CMPT 413 - SPRING 2008 - FINAL EXAM

Please write down "FINAL" on the top of the answer booklet.

There are seven questions in this final, each with 10 points. Choose any six to answer.

List the six question numbers to be graded on the 1st page of the question booklet.

When you have finished, return your answer booklet along with this question booklet.

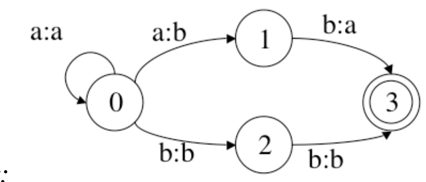
Apr 18, 2008

(1) **Finite-state transducers**:

a. Provide a finite state transducer for each of the following regular (relation) expressions:

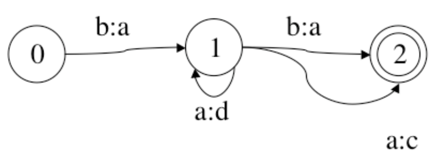
i. T_1: `( (a:0 | a:1) (b:0 | b:1) )*`

ii. T_2: `(a:a)* ( (a:b b:a) | (b:b b:b) )`



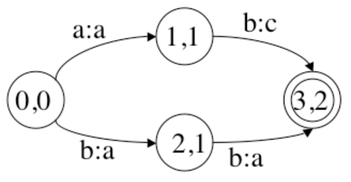
Answer:

iii. T_3: `b:a (a:d)* (b:a | a:c)`



Answer:

b. Provide the finite state transducer $T_4 = T_2 \circ T_3$, the result of composing T_2 with T_3.



Answer:

c. Let $L(f)$ be the language generated by a finite-state machine f and let $R(t)$ be the regular relation generated by a finite-state transducer t. A **transduction** for a given input string i and a given finite-state transducer t is the set of all output strings o such that $(i, o) \in R(t)$. Provide a precise algorithm for implementing the **transduction** operation choosing from the following operations as components of your algorithm. You are given as input: string s and FST t.

- The operation *makefsm*(s) produces a finite-state machine that recognizes only string s.
- The composition operation $\circ$, for two transducers $t_1 \circ t_2$, it returns the composition of the two transducers.
- For finite-state machine f, $Id(f)$ is the transducer such that for each $w \in L(f), (w, w) \in Id(f)$.
- For a finite-state transducer t, $\pi_1(t)$ is the finite-state machine that corresponds to the projection of the input language of t.
- For a finite-state transducer t, $\pi_2(t)$ is the finite-state machine that corresponds to the projection of the output language of t.

- The operation *getstring*(f) returns the set of strings encoded by a finite-state machine f (this set could be infinite).

Answer: Let t be the FST and s be the input string. The set of output strings o such that $(s, o) \in R(t)$ is provided by:

$$getstring\,(\pi_2\,(Id(makefsm(s)) \circ t))$$

d. Consider the input alphabet $\Sigma_1 = \{a, b\}$ and the output alphabet $\Sigma_2 = \{0, 1\}$.

i. Write down a *single-state* finite state transducer that does all insertions, deletions and substitutions for any pair of strings in $\langle \Sigma_1^*, \Sigma_2^* \rangle$. Align with ϵ for insertions/deletions.

ii. Assume that the cost for computing the minimum edit distance is the number of transitions taken in the FST: each transition whether it is insertion, deletion or substitution has cost 1. For the input *ab* provide all the alignments with minimum cost.

