Homework 3: CFGs | PDAs | Stack-based parsing

Kevin Liang Ling 185A Due: 08/28/2024, 11:59 PM PDT

This assignment does not involve any programming. You may submit your answers on BruinLearn as a PDF (either scanned or typeset) or as photos of handwritten work. Please make sure your writing is intelligible and the assignment itself is downloadable.

Total: 28 pts + 5 extra credits

1 Converting a CFG to a PDA (5 points)

Convert the CFG below into a pushdown automaton using the algorithm that we discussed in class.

$N = \{\text{VP}, \text{NP}, \text{PP}, \text{V}, \text{P}\}$
$\Sigma = \{\text{watches, spies, telescopes, with}\}$
$I = \{\text{VP}\}$

VP	→	V NP		V	→	watches
VP	→	VP PP		VP	→	watches
NP	→	NP PP		VP	→	spies
PP	→	P NP		NP	→	watches
				NP	→	spies
				NP	→	telescopes
				P	→	with

Give all the pieces of the PDA six-place tuple $\langle Q, \Sigma, \Gamma, I, F, \Delta \rangle$. Two core pieces of this algorithm are given below to jog your memory ...

A. For each rule of the grammar $A \to \psi$, $(q_1, \epsilon, A, q_1, \psi) \in \Delta$.

B. For each symbol $a \in \Sigma_G$, $(q_1, a, a, q_1, \epsilon) \in \Delta$.

$Q = \{q_0, q_1\}$

$I = \{q_0\}$

$F = \{q_1\}$

$\Sigma = \{\text{watches, spies, telescopes, with}\}$

$\Gamma = \{\text{VP, NP, PP, V, P, watches, spies, telescopes, with}\}$

$\Delta = \{(q_0, \epsilon, \epsilon, q_1, S),$
$(q_1, \epsilon, \text{VP}, q_1, \text{V NP}), (q_1, \epsilon, \text{VP}, q_1, \text{VP PP}),$
$(q_1, \epsilon, \text{VP}, q_1, \text{watches}), (q_1, \epsilon, \text{VP}, q_1, \text{spies})$
$(q_1, \epsilon, \text{NP}, q_1, \text{NP PP}), (q_1, \epsilon, \text{NP}, q_1, \text{watches}),$
$(q_1, \epsilon, \text{NP}, q_1, \text{spies}), (q_1, \epsilon, \text{NP}, q_1, \text{telescopes}),$
$(q_1, \epsilon, \text{PP}, q_1, \text{P NP}), (q_1, \epsilon, \text{V}, q_1, \text{watches}), (q_1, \epsilon, \text{P}, q_1, \text{with}),$

$(q_1, watches, watches, q_1, \epsilon)$,

$(q_1, spies, spies, q_1, \epsilon)$,

$(q_1, telescopes, telescopes, q_1, \epsilon)$,

$(q_1, with, with, q_1, \epsilon)\}$

2 Stack depth and embedding structures (15 points + 5 points)

A sample grammar

Here's a simple CFG you should use for most of the questions below.

S → NP VP	N → baby, boy, actor, award, boss
S → WHILE S S	NP → Mary, John
NP → NP POSS N	V → met, saw, won, cried, watched
NP → (D) N (PP) (SRC) (ORC)	D → the
VP → V (NP) (PP)	P → on, in, with
PP → P NP	THAT → that
SRC → THAT VP	POSS → 's
ORC → NP V	WHILE → while

Recall the three kinds of embedding structures we've considered, repeated here. Increasing the depth of center-embedding structures seems to cause a certain kind of processing difficulty in humans, which does not arise when we increase the depth of left-branching or right-branching structures.

1. **Left-branching structures**
   (a) Mary won
   (b) Mary 's baby won
   (c) Mary 's boss 's baby won
2. **Right-branching structures**
   (a) John met the boy
   (b) John met the boy that saw the actor
   (c) John met the boy that saw the actor that won the award
3. **Center-embedding structures**
   (a) the actor won
   (b) the actor the boy met won
   (c) *the actor the boy the baby saw met won

I've put an asterisk on (3c) just to indicate that this sentence produces a "yuck, something went wrong" reaction from English speakers. Logically speaking, this *could* be because the sentence is *ungrammatical*, i.e. not generated by English speakers' mental grammars; but based on the other sentences in (3.), a more attractive hypothesis might be that there is *something else* going wrong in (3c), which is producing the "yuck" reaction that we are recording with the asterisk. (When we want to be careful about this kind of thing, we might say that the asterisk on (3c) indicates only that it is *unacceptable*, which may or may not be because it is *ungrammatical*.[1] Unfortunately there is some inconsistency, in general, about whether asterisks are indicators of unacceptability or ungrammaticality, because the large majority of the time they are taken to coincide.)

So the important generalization is this: considering larger and larger center-embedding structures eventually leads to sentences that produce a "something went wrong" reaction, whereas we do not see this for left-branching or right-branching structures.

[1] Besides (3c), some other examples of sentences that are unacceptable for many English speakers but probably grammatical are 'Police police police police police police' and 'The horse raced past the barn fell'; although they definitely produce a "something wrong" reaction, our guess is that this is best attributed to non-grammatical factors. Some pretty mysterious examples of sentences that are acceptable for many English speakers but probably ungrammatical are 'The actor the boy the baby saw won' and 'More people have been to Russia than I have'. And there are always people pursuing the idea that sentences like 'Who do you wonder whether John met yesterday?', while certainly unacceptable, are in fact grammatical — although people developing the kind of theories we teach in linguistics classes here generally take these to be ungrammatical.

We discovered in class that when the bottom-up parsing schema is applied to left-branching structures, the required memory load (i.e. the largest stack size that is required) *does not* increase when the pattern shown in (1.) is extended to longer and longer sentences. With right-branching and center-embedding structures, on the other hand, the memory load on a bottom-up parser *does* increase as these patterns are extended. We also saw that on right-branching structures, the top-down parser behaves differently: the required memory load for a top-down parser *does not* increase as the pattern in (2.) is extended to longer and longer sentences. But with the top-down parser, alas, the memory load *does* increase when the pattern in (1.) is extended. The memory load for center-embedding also increases with top-down parsing, although we didn't look at this case in class.

We can summarize these findings, and compare them with the facts about humans that we'd like to account for, in the table below. Each cell describes what happens with long sentences of a particular sort *compared to* what happens with short sentences of the same sort. So in the row describing the bottom-up parser, 'no increase' in the left-branching column indicates that (1c) *does not* require any more stack space than (1b) requires, whereas 'increase' in the right-branching column indicates that (2c) *does* require more stack space than (2b) requires.

	left-branching (1.)	right-branching (2.)	center-embedding (3.)
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

A. Your task here — in case you haven't already guessed — is to fill in the remaining three cells of the table. Use the grammar given above and the sentences in (1.)–(3.). So for each cell, you need to consider what happens in the relevant 'b.' sentence and the relevant 'c.' sentence.

- For the 'b.' sentences, show the *full* sequence of configurations from start to goal; use the format of the table on our lecture handout, *showing the transition type and rule used for each step.* — 9 points (3 for each cell)
- For the 'c.' sentences, you don't need to show all the gory details, but for full credit you must at least write down the configurations that impose the largest memory load — 3 points (1 for each cell)
- and clearly state how that load compares with the corresponding 'b.' sentence; — 3 points (1 for each cell)

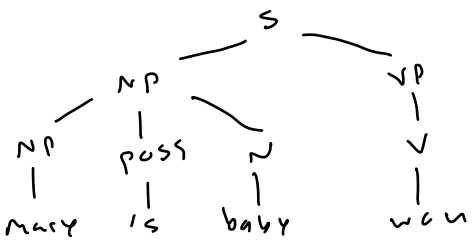
Provide a brief explanation of your thinking in addition might allow me to assign partial credit if any of the specifics are wrong.

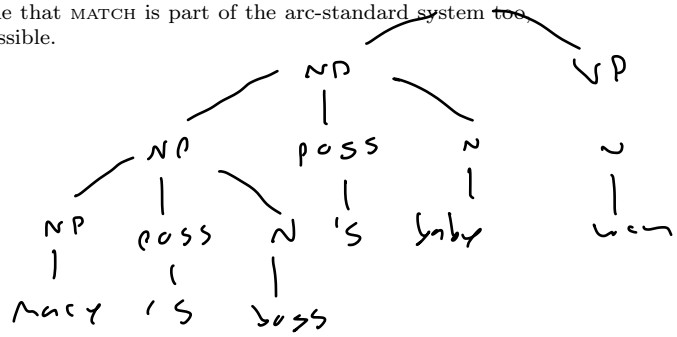
B. The left-corner parsing system that we've introduced so far is, more specifically, known as *arc-eager left-corner parsing*. Another variant is known as *arc-standard left-corner parsing*. In this alternative system, the SHIFT, MATCH and LC-PREDICT rules are unchanged, but there is no LC-CONNECT rule; instead, there is a different fourth rule, which we can call CANCEL, which is defined like this: — optional: 5 points

CANCEL step: $(A\bar{A}\Phi, x_i \dots x_n) \equiv (\Phi, x_i \dots x_n)$
where A is any nonterminal in the grammar

To show how this works, have a look at the two tables below. The table on the left shows the now-familiar, arc-eager left-corner parse for 'the baby saw John'. The table on the right shows the slightly different arc-standard left-corner parse for the same sentence, which uses the new CANCEL rule. Each application of LC-CONNECT in the first table corresponds to one application of LC-PREDICT and one application of CANCEL in the second table, as indicated by the color coding of the transition labels.[2] (The color-coded stack symbols are a further clue to understanding the correspondence.)

