADV NP, said loudly Mary said quietly John ate)
5	shift	SAID -> said	(NP SAID ADV NP SAID, loudly Mary said quietly John ate)
6	shift	ADV -> loudly	(NP SAID ADV NP SAID ADV, Mary said quietly John ate)
7	shift	NP -> Mary	(NP SAID ADV NP SAID ADV NP, said quietly John ate)
8	shift	SAID -> said	(NP SAID ADV NP SAID ADV NP SAID, quietly John ate)
9	shift	ADV -> quietly	(NP SAID ADV NP SAID ADV NP SAID ADV, John ate)
10	shift	NP -> John	(NP SAID ADV NP SAID ADV NP SAID ADV NP, ate)
11	shift	VP -> ate	(NP SAID ADV NP SAID ADV NP SAID ADV NP VP, ε)
12	reduce	S -> NP VP	(NP SAID ADV NP SAID ADV NP SAID ADV S, ε)
13	reduce	VP -> SAID ADV S	(NP SAID ADV NP SAID ADV NP VP, ε)
14	reduce	S -> NP VP	(NP SAID ADV NP SAID ADV S, ε)
15	reduce	VP -> SAID ADV S	(NP SAID ADV NP VP, ε

6d. Hypothesis #2 - memory load 7

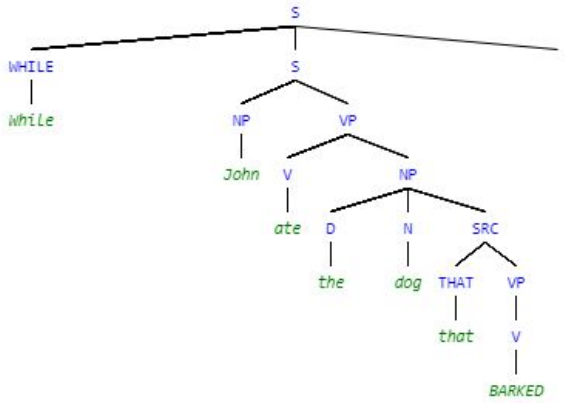
	Type of step	Rule used	Configuration
0	-	-	(ε , John said slowly John said loudly Mary said quietly John ate)

1	shift	NP -> John	(NP, said slowly John said loudly Mary said quietly John ate)
2	shift	SAID -> said	(NP SAID, slowly John said loudly Mary said quietly John ate)
3	shift	ADV -> slowly	(NP SAID ADV, John said loudly Mary said quietly John ate)
4	reduce	X -> SAID ADV	(NP X, John said loudly Mary said quietly John ate)
5	shift	NP -> John	(NP X NP, said loudly Mary said quietly John ate)
6	shift	SAID -> said	(NP X NP SAID, loudly Mary said quietly John ate)
7	shift	ADV -> loudly	(NP X NP SAID ADV, Mary said quietly John ate)
8	reduce	X -> SAID ADV	NP X NP X, Mary said quietly John ate)

3. Garden paths and reanalysis



A. Full correct tree:



Partial tree:

B. Bottom-up parsing full correct tree

	Type of step	Rule used	Configuration
0	-	-	(ε, While John ate the dog that barked fled)
1	shift	WHILE -> while	(WHILE, John ate the dog that barked fled)

2	shift	NP -> John	(WHILE NP, ate the dog that barked fled)
3*	shift	V -> ate	(WHILE NP V, the dog that barked fled)
4	reduce	VP -> V	(WHILE NP VP, the dog that barked fled)
5	reduce	S -> NP VP	(WHILE S, the dog that barked fled)
6	shift	D -> the	(WHILE S D, dog that barked fled)
7	shift	N -> dog	(WHILE S D N, that barked fled)
8	shift	THAT -> that	(WHILE S D N THAT, barked fled)
9	shift	V -> barked	(WHILE S D N THAT V, fled)
10	reduce	VP -> V	(WHILE S D N THAT VP, fled)
11	reduce	SRC -> THAT VP	(WHILE S D N SRC, fled)
11	reduce	NP -> D N SRC	(WHILE S NP, fled)
12	shift	V -> fled	(WHILE S NP V, ε)
13	reduce	VP -> V	(WHILE S NP VP, ε)
14	reduce	S -> NP VP	(WHILE S S, ε)
15	reduce	S -> WHILE S S	(S, ε)