Answer:

$s_1 \rightarrow s_1$ transition on:

- ϵ:0
- ϵ:1
- a:ϵ
- b:ϵ
- a:0
- b:1
- a:1
- b:0

min cost alignments for *ab* will be:

$$\begin{array}{cccc} ab & ab & ab & ab \\ 00 & 01 & 10 & 11 \end{array}$$

Note that following kinds of alignments will be more expensive:

$$\begin{array}{cccc} a & \epsilon & b & \epsilon \\ \epsilon & 0 & \epsilon & 1 \end{array}$$

(2) **Language Modeling**:
TrueCasing is the process of taking text with missing or unreliable case information, e.g. if the input looks like:

```
as previously reported , target letters were issued last month to
michael milken , drexel 's chief of junk-bond operations ; mr. milken 's
brother lowell ; cary maultasch , a drexel trader ; james dahl , a
drexel bond salesman ; and bruce newberg , a former drexel trader .
```

Then the output of the TrueCasing program should be:

```
As previously reported , target letters were issued last month to
Michael Milken , Drexel 's chief of junk-bond operations ; Mr. Milken 's
brother Lowell ; Cary Maultasch , a Drexel trader ; James Dahl , a
Drexel bond salesman ; and Bruce Newberg , a former Drexel trader .
```

Assume we are given a *translation probability* $P(w \mid W)$ where w is the lowercase variant of the TrueCase word W (note that the TrueCase word might still be lowercase). A language model $P(W_1, \ldots, W_n)$ is used to provide the probability of a sentence. A *bigram language model* approximates the probability of a sentence as follows:

$$P(W_1, \ldots, W_n) = \prod_{i=1}^{n} P(W_i \mid W_{i-1})$$

We assume that $W_{-1} = w_{-1} = none$ is a dummy word that begins each sentence.

a. Describe how you can train the translation probability $P(w \mid W)$. Assume you have access to a sufficient amount of TrueCase text.

Answer: For each TrueCase word W convert it to lowercase w and count $f(w, W)$ and $f(W)$. Then,

$$P(w \mid W) = \frac{f(w, W)}{f(W)}$$

b. Complete the following formula to provide a model of the TrueCasing task by using only the translation probability and the bigram language model:

$$\begin{aligned} W_1^*, \ldots, W_n^* &= \operatorname*{argmax}_{W_1,\ldots,W_n} P(W_1, \ldots, W_n \mid w_1, \ldots, w_n) \\ &= \textit{provide this formula} \end{aligned}$$

Answer:

$$\begin{aligned} W_1^*, \ldots, W_n^* &= \operatorname*{argmax}_{W_1,\ldots,W_n} P(W_1, \ldots, W_n \mid w_1, \ldots, w_n) \\ &= \frac{P(W_1, \ldots, W_n) \cdot P(w_1, \ldots, w_n \mid W_1, \ldots, W_n)}{P(w_1, \ldots, w_n)} \\ &\approx P(W_1, \ldots, W_n) \cdot P(w_1, \ldots, w_n \mid W_1, \ldots, W_n) \\ &= \prod_{i=1}^{n} \underbrace{P(w_i \mid W_i)}_{\textit{translation probability}} \cdot \underbrace{P(W_i \mid W_{i-1})}_{\textit{bigram language model}} \end{aligned}$$

c. Why was the mapping between TrueCase words and their lowercase variant defined using $P(w \mid W)$ instead of $P(W \mid w)$?

Answer: To use the noisy channel model, this has to be the way to model this task. There can be other ways to directly find the most appropriate mapping which could use $P(W \mid w)$.

d. Provide the pseudo code or recurrence relation that will find the score for $W_1^*, \ldots, W_n^*$, which is the most likely TrueCase output according to our model. Provide the time complexity of the algorithm.

Answer: We can use the Viterbi algorithm. For input $w_1, \ldots, w_n$ we can efficiently compute the best score upto TrueCase word candidate W_{i+1} recursively as follows:

$$\text{Viterbi}[i+1, W_{i+1}] = \max_{W_i} \left(\text{Viterbi}[i, W_i] \times P(w_{i+1} \mid W_{i+1}) \times P(W_{i+1} \mid W_i)\right)$$

We initialize the recursion with the value for Viterbi$[1, W_1]$.

If the number of lowercase words is $|w|$ and the number of TrueCase words is $|W|$ then the time complexity is $O(|w| \cdot |W|^2)$.

e. It is possible that due to some mappings between lowercase and TrueCase words, that the value of the bigram probability could be zero if the particular bigram W_{i-1}, W_i was not observed in training data. If you were to pick a smoothing method for smoothing n-grams, from the following options which one would you pick and why?