[2]Once you get the hang of this, you might also realize that in arc-standard parsing with CANCEL we can do without MATCH too; because MATCH is to SHIFT as LC-CONNECT is to LC-PREDICT. But the decision about what to do with MATCH doesn't have any effect on the memory-load patterns, so let's assume that MATCH is part of the arc-standard system too, just to keep the differences between the two systems as simple as possible.





left-branching (1)

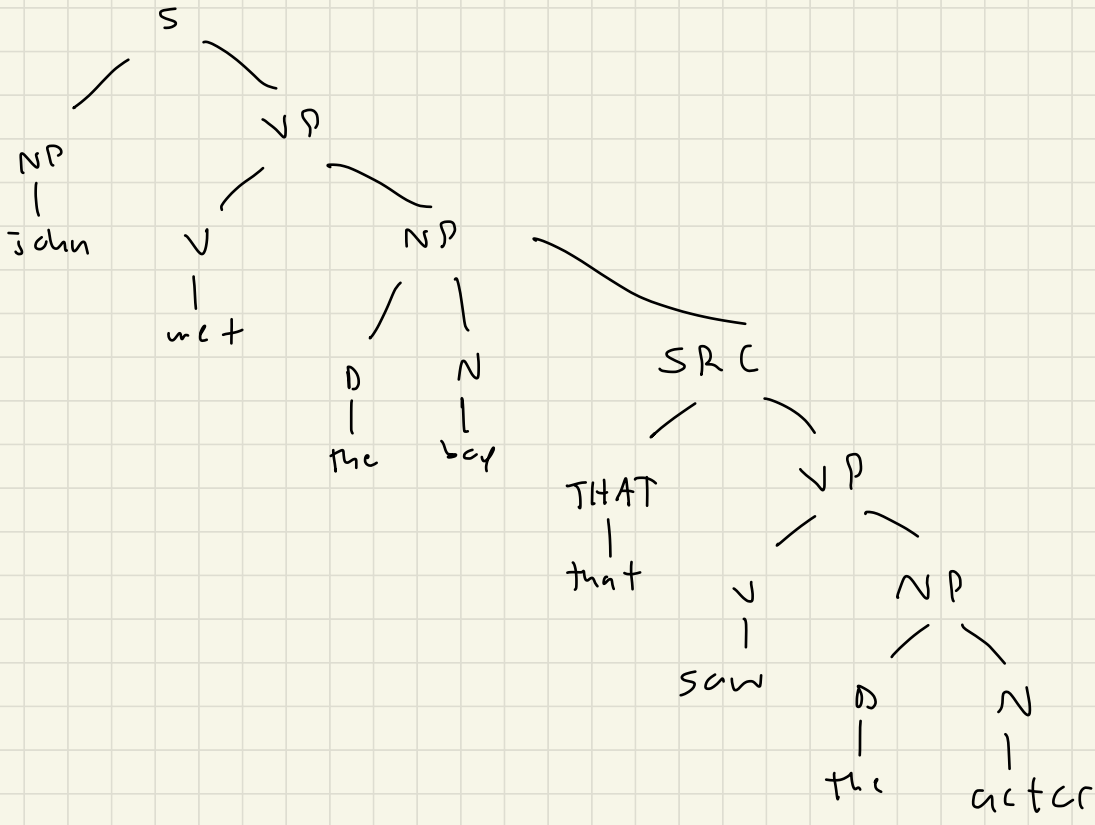
(b) Mary's baby won

	Type of transition	Rule used	Configuration
0	—	—	($\bar{S}$, Mary's baby won)
1	Shift	NP → Mary	(NP $\bar{S}$, 's baby won)
2	LC-Predict	NP → NP Poss N	($\overline{\text{Poss}}$ $\bar{N}$ NP $\bar{S}$, 's baby won)
3	Match	Poss → 's	($\bar{N}$ NP $\bar{S}$, baby won)
4	Match	N → baby	(NP $\bar{S}$, won)
5	LC-Connect	S → NP VP	($\overline{VP}$, won)
6	Shift	V → won	(V $\overline{VP}$, ϵ)
7	LC-Connect	VP → V	(ϵ, ϵ)

(c) Mary's boss 's baby won

	Type of transition	Rule used	Configuration
0	—	—	($\bar{S}$, Mary's boss's baby won)
1	Shift	NP → Mary	(NP $\bar{S}$, 's boss's baby won)
2	LC-Predict	NP → NP Poss N	($\overline{\text{Poss}}$ $\bar{N}$ NP $\bar{S}$, 's boss's baby won)
3	Match	Poss → 's	($\bar{N}$ NP $\bar{S}$, boss's baby won)
4	Match	N → boss	(NP $\bar{S}$, 's baby won)
5	LC-Predict	NP → NP Poss N	($\overline{\text{Poss}}$ $\bar{N}$ NP $\bar{S}$, 's baby won)
6	Match	Poss → 's	($\bar{N}$ NP $\bar{S}$, baby won)
7	Match	N → baby	(NP $\bar{S}$, won)
8	LC-Connect	S → NP VP	($\overline{VP}$, won)
9	Shift	V → won	(V $\overline{VP}$, ϵ)
10	LC-Connect	VP → V	(ϵ, ϵ)

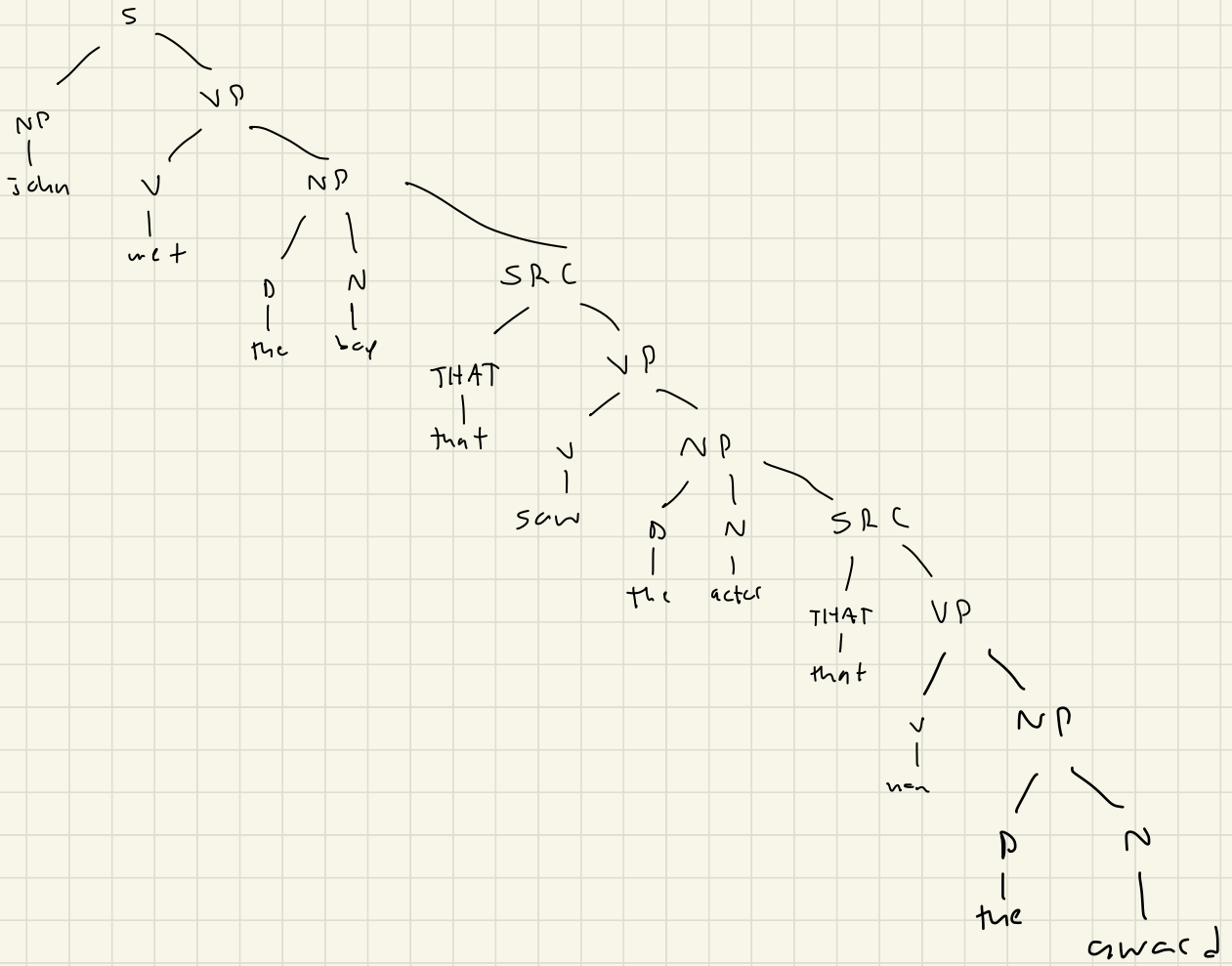
The largest memory load required in 1c is equivalent to the largest memory load in 1b.