*marks the step where the next step may diverge to wrong configurations
exp. 1

	Type of step	Rule used	Configuration
0	-	-	(ε, While John ate the dog that barked fled)
1	shift	WHILE -> while	(WHILE, John ate the dog that barked fled)
2	shift	NP -> John	(WHILE NP, ate the dog that barked fled)
3*	shift	V -> ate	(WHILE NP V, the dog that barked fled)
4	shift	D -> th	(WHILE NP V D, dog that barked fled)
5	shift	N -> dog	(WHILE NP V D N, that barked fled)
6	reduce	NP -> D N	(WHILE NP V NP, that barked fled)
7	reduce	VP -> V NP	(WHILE NP VP, that barked fled)
8	reduce	S -> NP VP	(WHILE S, that barked fled)

9	shift	THAT -> that	(WHILE S THAT, barked fled)
10	shift	V -> barked	(WHILE S THAT V, fled)
11	reduce	VP -> V	(WHILE S THAT VP, fled)
12	reduce	SRC -> THAT VP	(WHILE S SRC, fled)
13	shift	V -> FLED	(WHILE S SRC V, ε)

exp2.

	Type of step	Rule used	Configuration
0	-	-	(ε, While John ate the dog that barked fled)
1	shift	WHILE -> while	(WHILE, John ate the dog that barked fled)
2	shift	NP -> John	(WHILE NP, ate the dog that barked fled)
3*	shift	V -> ate	(WHILE NP V, the dog that barked fled)
4	shift	D -> the	(WHILE NP V D, dog that barked fled)
5	shift	N -> dog	(WHILE NP V D N, that barked fled)
6	shift	THAT -> that	(WHILE NP V D N THAT, barked fled)
7	shift	V -> barked	(WHILE NP V D N THAT V, fled)
8	reduce	VP -> V	(WHILE NP V D N THAT VP, fled)
9	reduce	SRC -> THAT VP	(WHILE NP V D N SRC, fled)
10	reduce	NP -> D N SRC	(WHILE NP V NP, fled)
11	reduce	VP -> V NP	(WHILE NP VP, fled)
12	reduce	S -> NP VP	(WHILE S, fled)
13	shift	V -> fled	(WHILE S V, ε)

Exp. 3

	Type of step	Rule used	Configuration
0	-	-	(ε, While John ate the dog that barked fled)

1	shift	WHILE -> while	(WHILE, John ate the dog that barked fled)
2	shift	NP -> John	(WHILE NP, ate the dog that barked fled)
3*	shift	V -> ate	(WHILE NP V, the dog that barked fled)
4	shift	D -> the	(WHILE NP V D, dog that barked fled)
5	shift	N -> dog	(WHILE NP V D N, that barked fled)
6	shift	THAT -> that	(WHILE NP V D N THAT, barked fled)
7	shift	V -> barked	(WHILE NP V D N THAT V, fled)
8	reduce	VP -> V	(WHILE NP V D N THAT VP, fled)
9	reduce	SRC -> THAT VP	(WHILE NP V D N SRC, fled)
10	reduce	NP -> D N SRC	(WHILE NP V NP, fled)
11	shift	V -> fled	(WHILE NP V NP V, ε)
12	reduce	ORC -> NP V	(WHILE NP V ORC, ε)

C. Top-down correct tree

	Type of step	Rule used	Configuration
0	-	-	(S, While John ate the dog that barked fled)
1	predict	S -> WHILE S S	(WHILE S S, While John ate the dog that barked fled)
2	match	WHILE -> while	(S S, John ate the dog that barked fled)
3	predict	S -> NP VP	(NP VP S, John ate the dog that barked fled)
4*	match	NP -> John	(VP S, ate the dog that barked fled)
5	predict	VP -> V	(V S, ate the dog that barked fled)
6	match	V -> ate	(S, the dog that barked fled)
7	predict	S -> NP VP	(NP VP, the dog that barked fled)
8	predict	NP -> D N SRC	(D N SRC VP, the dog that barked fled)
9	match	D -> the	(N SRC VP, dog that barked fled)
10	match	N -> dog	(SRC VP, that barked fled)
11	predict	SRC -> THAT VP	(THAT VP VP, that barked fled)