Option 1 Add-one, Add-delta, Simple Good-Turing

Option 2 Katz backoff, Jelinek-Mercer deleted interpolation

Do not focus on complexity of implementation or previous performance of these smoothing methods in implementations. Choose based on which option will always lead to lower perplexity and provide a reason.

(3) **Hidden Markov Models**:

The probability model $P(t_i \mid t_{i-2}, t_{i-1})$ is provided below where each t_i is a part of speech tag, e.g. $P(\mathrm{D} \mid \mathrm{N}, \mathrm{V}) = \frac{1}{3}$. Also provided is $P(w_i \mid t_i)$ that a word w_i has a part of speech tag t_i, e.g. $P(\text{flies} \mid \mathrm{V}) = \frac{1}{2}$.

$P(t_i \mid t_{i-2}, t_{i-1})$	t_{i-2}	t_{i-1}	t_i
1	bos	bos	N
$\frac{1}{2}$	bos	N	N
$\frac{1}{2}$	bos	N	V
$\frac{1}{2}$	N	N	V
$\frac{1}{2}$	N	N	P
$\frac{1}{3}$	N	V	D
$\frac{1}{3}$	N	V	V
$\frac{1}{3}$	N	V	P
1	V	D	N
1	V	V	D
1	N	P	D
1	V	P	D
1	P	D	N
1	D	N	eos

$P(w_i \mid t_i)$	t_i	w_i
1	D	an
$\frac{2}{5}$	N	time
$\frac{2}{5}$	N	arrow
$\frac{1}{5}$	N	flies
1	P	like
$\frac{1}{2}$	V	like
$\frac{1}{2}$	V	flies
1	eos	eos
1	bos	bos

The part of speech tag definitions are: `bos` (*begin sentence marker*), N (*noun*), V (*verb*), D (*determiner*), P (*preposition*), `eos` (*end of sentence marker*).

a. Consider a Jelinek-Mercer style interpolation smoothing scheme for $P(w_i \mid t_i)$:

$$P_{jm}(w_i \mid t_i) = \Lambda[t_i] \cdot P(w_i \mid t_i) + (1 - \Lambda[t_i]) \cdot P(w_i)$$

Λ is an array with a value $\Lambda[t_i]$ for each part of speech tag t_i, such that $0 \leq \Lambda[t_i] \leq 1$. Provide a condition on Λ that must be satisfied to ensure that P_{jm} is a well-defined probability model.

Answer: Because of the following fact about $P(w_i \mid t_i)$:

$$\sum_{w_i} P(w_i \mid t_i) = 1$$

and in P_{jm} we are given t_i, so to interpolate with $P(w_i)$ the following condition has to hold:

$$\sum_{t_i} \Lambda[t_i] = 1$$

b. Provide a Hidden Markov Model (*hmm*) that uses the trigram part of speech probability $P(t_i \mid t_{i-2}, t_{i-1})$ as the transition probability $P_{hmm}(s_j \mid s_k)$ and the probability $P(w_i \mid t_i)$ as the emission probability $P_{hmm}(w_j \mid s_j)$.

Important: Provide the *hmm* in the form of two tables as shown below. The first table contains transitions between states in the *hmm* and the transition probabilities and the second table contains the words emitted at each state and the emission probabilities. Do not provide entries with zero probability.

from-state s_k	to-state s_j	$P(s_j \mid s_k)$

state s_j	emission w	$P(w \mid s_j)$

Hint: In your *hmm* the state $\langle N, \texttt{eos}\rangle$ will have emission of word `eos` with probability 1 and will not have transitions to any other states.