S
NP
John
VP
V
met
NP
D
the
N
boy
SRC
THAT
that
VP
V
saw
NP
D
the
N
actor

right-branching (2)

(b) John met the boy that saw the actor

	Type of transition	Rule used	Configuration
0	—	—	($\bar{S}$, John met the boy that saw the actor)
1	Shift	NP → John	(NP $\bar{S}$, met the boy that saw the actor)
2	LC-Connect	S → NP VP	($\bar{VP}$, met the boy that saw the actor)
3	Shift	V → met	(V $\bar{VP}$, the boy that saw the actor)
4	LC-Connect	VP → V NP	($\bar{NP}$, the boy that saw the actor)
5	Shift	D → the	(D $\bar{NP}$, boy that saw the actor)
6	LC-Connect	NP → D N SRC	($\bar{N}$ $\bar{SRC}$, boy that saw the actor)
7	Match	N → boy	($\bar{SRC}$, that saw the actor)
8	Shift	THAT → that	(THAT $\bar{SRC}$, saw the actor)
9	LC-Connect	SRC → THAT VP	($\bar{VP}$, saw the actor)
10	Shift	V → saw	(V $\bar{VP}$, the actor)
11	LC-Connect	VP → V NP	($\bar{NP}$, the actor)
12	Shift	D → the	(D $\bar{NP}$, actor)
13	LC-Connect	NP → D N	($\bar{N}$, actor)
14	Match	N → actor	(ϵ, ϵ)


S
NP
John
VP
V
met
NP
D
the
N
boy
SRC
THAT
that
VP
V
saw
NP
D
the
N
actor
SRC
THAT
that
VP
V
won
NP
D
the
N
award

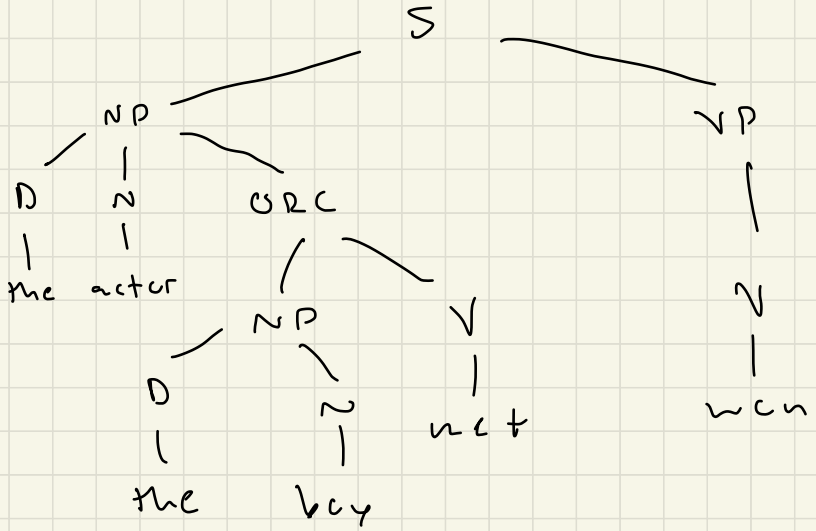
right-branching (2)

(L) John met the boy that saw the actor that won the award

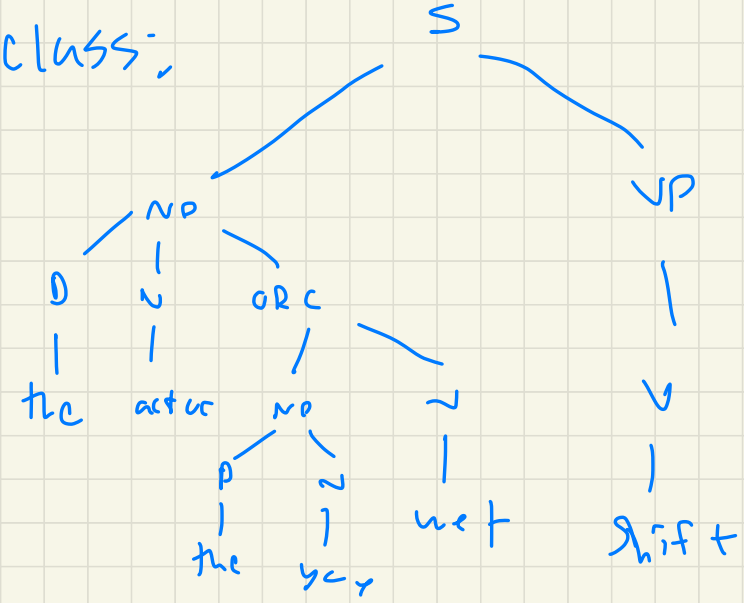
	Type of transition	Rule used	Configuration
0	—	—	($\bar{S}$, John met the boy that saw the actor that won the award)
1	Shift	NP → John	(NP $\bar{S}$, met the boy that saw the actor that won the award)
2	LC-Connect	S → NP VP	($\bar{VP}$, met the boy that saw the actor that won the award)
3	Shift	V → met	(V $\bar{VP}$, the boy that saw the actor that won the award)
4	LC-Connect	VP → V NP	($\bar{NP}$, the boy that saw the actor that won the award)
5	Shift	D → the	(D $\bar{NP}$, boy that saw the actor that won the award)
6	LC-Connect	NP → D N SRC	($\bar{N}$ $\bar{SRC}$, boy that saw the actor that won the award)
7	Match	N → boy	($\bar{SRC}$, that saw the actor that won the award)
8	Shift	THAT → that	(THAT $\bar{SRC}$, saw the actor that won the award)
9	LC-Connect	SRC → THAT VP	($\bar{VP}$, saw the actor that won the award)
10	Shift	V → saw	(V $\bar{VP}$, the actor that won the award)
11	LC-Connect	VP → V NP	($\bar{NP}$, the actor that won the award)
12	Shift	D → the	(D $\bar{NP}$, actor that won the award)
13	LC-Connect	NP → D N SRC	($\bar{N}$ $\bar{SRC}$, actor that won the award)
14	Match	N → actor	($\bar{SRC}$, that won the award)
15	Shift	THAT → that	(THAT $\bar{SRC}$, won the award)
16	LC-Connect	SRC → THAT VP	($\bar{VP}$, won the award)
17	Shift	V → won	(V $\bar{VP}$, the award
18	LC-Connect	VP → V NP	($\bar{NP}$, the award)

19	Shift	D → the	(D $\overline{NP}$, award)
20	LC-connect	NP → D N	($\overline{N}$, award
21	match	N → award	(t, t)

The largest memory load required in 2C is equivalent to the largest memory load in 2b.



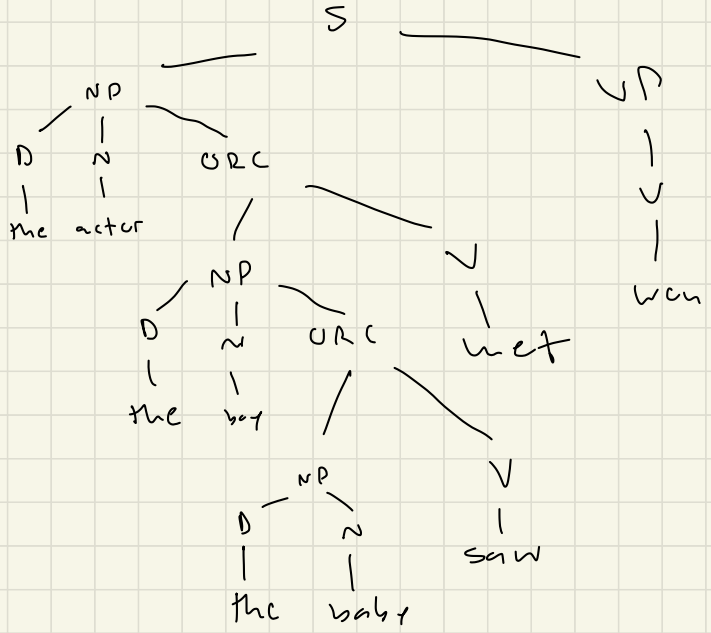
Class:



3, Center-embeddings

(b) the actor the boy met won

	Type of transition	Rule used	Configuration
0	—	—	($\bar{S}$, the actor the boy met won)
1	Shift	D → the	(D $\bar{S}$, actor the boy met won)
2	LC-Predict	NP → D N ORC	($\bar{N}$ $\bar{ORC}$ NP $\bar{S}$, actor the boy met won)
3	Match	N → actor	($\bar{ORC}$ NP $\bar{S}$, the boy met won)
4	Shift	D → the	(D $\bar{ORC}$ NP $\bar{S}$, boy met won)
5	LC-Predict	NP → D N	($\bar{N}$ NP $\bar{ORC}$ NP $\bar{S}$, boy met won)
6	Match	N → boy	(NP $\bar{ORC}$ NP $\bar{S}$, met won)
7	LC-Connect	ORC → NP V	($\bar{V}$ NP $\bar{S}$, met won)
8	Match	V → met	(NP $\bar{S}$, won)
9	LC-Connect	S → NP VP	($\bar{VP}$, won)
10	Shift	V → won	(V $\bar{VP}$, ε)
11	LC-Connect	VP → V	(ε, ε)