12	match	THAT -> that	(VP VP, barked fled)
13	predict	VP -> V	(V VP, barked fled)
14	match	V -> barked	(VP, fled)
15	predict	VP -> V	(V, fled)
16	match	V -> fled	(ε, ε)

*marks the step where depending on what is predicted, the parse may diverge to wrong configurations

Exp. 1

	Type of step	Rule used	Configuration
0	-	-	(S, While John ate the dog that barked fled)
1	predict	S -> WHILE S S	(WHILE S S, While John ate the dog that barked fled)
2	match	WHILE -> while	(S S, John ate the dog that barked fled)
3	predict	S -> NP VP	(NP VP S, John ate the dog that barked fled)
4*	match	NP -> John	(VP S, ate the dog that barked fled)
5	predict	VP -> V NP	(V NP S, ate the dog that barked fled)
6	match	V -> ate	(NP S, the dog that barked fled)
7	predict	NP -> D N SRC	(D N SRC S, the dog that barked fled)
8	match	D -> the	(N SRC S, dog that barked fled)
9	match	N -> dog	(SRC S, that barked fled)
10	predict	SRC -> THAT VP	(THAT VP S, that barked fled)
11	match	THAT -> that	(VP S, barked fled)
12	predict	VP -> V	(V S, barked fled)
13	match	V -> barked	(S, fled)
14	predict	S -> NP VP	(NP VP, fled)
15	predict	NP -> N	(N VP, fled)

Exp. 2

	Type of step	Rule used	Configuration
0	-	-	(S, While John ate the dog that barked fled)
1	predict	S -> WHILE S S	(WHILE S S, While John ate the dog that barked fled)
2	match	WHILE -> while	(S S, John ate the dog that barked fled)
3	predict	S -> NP VP	(NP VP S, John ate the dog that barked fled)
4*	match	NP -> John	(VP S, ate the dog that barked fled)
5	predict	VP -> V NP	(V NP S, ate the dog that barked fled)
6	match	V -> ate	(NP S, the dog that barked fled)
7	predict	NP -> D N ORC	(D N ORC S, the dog that barked fled)
8	match	D -> the	(N ORC S, dog that barked fled)
9	match	N -> dog	(ORC S, that barked fled)

Exp. 3

	Type of step	Rule used	Configuration
0	-	-	(S, While John ate the dog that barked fled)
1	predict	S -> WHILE S S	(WHILE S S, While John ate the dog that barked fled)
2	match	WHILE -> while	(S S, John ate the dog that barked fled)
3	predict	S -> NP VP	(NP VP S, John ate the dog that barked fled)
4*	match	NP -> John	(VP S, ate the dog that barked fled)
5	predict	VP -> V NP	(V NP S, ate the dog that barked fled)
6	match	V -> ate	(NP S, the dog that barked fled)
7	predict	NP -> D N	(D N S, the dog that barked fled)
8	match	D -> the	(N S, dog that barked fled)
9	match	N -> dog	(S, that barked fled)
10	predict	S -> NP VP	(NP VP, that barked fled)

D. I would set two points of interest, *ate* and *barked*. Based off of where the subject spends more time inspecting the word, we can tell whether they use top-down or bottom-up parsing. A longer inspection time on *ate* may suggest that the subject considers that the ending of a

sentence - which corresponds to top-down parsing, as they parse the first S before beginning the second S. A longer inspection time on *barked* may suggest bottom-up parsing, as they would backtrack from *barked* to parse the VP.

4. FSAs and finite memory, CFGs and unbound memory

A. FSA parsing

	Type of step	Rule used	Configuration
0	-	-	(0A, aaaacbbbb)
1	consume	0A -> 1A a	(1A, aaacbbbb)
2	consume	1A -> 2A a	(2A, aacbbbb)
3	consume	2A -> 3A a	(3A, acbbbb)
4	consume	3A -> 4A a	(4A, cbbbb)
5	consume	4A -> 4B c	(4B, bbbb)
6	consume	4B -> 3B b	(3B, bbb)
7	consume	3B -> 2B b	(2B, bb)
8	consume	2B -> 1B b	(1B, b)
9	consume	1B -> 0B b	(0B, ε)