Answer: Here are the two tables that define the HMM, the transition table on the left and the emission table on the right:

from-state s_k	to-state s_j	$P(s_j \mid s_k)$	
bos, bos	bos, N	$P(N \mid bos, bos)$	1
bos, N	N, N	$P(N \mid bos, N)$	$\frac{1}{2}$
bos, N	N, V	$P(V \mid bos, N)$	$\frac{1}{2}$
N, N	N, V	$P(V \mid N, N)$	$\frac{1}{2}$
N, N	N, P	$P(P \mid N, N)$	$\frac{1}{2}$
N, V	V, D	$P(D \mid N, V)$	$\frac{1}{3}$
N, V	V, V	$P(V \mid N, V)$	$\frac{1}{3}$
N, V	V, P	$P(P \mid N, V)$	$\frac{1}{3}$
V, D	D, N	$P(N \mid V, D)$	1
V, V	V, D	$P(D \mid V, V)$	1
N, P	P, D	$P(D \mid N, P)$	1
V, P	P, D	$P(D \mid V, P)$	1
P, D	D, N	$P(N \mid P, D)$	1
D, N	N, eos	$P(eos \mid D, N)$	1

state s_j	emission w	$P(w \mid s_j)$
bos, bos	bos	1
bos, N	time	$\frac{2}{5}$
bos, N	arrow	$\frac{2}{5}$
bos, N	flies	$\frac{1}{5}$
N, N	time	$\frac{2}{5}$
N, N	arrow	$\frac{2}{5}$
N, N	flies	$\frac{1}{5}$
N, V	like	$\frac{1}{2}$
N, V	flies	$\frac{1}{2}$
V, D	an	1
V, V	like	$\frac{1}{2}$
V, V	flies	$\frac{1}{2}$
N, P	like	1
V, P	like	1
P, D	an	1
D, N	time	$\frac{2}{5}$
D, N	arrow	$\frac{2}{5}$
D, N	flies	$\frac{1}{5}$

c. Based on your *hmm* constructed in 3b. what is the state sequence that would be provided by the Viterbi algorithm for the following input sentence:

`bos bos` time flies like an arrow `eos`

Answer:

bos	time	flies	like	an	arrow	eos	
(bos,bos) 1	(bos,N) $\times 1 \times \frac{2}{5}$	(N,V) $\times\frac{1}{2} \times \frac{1}{2}$	(V,P) $\times\frac{1}{3} \times 1$	(P,D) $\times 1 \times 1$	(D,N) $\times 1 \times \frac{2}{5}$	(N,eos) $\times 1 \times 1$	$= \frac{1}{75}*$
(bos,bos) 1	(bos,N) $\times 1 \times \frac{2}{5}$	(N,V) $\times\frac{1}{2} \times \frac{1}{2}$	(V,V) $\times\frac{1}{3} \times \frac{1}{2}$	(V,D) $\times 1 \times 1$	(D,N) $\times 1 \times \frac{2}{5}$	(N,eos) $\times 1 \times 1$	$= \frac{1}{150}$
(bos,bos) 1	(bos,N) $\times 1 \times \frac{2}{5}$	(N,N) $\times\frac{1}{2} \times \frac{1}{5}$	(N,P) $\times\frac{1}{2} \times 1$	(P,D) $\times 1 \times 1$	(D,N) $\times 1 \times \frac{2}{5}$	(N,eos) $\times 1 \times 1$	$= \frac{1}{125}$
(bos,bos) 1	(bos,N) $\times 1 \times \frac{2}{5}$	(N,N) $\times\frac{1}{2} \times \frac{1}{5}$	(N,V) $\times\frac{1}{2} \times \frac{1}{2}$	(V,D) $\times\frac{1}{3} \times 1$	(D,N) $\times 1 \times \frac{2}{5}$	(N,eos) $\times 1 \times 1$	$= \frac{1}{750}$

(4) **Context-free Grammars**:

a. Consider the following context-free grammar:

$$NP \rightarrow NP\ NP$$
$$NP \rightarrow natural \mid language \mid processing \mid course$$

i. Provide all the distinct parse trees that the above grammar gives for the input string: *natural language processing course*.