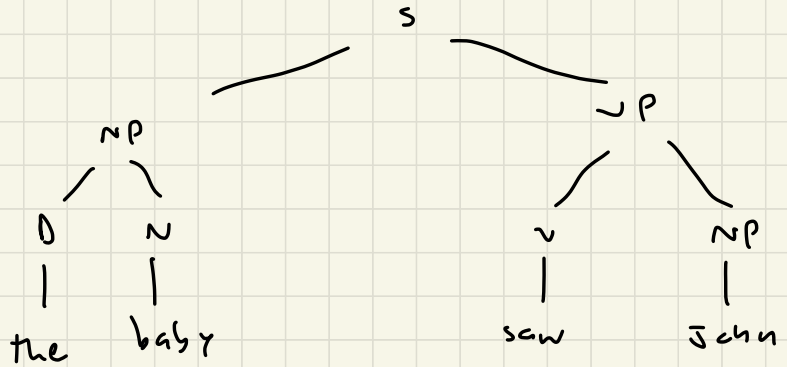

S
NP
VP
D
N
ORC
V
the
actor
NP
D
N
ORC
V
met
the
boy
NP
V
D
N
saw
the
baby

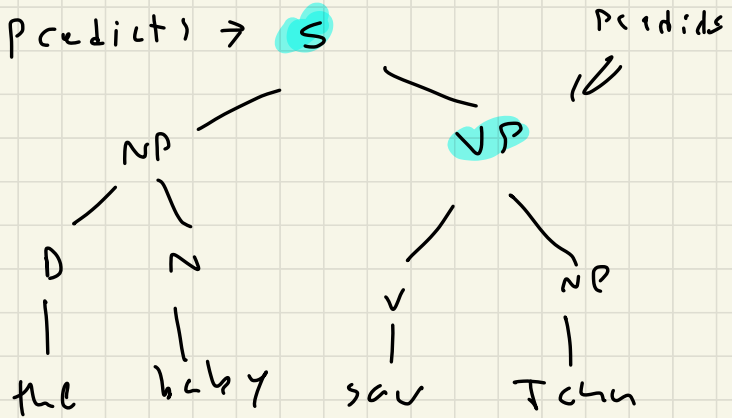
3 Center-embedding

(c) the actor the boy the baby saw met won

	Type of transition	Rule used	Configuration
0	—	—	($\bar{S}$, the actor the boy the baby saw met won)
1	Shift	D → the	(D $\bar{S}$, actor the boy the baby saw met won)
2	LC-Predict	NP → D N ORC	($\bar{N}$ $\overline{ORC}$ NP $\bar{S}$, actor the boy the baby saw met won)
3	Match	N → actor	($\overline{ORC}$ NP $\bar{S}$, the boy the baby saw met won)
4	Shift	D → the	(D $\overline{ORC}$ NP $\bar{S}$, boy the baby saw met won)
5	LC-Predict	NP → D N ORC	($\bar{N}$ $\overline{ORC}$ NP $\overline{ORC}$ NP $\bar{S}$, boy the baby saw met won)
6	Match	N → boy	($\overline{ORC}$ NP $\overline{ORC}$ NP $\bar{S}$, the baby saw met won)
7	Shift	D → the	(D $\overline{ORC}$ NP $\overline{ORC}$ NP $\bar{S}$, baby saw met won)
8	LC-Predict	NP → D N	($\bar{N}$ NP $\overline{ORC}$ NP $\overline{ORC}$ NP $\bar{S}$, baby saw met won)
9	Match	N → baby	($\bar{N}$P $\overline{ORC}$ NP $\overline{ORC}$ NP $\bar{S}$, saw met won)
10	LC-Connect	ORC → NP V	($\bar{V}$ NP $\overline{ORC}$ NP $\bar{S}$, saw met won)
11	Match	V → saw	(NP $\overline{ORC}$ NP $\bar{S}$, met won)
12	LC-Connect	ORC → NP V	($\bar{V}$ NP $\bar{S}$, met won)
13	Match	V → met	(NP $\bar{S}$, won)
14	LC-Connect	S → NP VP	($\overline{VP}$, won)
15	Shift	V → won	(V $\overline{VP}$, won)
16	LC-Connect	VP → V	(ϵ, ϵ)

The largest memory load required in 3C is larger than the largest memory load required in 3B.





$_0$ the $_1$ baby $_2$ saw $_3$ John $_4$

old

Type of transition	Rule used	Configuration
—	—	$(\bar{S}, 0)$
SHIFT	D → the	$(D\ \bar{S}, 1)$
LC-PREDICT	NP → D N	$(\bar{N}\ NP\ \bar{S}, 1)$
MATCH	N → baby	$(NP\ \bar{S}, 2)$
LC-CONNECT	S → NP VP	$(\overline{VP}, 2)$
SHIFT	V → saw	$(V\ \overline{VP}, 3)$
LC-CONNECT	VP → V NP	$(\overline{NP}, 3)$
MATCH	NP → John	$(\epsilon, 4)$

new

Type of transition	Rule used	Configuration
—	—	$(\bar{S}, 0)$
SHIFT	D → the	$(D\ \bar{S}, 1)$
LC-PREDICT	NP → D N	$(\bar{N}\ NP\ \bar{S}, 1)$
MATCH	N → baby	$(NP\ \bar{S}, 2)$
LC-PREDICT	S → NP VP	$(\overline{VP}\ S\ \bar{S}, 2)$
SHIFT	V → saw	$(V\ \overline{VP}\ S\ \bar{S}, 3)$
LC-PREDICT	VP → V NP	$(\overline{NP}\ VP\ \overline{VP}\ S\ \bar{S}, 3)$
MATCH	NP → John	$(VP\ \overline{VP}\ S\ \bar{S}, 4)$
CANCEL	—	$(S\ \bar{S}, 4)$
CANCEL	—	$(\epsilon, 4)$

How does this arc-standard variant of left-corner parsing differ from the arc-eager version that you used above, with respect to memory load patterns (i.e. "increase" vs. "no increase") for left-branching, right-branching and center-embedding structures? For full credit you must at least (i) show the complete sequence of configurations that an arc-standard left-corner parser goes through to process sentence (3b), using the format of the table as in our handout, and (ii) write down the configurations that impose the largest memory load for all three 'b.' sentences and all three 'c.' sentences.

- new

3b. the actor the boy met ran

	left-branching	right branching	center embedding
arc-standard variant	no increase	increase	increase

S
NP
VP
NP
POSS
N
V
Mary
's
baby
cries

left-branching

arc-standard variant

(15) Mary's baby won

	Type of transition	Rule used	Configuration
0	—	—	($\bar{S}$, Mary's baby won)
1	Shift	NP → Mary	(NP $\bar{S}$, 's baby won)
2	LC-Predict	NP → NP Poss N	($\overline{\text{Poss}}$ $\bar{N}$ NP $\bar{S}$, 's baby won)
3	Match	Poss → 's	($\bar{N}$ NP $\bar{S}$, baby won)
4	Match	N → baby	(NP $\bar{S}$, won)
5	LC-Predict	S → NP VP	($\overline{VP}$ S $\bar{S}$, won)
6	Shift	V → won	(V $\overline{VP}$ S $\bar{S}$, t)
7	LC-Predict	VP → V	(VP $\overline{VP}$ S $\bar{S}$, t)
8	Cancel	—	(S $\bar{S}$, t)
9	Cancel	—	(t, t)

