Algorithms for AI and NLP (Fall 2008)
— Final Exam —

General Instructions

- Please read through the complete exam once before starting to answer questions. About thirty minutes into the exam, the instructor will come around to answer questions of clarification.
- Please follow the instructions closely. Most of the questions ask for short answers. When a maximum length is given (e.g. 'in no more than two sentences'), please try to stick to those limits.
- As discussed in class, the exam is given in English, but you are free to answer in any of *Bokmål*, English, or *Nynorsk*.

1 Finite-State Technology and General Search (100 Points)

(a) Draw a finite-state automaton (FSA) that recognises the language $\texttt{a}^n$, where n is a number greater or equal to zero and divisible by three or four. Thus, the automaton should recognize strings like `aaa`, `aaaa`, `aaaaaa`, `aaaaaaaa`, etc.

(b) Recall the distinction between deterministic and non-deterministic FSAs. What does it mean for an FSA to be non-deterministic? Is your solution to part (a) above deterministic or not? In a few sentences, sketch the procedure for converting a non-deterministic FSA into a deterministic one. In the general case, what is the maximum number of states in the new, deterministic automaton, assuming the original non-deterministic FSA had n states?

(c) In the light of our discussion of FSAs, though not necessarily limiting yourself to the finite-state universe, what is a common characteristic of computations that we describe as *search problems*? Recall the notions of memoization and dynamic programming: give an example of a problem that benefits from dynamic programming. In no more than two sentences, what is the general idea in memoization, and which types of computation lend themselves especially well to dynamic programming?

2 Hidden Markov Models (120 Points)

Assume the following part-of-speech tagged training 'corpus' of just one sentence:

still	,	time	s	move	is	being	received	well	,	once	again	.
RB	,	NNP	POS	NN	VBZ	VBG	VBN	RB	,	RB	RB	.

(a) In a few sentences, discuss the concept of smoothing and explain why it is important. Next, ignoring smoothing and making the standard simplifying assumptions for a naïve bi-gram HMM (including the assumption that the training corpus provides the full inventory of distinct tags and complete vocabulary), calculate the following:

(i) For each tag t, the probability of t following the tag RB, i.e. $P(t|\text{RB})$

(ii) The emission probabilities $P(move|\text{NNP})$, $P(move|\text{NN})$, and $P(well|\text{RB})$.

(b) Assume further that for each tag t, $P(t|\texttt{<s>}) = P(t)$; in one sentence, what does it mean to make this assumption. Construct part of the Viterbi trellis for tagging the utterance *once again , time .* Rather than calculating all values, indicate the total size of the trellis and the computations for filling in the first two columns.

(c) In a few sentences, summarize the key points of the Viterbi algorithm. What is the interpretation of each cell in the trellis? What is the complexity of the algorithm, i.e. the number of computations performed in relation to (i) the length of the input sequence and (b) the size of the tag set? Very briefly, sketch an alternative, naïve method for computing the most probable tag sequence t_1^n, given an input string w_1^n; state how the Viterbi algorithm improves over this approach.

3 Context-Free Grammars and Parsing (130 Points)

Consider the language defined by the following grammar:

S → VP NP	NP → *kim*
VP → PP VP	NP → *oslo*
NP → PP NP	NP → *snow*
VP → NP V	V → *adores*
PP → NP P	P → *in*

(a) For each of the following items, identify the number of readings (distinct analyses) that the grammar of Mirror English assigns, and draw the parse trees for each of the readings.

(i) *oslo in snow adores kim.*

(ii) *kim adores snow in oslo.*

(iii) *snow adores in oslo kim.*

(b) If possible, provide one example each of a sentence of Mirror English with exactly (i) three and (ii) four readings.

(c) Identify which of the following parsing strategies, if any, will run into difficulties with the grammar of Mirror English, and briefly explain why: (i) top-down parsing or (ii) bottom-up parsing. Is the grammar of Mirror English suitable for use with the CKY parser? If so, why? If not, why not?

(d) In a few sentences, discuss the concept of *local ambiguity* that is at the core of the CKY and generalized chart parsers. Do any of the examples (i) to (iii) from part (a) above contain local ambiguity? If so, where exactly, and what would happen in chart parsing.

(e) Recall very briefly the role of the chart in the CKY and generalized chart parsers. What types of information are associated with each edge in the chart? How are active edges different from passive ones (feel free to use the 'dating' metaphor, if you find it useful), and what is the general form of the *fundamental rule* in chart parsing?