B. top-down parser

	Type of step	Rule used	Configuration
0	-	-	(S, aaaacbbbb)
1	predict	S -> A S B	(A S B, aaaacbbbb)
2	match	A -> a	(S B, aaacbbbb)
3	predict	S -> A S B	(A S B B, aaacbbbb)
4	match	A -> a	(S B B, aacbbbb)
5	predict	S -> A S B	(A S B B B, aacbbbb)
6	match	A -> a	(S B B B, acbbbb)
7	predict	S -> A S B	(A S B B B B, acbbbb)
8	match	A -> a	(S B B B B, cbbbb)
9	predict	S -> C	(C B B B B, cbbbb)

10	match	C -> c	(B B B B, bbbb)
11	match	B -> b	(B B B, bbb)
12	match	B -> b	(B B, bb)
13	match	B -> b	(B, b)
14	match	B -> b	(ε, ε)

C. bottom-up parser

	Type of step	Rule used	Configuration
0	-	-	(ε, aaaacbbbb)
1	shift	A -> a	(A, aaacbbbb)
2	shift	A -> a	(A A, aacbbbb)
3	shift	A -> a	(A A A, acbbbb)
4	shift	A -> a	(A A A A, cbbbb)
5	shift	C -> c	(A A A A C, bbbb)
6	reduce	S -> C	(A A A A S, bbbb)
7	shift	B -> b	(A A A A S B, bbb)
8	reduce	S -> A S B	(A A A S, bbb)
9	shift	B -> b	(A A A S B, bb)
10	reduce	S -> A S B	(A A S, bb)
11	shift	B -> b	(A A S B, b)
12	reduce	S -> A S B	(A S, b)
13	shift	B -> b	(A S B, ε)
14	reduce	S -> A S B	(S, ε)

D. left-corner parser

	Type of step	Rule used	Configuration
0	-	-	($\overline{S}$, aaaacbbbb)
1	shift	A -> a	(A $\overline{S}$, aaacbbbb)
2	lc-predict	S -> A S B	($\overline{S}$ $\overline{B}$ S $\overline{S}$, aaacbbbb)
3	shift	A -> a	(A $\overline{S}$ $\overline{B}$ S $\overline{S}$, aacbbbb)
4	lc-predict	S -> A S B	($\overline{S}$ $\overline{B}$ S $\overline{S}$ $\overline{B}$ S $\overline{S}$, aacbbbb)
5	shift	A -> a	(A $\overline{S}$ $\overline{B}$ S $\overline{S}$ $\overline{B}$ S $\overline{S}$, aacbbbb)
6	lc-predict	S -> A S B	($\overline{S}$ $\overline{B}$ S $\overline{S}$ $\overline{B}$ S $\overline{S}$ $\overline{B}$ S $\overline{S}$, acbbbb)
7	shift	A -> a	(A $\overline{S}$ $\overline{B}$ S $\overline{S}$ $\overline{B}$ S $\overline{S}$ $\overline{B}$ S $\overline{S}$, acbbbb)
8	lc-predict	S -> A S B	($\overline{S}$ $\overline{B}$ S $\overline{S}$ $\overline{B}$ S $\overline{S}$ $\overline{B}$ S $\overline{S}$ $\overline{B}$ S $\overline{S}$, cbbbb)
9	lc-connect	S -> C	($\overline{C}$ $\overline{B}$ S $\overline{S}$ $\overline{B}$ S $\overline{S}$ $\overline{B}$ S $\overline{S}$ $\overline{B}$ S $\overline{S}$, cbbbb)
10	match	C -> c	($\overline{B}$ S $\overline{S}$ $\overline{B}$ S $\overline{S}$ $\overline{B}$ S $\overline{S}$ $\overline{B}$ S $\overline{S}$, cbbbb)
11	match	B -> b	(S $\overline{S}$ $\overline{B}$ S $\overline{S}$ $\overline{B}$ S $\overline{S}$ $\overline{B}$ S $\overline{S}$, bbbb)
12	match	B -> b	($\overline{B}$ S $\overline{S}$ $\overline{B}$ S $\overline{S}$ $\overline{B}$ S $\overline{S}$, bbb)
13	match	B -> b	($\overline{B}$ S $\overline{S}$ $\overline{B}$ S $\overline{S}$, bb)
14	match	B -> b	($\overline{B}$ S $\overline{S}$, b)
15	match	B -> b	(ε, ε)