* largest memory load

S
NP
VP
NP
poss
N
V
NP
poss
N
's
baby
cry
Mary
's
boss

left-branching

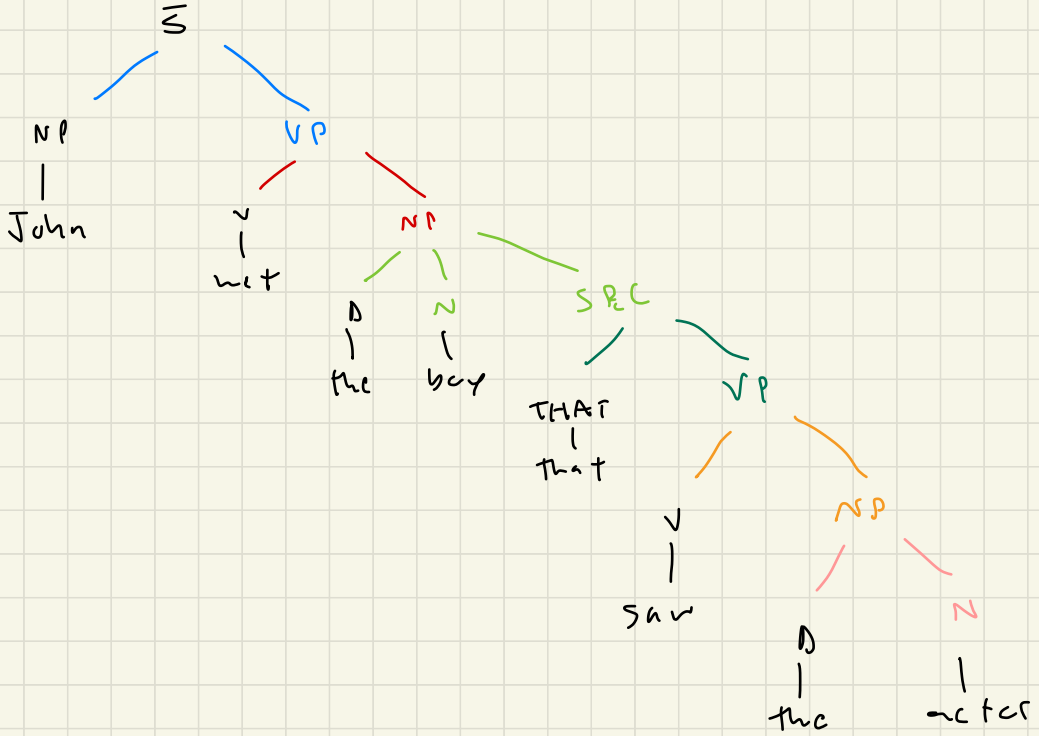
arc-standard variant

(1C) Mary's boss's baby won

	Type of transition	Rule used	Configuration
0	—	—	($\bar{S}$, Mary's boss's baby won)
1	Shift	NP → Mary	(NP $\bar{S}$, 's boss's baby won)
2	LC-Predict	NP → NP Poss N	(Poss $\bar{N}$ NP $\bar{S}$, 's boss's baby won)
3	match	Poss → 's	($\bar{N}$ NP $\bar{S}$, boss's baby won)
4	match	N → boss	(NP $\bar{S}$, 's baby won)
5	LC-Predict	NP → NP Poss N	(Poss $\bar{N}$ NP $\bar{S}$, 's baby won)
6	match	Poss → 's	($\bar{N}$ NP $\bar{S}$, baby won)
7	match	N → baby	(NP $\bar{S}$, won)
8	LC-Predict	S → NP VP	($\bar{VP}$ S $\bar{S}$, won)
9	shift	V → won	(V $\bar{VP}$ S $\bar{S}$, ε)
10	LC-Predict	VP → V	(VP $\bar{VP}$ S $\bar{S}$, ε)
11	cancel	—	(S $\bar{S}$, ε)
12	cancel	—	(ε, ε)

★ largest memory load

The largest memory loads for 1B and 1C are equivalent.

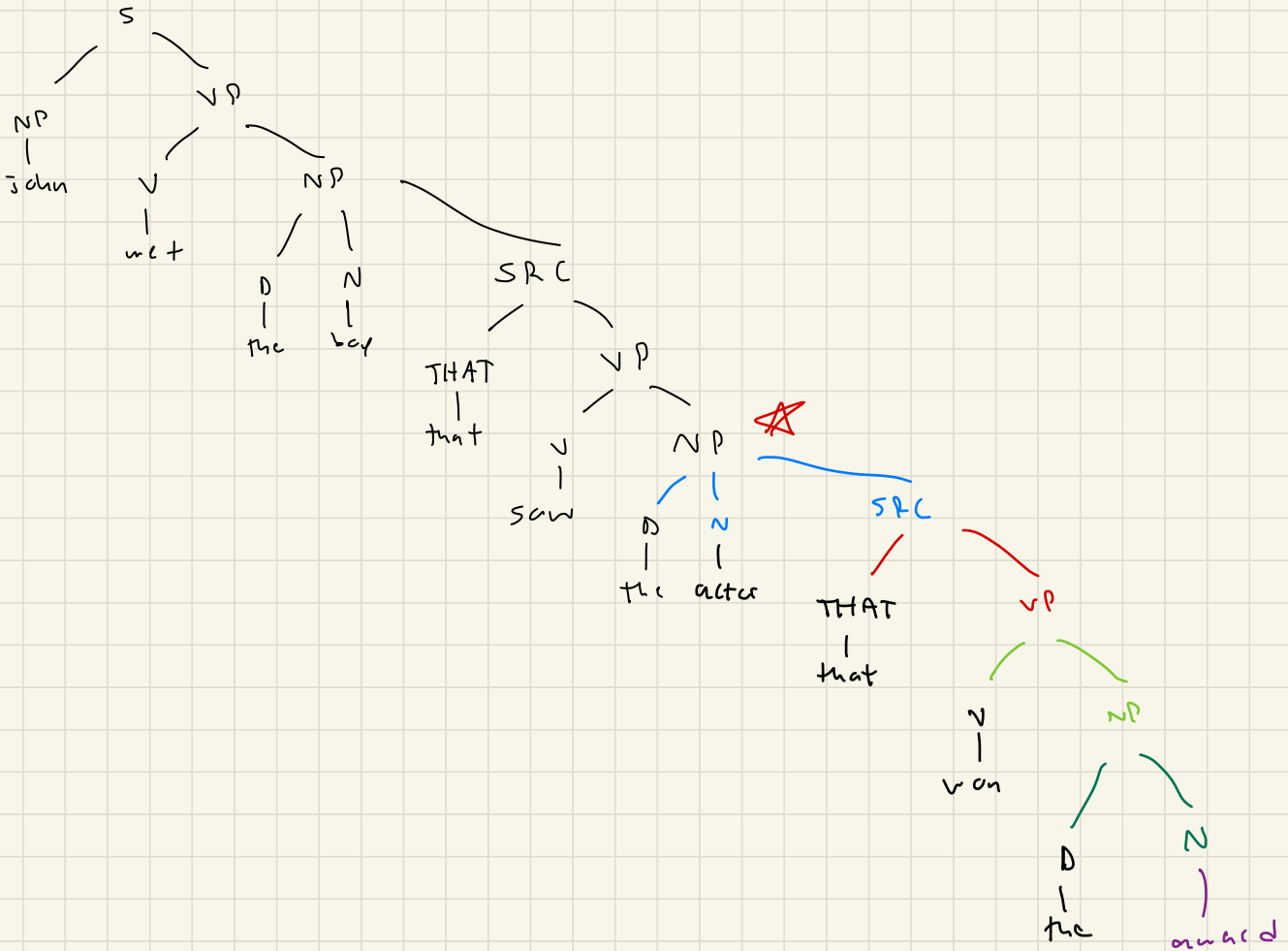

S
NP
John
VP
V
met
NP
D
the
N
boy
SBC
THAT
that
VP
V
saw
NP
D
the
N
actor

right-branching

(26) John met the boy that saw the actor — arc-standard variant

	Type of transition	Rule used	Configuration
0	—	—	($\bar{S}$, John met the boy that saw the actor)
1	Shift	NP → John	(NP $\bar{S}$, met the boy that saw the actor)
2	LC-Predict	S → NP VP	($\bar{V}P$ S $\bar{S}$, met the boy that saw the actor)
3	Shift	V → met	(V $\bar{V}P$ S $\bar{S}$, the boy that saw the actor)
4	LC-Predict	VP → V NP	($\bar{N}P$ VP $\bar{V}P$ S $\bar{S}$, the boy that saw the actor)
5	Shift	D → the	(D $\bar{N}P$ VP $\bar{V}P$ S $\bar{S}$, boy that saw the actor)
6	LC-Predict	NP → D N SRC	($\bar{N}$ SRC NP $\bar{N}P$ VP $\bar{V}P$ S $\bar{S}$, boy that saw the actor)
7	Match	N → boy	(SRC NP $\bar{N}P$ VP $\bar{V}P$ S $\bar{S}$, that saw the actor)
8	Shift	THAT → that	(THAT SRC NP $\bar{N}P$ VP $\bar{V}P$ S $\bar{S}$, saw the actor)
9	LC-Predict	SRC → THAT VP	($\bar{V}P$ SRC SRC NP $\bar{N}P$ VP $\bar{V}P$ S $\bar{S}$, saw the actor)
10	Shift	V → saw	(V $\bar{V}P$ SRC SRC NP $\bar{N}P$ VP $\bar{V}P$ S $\bar{S}$, the actor)
11	LC-Predict	VP → V NP	($\bar{N}P$ VP $\bar{V}P$ SRC SRC NP $\bar{N}P$ VP $\bar{V}P$ S $\bar{S}$, the actor)
12	Shift	D → the	(D $\bar{N}P$ VP $\bar{V}P$ SRC SRC NP $\bar{N}P$ VP $\bar{V}P$ S $\bar{S}$, actor)
13	LC-Predict	NP → D N	(N NP $\bar{N}P$ VP $\bar{V}P$ SRC SRC NP $\bar{N}P$ VP $\bar{V}P$ S $\bar{S}$, actor) *
14	Match	N → actor	(NP $\bar{N}P$ VP $\bar{V}P$ SRC SRC NP $\bar{N}P$ VP $\bar{V}P$ S $\bar{S}$, ϵ)
15	cancel	—	(VP $\bar{V}P$ SRC SRC NP $\bar{N}P$ VP $\bar{V}P$ S $\bar{S}$, ϵ)
16	cancel	—	(SRC SRC NP $\bar{N}P$ VP $\bar{V}P$ S $\bar{S}$, ϵ)
17	cancel	—	(NP $\bar{N}P$ VP $\bar{V}P$ S $\bar{S}$, ϵ)
18	cancel	—	(VP $\bar{V}P$ S $\bar{S}$, ϵ)
19	cancel	—	(S $\bar{S}$, ϵ)
20	cancel	—	(ϵ, ϵ)