4 Probabilistic Context-Free Grammar (100 Points)

(a) When adapting CFGs to Probabilistic Context-Free Grammars, we made one extension to the elements of the original CFG and added an additional condition on the elements of the set P of productions. Sketch the formalisation of a context-free grammar (as an n-tuple) that we used in the class, and summarize the revisions we made in extending CFGs to PCFGs.

(b) Following is a probabilistic variant of the grammar of Mirror English:

S → VP NP [1.0]	NP → *kim* [0.2]
VP → PP VP [0.2]	NP → *oslo* [0.2]
NP → PP NP [0.2]	NP → *snow* [0.4]
VP → NP V [0-9]	V → *adores* [0.2]
PP → NP P [1.0]	P → *in* [0.2]

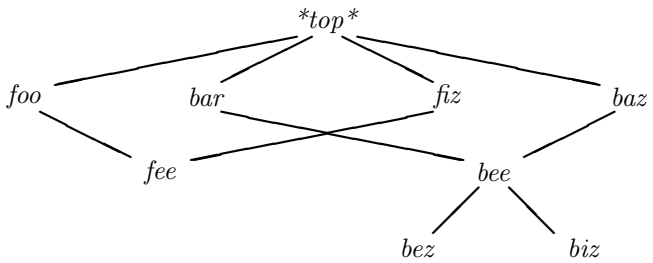
Assuming 'S' as the start symbol and sets of non-terminal and terminal symbols as implicitly given by the rules above, is this grammar a valid PCFG? If not, why not?

(c) Whether or not this grammar is a valid PCFG, determine the probability of the sentence *oslo in snow adores kim* according to the rules above.

5 Feature Structures and Unifiction (100 Points)

(a) What is a good basic measure to quantify the cost of feature structure manipulation, i.e. the amount of effort required to copy a feature structure, or unify two structures? Recall the unification and copy procedures that we implemented in the course; for each of the two, is it destructive, non-destructive, or quasi-destructive in nature?

(b) Assume the following type hierarchy:



(c) Draw the following feature structures as DAGs, i.e. as graphs of labeled nodes and labeled, directed arcs:

$$foo\begin{bmatrix} \texttt{A}\ \boxed{1}_{bar}\begin{bmatrix}\texttt{C}\ bee\end{bmatrix} \\ \texttt{B}\ \boxed{1}\end{bmatrix} \qquad\qquad fee\begin{bmatrix} \texttt{A}\ _{bar}\begin{bmatrix}\texttt{C}\ baz\end{bmatrix} \\ \texttt{B}\ _{baz}\begin{bmatrix}\texttt{D}\ biz\end{bmatrix}\end{bmatrix}$$

In no more than two sentences, comment on the correspondences between elements of the feature structure and elements of the DAG.

(d) Graphically, show the result of destructively unifying the two DAGs. Are the two structure compatible, i.e. does unification actually succeed? How does our unifier implement the notion of equivalence classes of nodes?

6 Common-Lisp (150 Points)

(a) How many elements are contained in the list returned by the following expression? What will happen when we use the function `length()` to count them?

```
(let ((foo (list 42)))
  (setf (rest foo) foo))
```

Describe the effects of the function `?()` below. Discuss at least one calling example and show the 'box notation' used to keep track of `cons()` cells.

(b) At various points in the class we talked about data structures to index and efficiently retrieve information, for example the so-called transition and emission matrices in the context of HMMs. Recall that, for the emission matrix, we typically looked up probabilities for a pair of the current state (encoded as an integer) and the current word (a string). Given this scenario, briefly discuss the relative strengths and weaknesses of lists, arrays, and hash tables for associative retrieval, i.e. the look-up of values associated with keys that (conceptually) pair an integer with a string. Reflect on the number of distinct values (in a typical HMM, say one used for PoS tagging) along both dimensions, and further take into consideration whether you expect the emission matrix to be densely or sparsely populated, i.e. what proportion of combinations (state plus word) out of the set of possible combinations will typically be used.

(c) Write a two-place function `ditch()` that takes an atom as its first and a list as its second argument; `ditch()` removes *all* occurrences of the atom in the list, e.g.

```
? (ditch 'c '(a b c d e c))
→ (A B D E)
? (ditch 'f '(a b c d e c))
→ (A B C D E C)
```

Note that we are not primarily concerned with specific details of Lisp syntax here. If you find that easier, feel free to use elements of 'pseudo code' in your function definition, as long as it is clear how exactly everything will work. Give a brief informal summary of the basic principles (e.g. in terms of base case(s) vs. recursive cases) and general approach of your implementation.