Answer:
```
(NP (NP (NP natural) (NP language)) (NP (NP processing) (NP course)))
(NP (NP natural) (NP (NP language) (NP (NP processing) (NP course))))
(NP (NP (NP (NP natural) (NP language)) (NP processing)) (NP course))
(NP (NP natural) (NP (NP (NP language) (NP processing)) (NP course)))
(NP (NP (NP natural) (NP (NP language) (NP processing))) (NP course))
```

ii. From the various parse trees you've listed above, provide the tree that corresponds to the natural meaning of the phrase: a course that teaches the processing of natural language.

Answer:
```
(NP (NP (NP (NP natural) (NP language)) (NP processing)) (NP course))
```

iii. Show how you can predict the number of parse trees for the above input string using the notion of Catalan numbers.

Answer: Assume there is a hidden and between each *NP*, *NP* → *NP* and *NP*, and so we can transform the input string to *natural* and *language* and *processing* and *course* and just as in the coordination grammar covered in the lecture notes, the number of parse trees is given by *Cat(number of* and*s)* = *Cat*(3) = 5.

iv. True or false: The above grammar is in Chomsky Normal Form.

Answer: True!

b. Consider a Treebank where the following set of trees are repeated several times as indicated:

- 2× (S (B a a) (C a a))
- 1× (S (C a a a))
- 7× (S (B a))

i. What is the set of CFG rules that can be extracted from this Treebank.

ii. Based on the frequency of the trees shown above, compute the probability of each CFG rule.

Answer:

S -> B C	2/10	B -> aa	2/9
S -> C	1/10	B -> a	7/9
S -> B	7/10	C -> aa	2/3
		C -> aaa	1/3

iii. Given this probabilistic CFG what is the most likely tree for the input: *aaaa*

Answer: (S (B a) (C aaa)) is the most likely tree for input aaaa

iv. Does the most likely tree for input *aaaa* appear in the Treebank? If not, why not?

Answer: The subtrees for *B* and *C* are chosen independently due to the independence assumptions made by PCFGs, so the most likely tree contains the most likely *B* subtree and the most likely *C* subtree despite that fact that the most likely *B* subtree may have never co-occured with the most likely *C* subtree in the Treebank.

(5) **Parsing**:

a. Consider the CFG below with S as the start symbol:

$$\begin{array}{rcl} S & \rightarrow & NP\ VP \\ VP & \rightarrow & V\ NP \\ VP & \rightarrow & VP\ PP \\ NP & \rightarrow & NP\ PP \\ NP & \rightarrow & Det\ N \\ PP & \rightarrow & P\ NP \end{array} \qquad \begin{array}{rcl} NP & \rightarrow & Calvin \\ V & \rightarrow & saw \\ Det & \rightarrow & the \\ N & \rightarrow & man \mid telescope \mid hill \\ P & \rightarrow & with \mid on \end{array}$$

Consider the input sentence: *Calvin saw the man on the hill with a telescope*.

i. In an Earley parser, what states can we predict from the state ($S \rightarrow \bullet\ NP\ VP, [0,0]$).

Answer:
($NP \rightarrow \bullet\ NP\ PP, [0,0]$)
($NP \rightarrow \bullet\ Calvin, [0,0]$)
($NP \rightarrow \bullet\ Det\ N, [0,0]$)
($Det \rightarrow \bullet\ the, [0,0]$)

ii. Subsequently, what states can we complete from the state ($NP \rightarrow Calvin\ \bullet, [0,1]$).

Answer: The span [0,1] has to be correct!
($S \rightarrow NP\ \bullet\ VP, [0,1]$)
($NP \rightarrow NP\ \bullet\ PP, [0,1]$)

b. Can the CKY algorithm be used with the above grammar. Why?

Answer: Yes. G is in CNF.

c. Consider a sub-class of CFGs that is only allowed to have rules of the *type* $A \rightarrow a\ B\ b$ or $A \rightarrow a$, where A, B are any non-terminals, and a, b are any terminals. For example, the following CFG for *ambiguous palindromes* is in this sub-class:

$$\begin{array}{rcl} S & \rightarrow & aXa \mid bXb \mid aYa \mid cYc \\ X & \rightarrow & aXa \mid bXb \mid a \mid b \\ Y & \rightarrow & aYa \mid cYc \mid a \mid c \end{array}$$

By analogy to the analysis used for the CKY algorithm, show that this class of CFGs can be parsed in time $O(n^2)$.