* largest memory load


S
NP
VP
John
V
NP
met
D
N
the
boy
SRC
THAT
VP
that
V
NP
saw
D
N
the
actor
SRC
THAT
VP
that
V
NP
won
D
N
the
award

right-branching

(2c) John met the boy that saw the acter that won the award

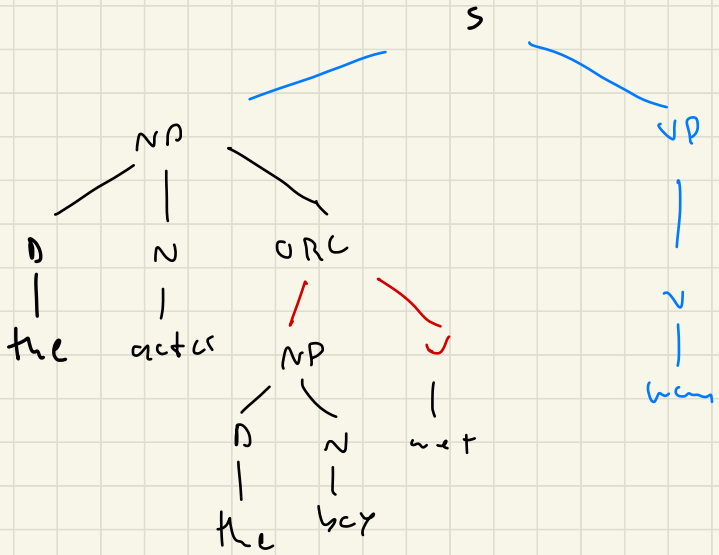
arc-standard variant

	Type of transition	Rule used	Configuration
0	—	—	($\bar{S}$, John met the boy that saw the acter)
1	Shift	NP → John	(NP $\bar{S}$, met the boy that saw the acter that won the award)
2	LC-Predict	S → NP VP	($\bar{VP}$ S $\bar{S}$, met the boy that saw the acter that won the award)
3	Shift	V → met	(V $\bar{VP}$ S $\bar{S}$, the boy that saw the acter that won the award)
4	LC-Predict	VP → V NP	($\bar{NP}$ VP $\bar{VP}$ S $\bar{S}$, the boy that saw the acter that won the award)
5	Shift	D → the	(D $\bar{NP}$ VP $\bar{VP}$ S $\bar{S}$, boy that saw the acter that won the award)
6	LC-Predict	NP → D N SRC	($\bar{N}$ $\bar{SRC}$ NP $\bar{NP}$ VP $\bar{VP}$ S $\bar{S}$, boy that saw the acter that won the award)
7	match	N → boy	($\bar{SRC}$ NP $\bar{NP}$ VP $\bar{VP}$ S $\bar{S}$, that saw the acter that won the award)
8	Shift	THAT → that	(THAT $\bar{SRC}$ NP $\bar{NP}$ VP $\bar{VP}$ S $\bar{S}$, saw the acter that won the award)
9	LC-Predict	SRC → THAT VP	($\bar{VP}$ SRC $\bar{SRC}$ NP $\bar{NP}$ VP $\bar{VP}$ S $\bar{S}$, saw the acter that won the award)
10	Shift	V → saw	(V $\bar{VP}$ SRC $\bar{SRC}$ NP $\bar{NP}$ VP $\bar{VP}$ S $\bar{S}$, the acter that won the award)
11	LC-Predict	VP → V NP	($\bar{NP}$ VP $\bar{VP}$ SRC $\bar{SRC}$ NP $\bar{NP}$ VP $\bar{VP}$ S $\bar{S}$, the acter that won the award)
12	Shift	D → the	(D $\bar{NP}$ VP $\bar{VP}$ SRC $\bar{SRC}$ NP $\bar{NP}$ VP $\bar{VP}$ S $\bar{S}$, acter that won the award)
13	LC-Predict	NP → D N SRC	($\bar{N}$ $\bar{SRC}$ NP $\bar{NP}$ VP $\bar{VP}$ SRC $\bar{SRC}$ NP $\bar{NP}$ VP $\bar{VP}$ S $\bar{S}$, acter that won the award)
14	match	N → acter	($\bar{SRC}$ NP $\bar{NP}$ VP $\bar{VP}$ SRC $\bar{SRC}$ NP $\bar{NP}$ VP $\bar{VP}$ S $\bar{S}$, that won the award)
15	Shift	THAT → that	(THAT $\bar{SRC}$ NP $\bar{NP}$ VP $\bar{VP}$ SRC $\bar{SRC}$ NP $\bar{NP}$ VP $\bar{VP}$ S $\bar{S}$, won the award)
16	LC-Predict	SRC → THAT VP	($\bar{VP}$ SRC $\bar{SRC}$ NP $\bar{NP}$ VP $\bar{VP}$ SRC $\bar{SRC}$ NP $\bar{NP}$ VP $\bar{VP}$ S $\bar{S}$, won the award)
17	Shift	V → won	(V VP SRC $\bar{SRC}$ NP $\bar{NP}$ VP $\bar{VP}$ SRC $\bar{SRC}$ NP $\bar{NP}$ VP $\bar{VP}$ S $\bar{S}$, the award)
18	LC-Predict	VP → V NP	($\bar{NP}$ VP $\bar{VP}$ SRC $\bar{SRC}$ NP $\bar{NP}$ VP $\bar{VP}$ SRC $\bar{SRC}$ NP $\bar{NP}$ VP $\bar{VP}$ S $\bar{S}$, the award)
19	Shift	D → the	(D $\bar{NP}$ VP $\bar{VP}$ SRC $\bar{SRC}$ NP $\bar{NP}$ VP $\bar{VP}$ SRC $\bar{SRC}$ NP $\bar{NP}$ VP $\bar{VP}$ S $\bar{S}$, award)
20	LC-Predict	NP → D N	(N NP $\bar{NP}$ VP $\bar{VP}$ SRC $\bar{SRC}$ NP $\bar{NP}$ VP $\bar{VP}$ SRC $\bar{SRC}$ NP $\bar{NP}$ VP $\bar{VP}$ S $\bar{S}$, award)

largest memory load

21	match	N → award	$(NP\ \overline{NP}\ VP\ \overline{VP}\ SRC\ \overline{SRC}\ NP\ \overline{NP}\ VP\ \overline{VP}\ SRC\ \overline{SRC}\ NP\ \overline{NP}\ VP\ \overline{VP}\ S\ \overline{S},\ t)$
22	cancel	—	$(VP\ \overline{VP}\ SRC\ \overline{SRC}\ NP\ \overline{NP}\ VP\ \overline{VP}\ SRC\ \overline{SRC}\ NP\ \overline{NP}\ VP\ \overline{VP}\ S\ \overline{S},\ t)$
23	cancel	—	$(SRC\ \overline{SRC}\ NP\ \overline{NP}\ VP\ \overline{VP}\ SRC\ \overline{SRC}\ NP\ \overline{NP}\ VP\ \overline{VP}\ S\ \overline{S},\ t)$
24	cancel	✓	$(NP\ \overline{NP}\ VP\ \overline{VP}\ SRC\ \overline{SRC}\ NP\ \overline{NP}\ VP\ \overline{VP}\ S\ \overline{S},\ t)$
25	cancel	—	$(VP\ \overline{VP}\ SRC\ \overline{SRC}\ NP\ \overline{NP}\ VP\ \overline{VP}\ S\ \overline{S},\ t)$
26	cancel	—	$(SRC\ \overline{SRC}\ NP\ \overline{NP}\ VP\ \overline{VP}\ S\ \overline{S},\ t)$
27	cancel	✓	$(NP\ \overline{NP}\ VP\ \overline{VP}\ S\ \overline{S},\ t)$
28	cancel	—	$(VP\ \overline{VP}\ S\ \overline{S},\ t)$
29	cancel	—	$(S\ \overline{S},\ t)$
30	cancel	—	$(t,\ t)$

The largest memory load required for 2C is larger than the largest memory load required for 2D.

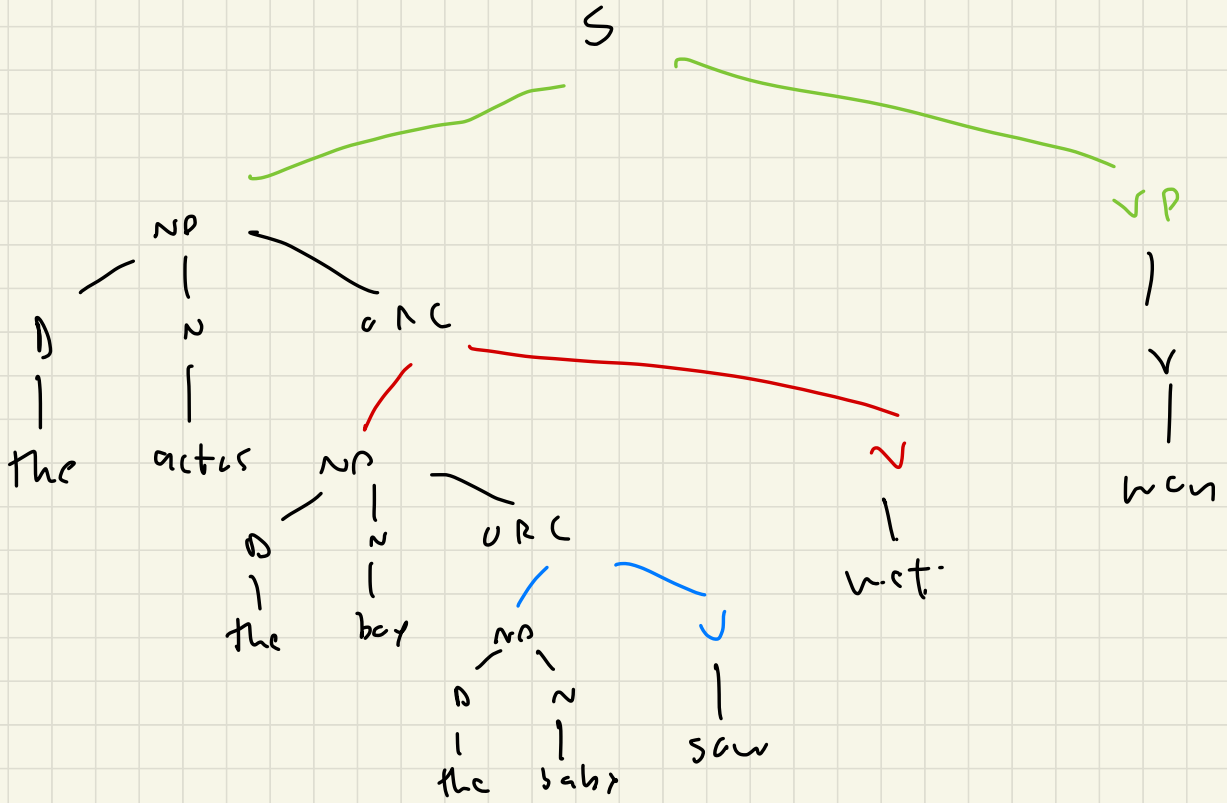

S
NP
VP
D
N
ORC
V
the
actors
NP
met
D
N
the
boy

Centre-embeddings

arc-standard variant

36 the actor the boy met won

	Type of transition	Rule used	Configuration
0	—	—	($\bar{S}$, the actor the boy met won)
1	shift	D → the	(D $\bar{S}$, actor the boy met won)
2	LC-Predict	NP → D N ORC	($\bar{N}$ $\overline{ORC}$ NP $\bar{S}$, actor the boy met won)
3	match	N → actor	($\overline{ORC}$ NP $\bar{S}$, the boy met won)
4	shift	D → the	(D $\overline{ORC}$ NP $\bar{S}$, boy met won)
5	LC-Predict	NP → D N	($\bar{N}$ NP $\overline{ORC}$ NP $\bar{S}$, boy met won) ★
6	match	N → boy	(NP $\overline{ORC}$ NP $\bar{S}$, met won)
7	LC-Predict	ORC → NP V	($\bar{V}$ ORC $\overline{ORC}$ NP $\bar{S}$, met won) ★ largest memory load
8	match	V → met	(ORC $\overline{ORC}$ NP $\bar{S}$, won)
9	Cancel	—	(NP $\bar{S}$, won)
10	LC-Predict	S → NP VP	($\bar{V}P$ S $\bar{S}$, won)
11	shift	V → won	(V $\bar{V}P$ S $\bar{S}$, ϵ)
12	LC-Predict	VP → V	(VP $\bar{V}P$ S $\bar{S}$, ϵ)
13	cancel	—	(S $\bar{S}$, ϵ)
14	cancel	—	(ϵ, ϵ)


S
NP
VP
D
N
ORC
the
actus
NP
N
V
D
N
ORC
the
boy
NP
V
D
N
saw
the
baby
was
met

(c) the actor the boy the baby saw met wcn

(center-embedding arc-standard variant)

	Type of transition	Rule used	Configuration
0	—	—	($\bar{S}$, the actor the boy the baby saw met wcn)
1	Shift	D → the	(D $\bar{S}$, actor the boy the baby saw met wcn)
2	LC-Predict	NP → D N ORC	($\bar{N}$ ORC NP $\bar{S}$, actor the boy the baby saw met wcn)
3	Match	N → actor	(ORC NP $\bar{S}$, the boy the baby saw met wcn)
4	Shift	D → the	(D ORC NP $\bar{S}$, boy the baby saw met wcn)
5	LC-Predict	NP → D N ORC	($\bar{N}$ ORC NP ORC NP $\bar{S}$, boy the baby saw met wcn)
6	Match	N → boy	(ORC NP ORC NP $\bar{S}$, the baby saw met wcn)
7	Shift	D → the	(D ORC NP ORC NP $\bar{S}$, baby saw met wcn)
8	LC-Predict	NP → D N	($\bar{N}$ NP ORC NP ORC NP $\bar{S}$, baby saw met wcn)
9	Match	N → baby	(NP ORC NP ORC NP $\bar{S}$, saw met wcn)
10	LC-Predict	ORC → NP V	($\bar{V}$ ORC ORC NP ORC NP $\bar{S}$, saw met wcn)
11	Match	V → saw	(ORC ORC NP ORC NP $\bar{S}$, met wcn)
12	Cancel	—	(NP ORC NP $\bar{S}$, met wcn)
13	LC-Predict	ORC → NP V	($\bar{V}$ ORC ORC NP $\bar{S}$, met wcn)
14	Match	V → met	(ORC ORC NP $\bar{S}$, wcn)
15	Cancel	—	(NP $\bar{S}$, wcn)
16	LC-Predict	S → NP VP	($\bar{VP}$ S $\bar{S}$, wcn)
17	Shift	V → wcn	($\bar{V}$ $\bar{VP}$ S $\bar{S}$, ϵ)
18	LC-Predict	VP → V	(VP $\bar{VP}$ S $\bar{S}$, ϵ)
19	Cancel	—	(S $\bar{S}$, ϵ)
20	Cancel	—	(ϵ, ϵ)

The largest memory load required for 3C is larger than the one required for 3B.

3 Garden paths and reanalysis (8 points)

Sentences like the one in (4.) tend to produce a "garden path" effect[3] (at least temporarily):

4. While John watched the baby that won cried

On the first pass through this sequence of words, people tend to interpret 'the baby' as the beginning of an NP that is the object of 'watched'.[4]

For the following questions, use the grammar given at the beginning of this assignment.

A. Draw the complete correct parse tree for (4.). Also draw a "partial tree" that shows the structure that the human processor seems to initially assign to the words up to and including 'won'. — 1 point

B. Show the full sequence of configurations that a bottom-up parser goes through to correctly parse the sentence in (4.). (Including the labels of the transitions and grammar rules used is optional here.) Identify the point in this correct sequence of configurations where humans (assuming for the moment that they use a bottom-up parsing system) initially head off in the wrong direction through the search space, i.e. towards the incomplete "partial tree" from above. Show at least the first three "wrong" configurations that it reaches when it goes in this direction. Then, do the same thing for a top-down parser. — 5 points

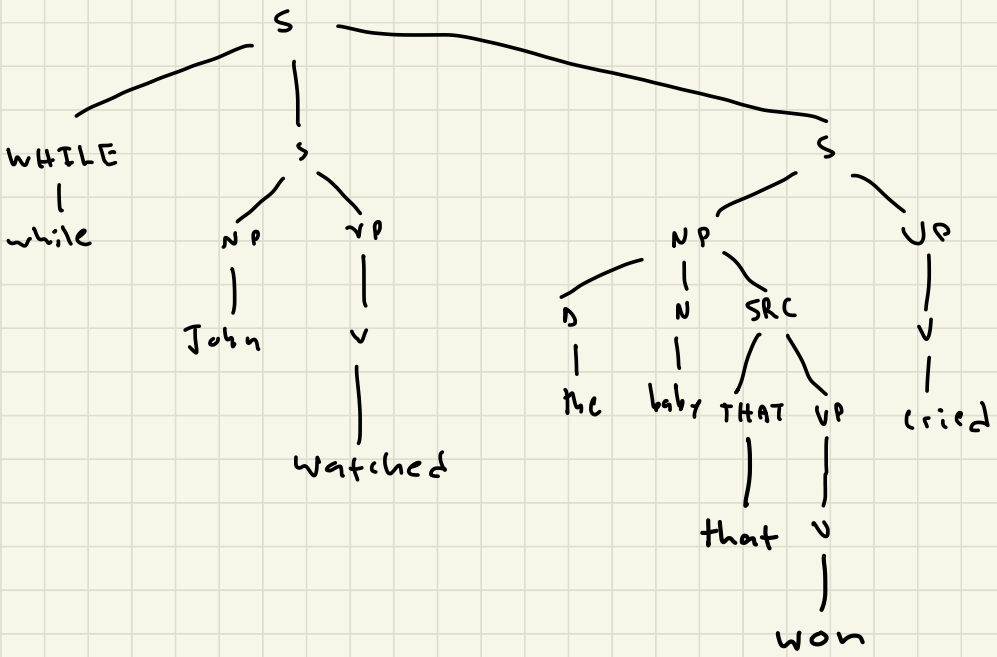
C. Let's suppose that when the human parsing system "backtracks" (i.e. reaches a dead end configuration and realizes that it took a wrong turn, and has to go back a few steps to start off down a different path), this works simply by looking back (right-to-left) through the sentence in an obvious way (i.e. going back to the point in the sentence where it took a wrong turn, and trying again); and suppose that this backtracking can be detected in experiments where we track the (leftward and rightward) movements of people's eyes as they read. Explain how we could use (4.) in such an experiment to decide whether humans use a bottom-up parser or a top-down parser. (Assume for this question that these are the only two possibilities.) — 2 points