Hint: Derive a normal form for the above sub-class of CFGs (analogous to Chomsky Normal Form and the relation of CNF to the CKY algorithm).

(6) **Feature unification**:

a. Define the operation of unification over feature structures (denoted as $\sqcup$) in terms of subsumption (denoted by $\sqsubseteq$).

Answer: $D = D' \sqcup D''$ is defined as the most general feature structure D such that $D' \sqsubseteq D$ and $D'' \sqsubseteq D$.

b. True or false: $(A \sqcup (B \sqcup C)) = ((A \sqcup B) \sqcup C)$

Answer: True, unification is order-independent.

c. Consider the following feature structure written in the feature path notation:

```
<cat>=VP
<subject agreement person>=3
<subject agreement number>=sg
<agreement>=<subject agreement>
```

Provide the feature structure equivalent to the above.

Answer: $\begin{bmatrix} \text{cat: VP} \\ \text{agreement: } \boxed{1} \\ \text{subject: } \begin{bmatrix} \text{agreement: } \boxed{1} \begin{bmatrix} \text{person: 3} \\ \text{number: sg} \end{bmatrix} \end{bmatrix} \end{bmatrix}$

d. Provide the directed acyclic graph for the following feature struture:

$$\begin{bmatrix} \text{a: } \boxed{1} \begin{bmatrix} \text{b: sg} \\ \text{c: pl} \end{bmatrix} \\ \text{d: } \begin{bmatrix} \text{e: } \boxed{1} \end{bmatrix} \end{bmatrix}$$

e. Provide the output of unifying the following two feature structures:

1. $\begin{bmatrix} \text{a: } \boxed{1} \begin{bmatrix} \text{b: } \boxed{2} \\ \text{c: } \boxed{2} \end{bmatrix} \\ \text{d: } \begin{bmatrix} \text{e: } \boxed{1} \end{bmatrix} \end{bmatrix}$

2. $\begin{bmatrix} \text{d: } \begin{bmatrix} \text{e: } \begin{bmatrix} \text{b: sg} \\ \text{c: pl} \end{bmatrix} \end{bmatrix} \end{bmatrix}$

Answer: Unification fails.

f. Consider the following feature-based context-free grammar

$$\begin{array}{lcl}
S\begin{bmatrix}\text{type: inverted}\end{bmatrix} & \rightarrow & \mathit{AUX}\ \mathit{NP}\ \mathit{VP}\begin{bmatrix}\text{tense: none}\end{bmatrix} \\
\mathit{AUX}\begin{bmatrix}\text{tense: pres} \\ \text{num: pl}\end{bmatrix} & \rightarrow & \mathit{do} \\
\mathit{AUX}\begin{bmatrix}\text{tense: pres} \\ \text{num: sg}\end{bmatrix} & \rightarrow & \mathit{does} \\
\mathit{AUX}\begin{bmatrix}\text{tense: past} \\ \text{num: } \boxed{1}\end{bmatrix} & \rightarrow & \mathit{did}
\end{array}$$

$$\begin{array}{rcl}
\mathit{NP} & \rightarrow & \mathit{DET}\ \mathit{N} \mid \mathit{children} \\
\mathit{DET} & \rightarrow & \mathit{the} \\
\mathit{N} & \rightarrow & \mathit{dog} \mid \mathit{dogs} \mid \mathit{children} \\
\mathit{VP}\begin{bmatrix}\text{tense: none}\end{bmatrix} & \rightarrow & \mathit{Vinf}\ \mathit{NP} \\
\mathit{VP}\begin{bmatrix}\text{tense: } \boxed{1}\end{bmatrix} & \rightarrow & \mathit{Vtns}\begin{bmatrix}\text{tense: } \boxed{1}\end{bmatrix} \mathit{NP} \\
\mathit{Vinf} & \rightarrow & \mathit{like} \\
\mathit{Vtns}\begin{bmatrix}\text{tense: pres}\end{bmatrix} & \rightarrow & \mathit{likes} \\
\mathit{Vtns}\begin{bmatrix}\text{tense: past}\end{bmatrix} & \rightarrow & \mathit{liked}
\end{array}$$

Add new feature structures to the above grammar so that it can correctly handle the following cases:

```
does the dog like children
do the dogs like children
*do the dog like children # should not get any parses
*do the dogs likes children # should not get any parses
did the dog like children
did the dogs like children
```

The sentences marked with $*$ should not get any parses.