[3] https://en.wikipedia.org/wiki/Garden-path_sentence

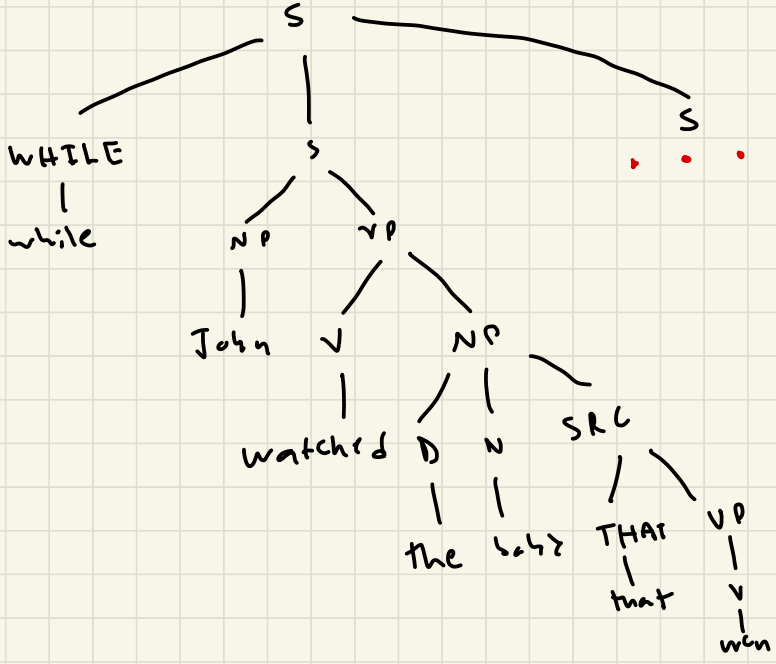
[4] More specifically, this is an example of what's known as the "Late Closure" parsing preference: the structure people go to first is one where the VP constituent is "closed off" later in the sentence than the first possibility, which is straight after 'watched'. If you don't get the effect, try something like Although John kept reading the story about the baby made him sad.

Correct Parse tree

4A.



"partial Tree"



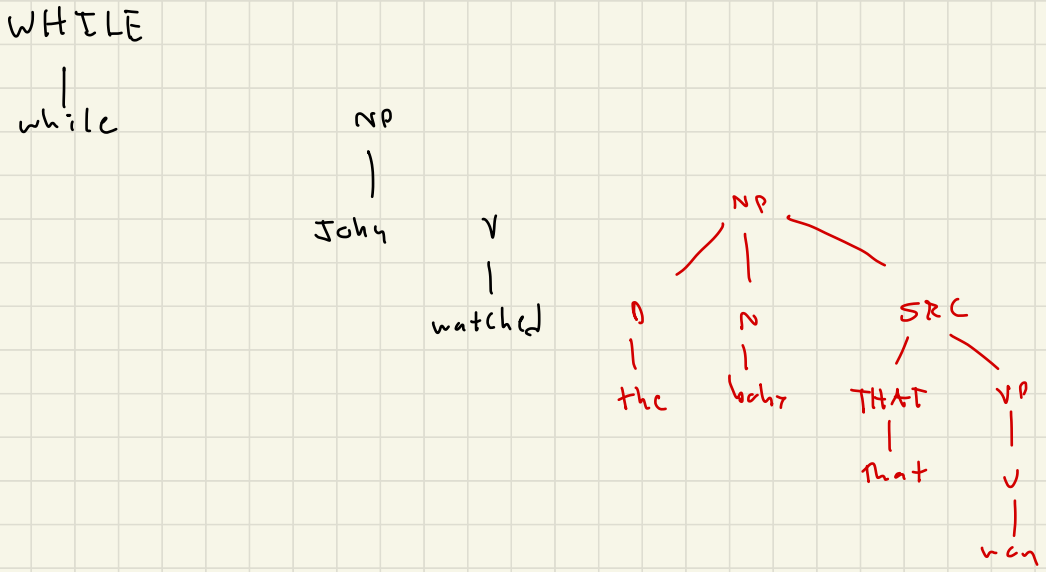
4B. bottom-up parser

While John watched the baby that won cried

	Type of transition	Rule used	Configuration
0	—	—	(ε, While John watched the baby that won cried)
1	Shift	WHILE → while	(WHILE, John watched the baby that won cried)
2	shift	NP → John	(WHILE NP, watched the baby that won cried)
3	Shift	V → watched	(WHILE NP V, the baby that won cried)
4	Reduce	VP → V	(WHILE NP VP, the baby that won cried)
5	Reduce	S → NP VP	(WHILE S, the baby that won cried)
6	Shift	D → the	(WHILE S D, baby that won cried)
7	shift	N → baby	(WHILE S D N, that won cried)
8	Shift	THAT → that	(WHILE S D N THAT, won cried)
9	Shift	V → won	(WHILE S D N THAT V, cried)
10	Shift	V → cried	(WHILE S D N THAT V V, ε)
11	Reduce	VP → V	(WHILE S D N THAT V VP, ε)
12	Reduce	VP → V	(WHILE S D N THAT VP VP, ε)
13	Reduce	SRC → THAT VP	(WHILE S D N SRC VP, ε)
14	Reduce	NP → D N SRC	(WHILE S NP VP, ε)
15	Reduce	S → NP VP	(WHILE S S, ε)
16	Reduce	S → WHILE S S	(S, ε)

* After step 3, humans head off in wrong direction ↓

4 shift	D → the	(WHILE NP V D, baby that was cried)
5 shift	N → baby	(WHILE NP V D N, that was cried)
6 shift	THAT → that	(WHILE NP V D N THAT, was cried)
7 shift	V → was	(WHILE NP V D N THAT V, cried)
8 Reduce	VP → V	(WHILE NP V D N THAT VP, cried)
9 Reduce	SRC → THAT VP	(WHILE NP V D N SRC, cried)
10 Reduce	NP → D N SRC	(WHILE NP V NP, cried)
...		



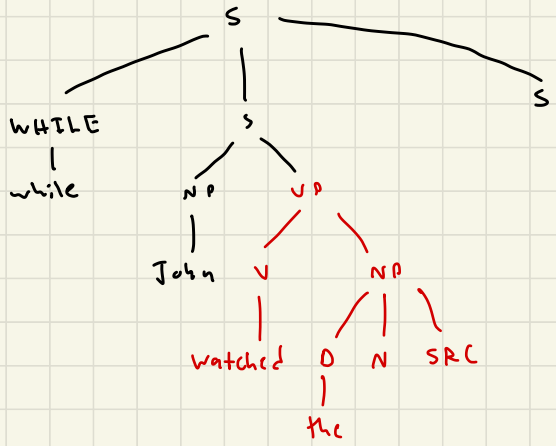
4B. Top-down parser

While John watched the baby that won cried

	Type of transition	Rule used	Configuration
0	—	—	(S, While John watched the baby that won cried)
1	Predict	S → WHILE S S	(WHILE S S, while John watched the baby that won cried)
2	Match	WHILE → while	(S S, John watched the baby that won cried)
3	Predict	S → NP VP	(NP VP S, John watched the baby that won cried)
4	Match	NP → John	(VP S, watched the baby that won cried)
5	Predict	VP → V	(V S, watched the baby that won cried)
6	Match	V → watched	(S, the baby that won cried)
7	Predict	S → NP VP	(NP VP, the baby that won cried)
8	Predict	NP → D N SRC	(D N SRC VP, the baby that won cried)
9	Match	D → the	(N SRC VP, baby that won cried)
10	Match	N → baby	(SRC VP, that won cried)
11	Predict	SRC → THAT VP	(THAT VP VP, that won cried)
12	Match	THAT → that	(VP VP, won cried)
13	Predict	VP → V	(V VP, won cried)
14	Match	V → won	(VP, cried)
15	Predict	VP → V	(V, cried
16	Match	V → cried	(ε, ε)

* After step 4, humans head off in wrong direction ↓

5	Predict	VP → V NP	(V NP S, watched the baby that won cried)
6	match	V → watched	(NP S, the baby that won cried)
7	Predict	NP → D N SRC	(D N SRC S, the baby that won cried)
8	match	D → the	(N SRC S, baby that won cried)
	...		



4c.

We can tell humans use a bottom-up parser if their eyes move from right-to-left after reading “cried”. This happens because they attached the elements up to “cried” too early and don't notice this until they have read the whole sentence.

A top-down parser will make an early prediction of the structure. Humans use a top-down parser if they backtrack when reaching “that won” as their initial predicted structure is wrong. Contrary to a bottom-up parser, the human will realize “the baby that won” can't be the object of “watched” causing them to backtrack earlier than the bottom-up parser.