(7) **Discourse**:

Centering theory defines coherence between successive pair based on transitions between sentences in a text. Transition states are ordered by preference: Continue > Retain > Smooth-Shift > Rough-Shift, where each transition state is defined as follows:

	$C_b(U_{n+1}) = C_b(U_n)$ or undefined $C_b(U_n)$	$C_b(U_{n+1}) \neq C_b(U_n)$
$C_p(U_{n+1}) = C_b(U_{n+1})$	Continue	Smooth-Shift
$C_p(U_{n+1}) \neq C_b(U_{n+1})$	Retain	Rough-Shift

$C_f(U_n)$ is the set of forward-looking centers ordered using standard syntactic roles in utterance U_n. C_p is the preferred center from this set and $C_b(U_{n+1})$ is the backward-looking center of the following utterance U_{n+1}. $C_b(U_{n+1})$ is the most highly ranked element from $C_f(U_n)$ mentioned in U_{n+1} and is undefined if no such element exists. $C_b(U_1)$ is undefined.

a. No pronouns can be an element of C_f, only the entities they refer to can be elements. Provide a constraint on referents for pronouns using $C_f(U_n)$ and $C_b(U_{n+1})$ that will enable the use of Centering theory for pronoun coreference disambiguation.

Answer: If any element of $C_f(U_n)$ is realized by a pronoun in U_{n+1} then $C_b(U_{n+1})$ must also be realized by a pronoun.

b. Given this theory of preferential ordering of transition states, which of the two sequences of utterances below is more coherent. Provide all the steps required to show that one discourse is more coherent than the other. You can assume that the noun phrase: *a student* is coreferent with the entity *John* so that in the following utterance sequences *He* always refers to *John*.

1. U_1: John is a student. U_2: He took the bus. U_3: It was crowded. U_4: He was trying to read.
2. U_1: John is a student. U_2: He took the bus. U_3: He was trying to read. U_4: It was crowded.

Answer:

Common utterances to both sequences:

- U_1: John is a student

$$\begin{array}{ll} C_f(U_1): & \{\text{John}\} \\ C_p(U_1): & \text{John} \\ C_b(U_1): & \text{undefined} \end{array}$$

- U_2: He took the bus

$$\begin{array}{ll} C_f(U_2): & \{\text{John, bus}\} \\ C_p(U_2): & \text{John} \\ C_b(U_2): & \text{John} \end{array}$$

- U_1 to U_2 = Continue.

For Sequence 1:

- U_3: It was crowded.

$$\begin{array}{ll} C_f(U_3): & \{\text{bus}\} \\ C_p(U_3): & \text{bus} \\ C_b(U_3): & \text{bus} \end{array}$$

- U_2 to U_3 = Smooth-shift
- U_4: He was trying to read.

$$\begin{array}{ll} C_f(U_4): & \{\text{John}\} \\ C_p(U_4): & \text{John} \\ C_b(U_4): & \text{undefined} \end{array}$$

- U_3 to U_4 = Retain

For Sequence 2:

- U_3: He was trying to read.

$$\begin{array}{ll} C_f(U_3): & \{\text{John}\} \\ C_p(U_3): & \text{John} \\ C_b(U_3): & \text{John} \end{array}$$

- U_2 to U_3 = Continue
- U_4: It was crowded.

$$\begin{array}{ll} C_f(U_4): & \{\text{bus}\} \\ C_p(U_4): & \text{bus} \\ C_b(U_4): & \text{undefined} \end{array}$$

- U_3 to U_4 = Retain