Algorithms for AI and NLP (Fall 2010)
— 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 any questions of clarification (including English terminology).
- 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 only given in English, but you are free to answer in any of *Bokmål*, English, or *Nynorsk*.

1 Language Models and HMMs (20 + 30 + 50 Points)

(a) How does an n-gram language model estimate the probability $P(s)$ for a string $s = w_1^n$, assuming a first-order n-gram model? Discuss the central assumption made in this modelling approach; what is the common name of said assumption? Name at least one practical problem that can be solved (to a certain degree) using language models.

(b) How does a Hidden Markov Model (HMM) compare to a language model? In a few sentences, sketch the differences between the two approaches and also comment on commonalities, if any. Name at least one practical problem that can be solved (to a certain degree) using HMMs. Is it possible to use a Hidden Markov Model to estimate the probability $P(s)$ for a string $s = w_1^n$; if so, how is this usage of HMMs different from the task of part of speech tagging?

(c) In the course, we introduced the Viterbi algorithm as our first example of dynamic programming. In a few sentences, discuss key characteristics of problems that benefit from dynamic programming; give at least one example. How exactly does the Viterbi algorithm achieve dynamic programming? Is the core assumption made in question (a) above related to the application of dynamic programming to HMMs? Although we did not discuss this extension in detail, think about what would happen when moving from first- to second-order HMMs, i.e. the use of tri-grams rather than bi-grams. Can you sketch, in rough terms, how the trellis and Viterbi algorithm would need to be adjusted?

2 Context-Free Grammars and Parsing (20 + 20 + 30 + 30 Points)

Consider the language defined by the following grammar (assuming the conventional start symbol 'S'):

S $\rightarrow$ NP VP	NP $\rightarrow$ *oslo*
NP $\rightarrow$ PP NP PP	NP $\rightarrow$ *dogs*
VP $\rightarrow$ VP PP	V $\rightarrow$ *barked*
VP $\rightarrow$ V	P $\rightarrow$ *in*
PP $\rightarrow$ P NP	P $\rightarrow$ *near*

(a) Draw the parse tree for each reading assigned by the grammar of Symmetric English to the following sentence, or explain why it has no readings:

(i) *Near in oslo dogs near oslo dogs in oslo barked.*

For each sentence of length from two to six words, give an example sentence in Symmetric English, if any grammatical sentences exist.

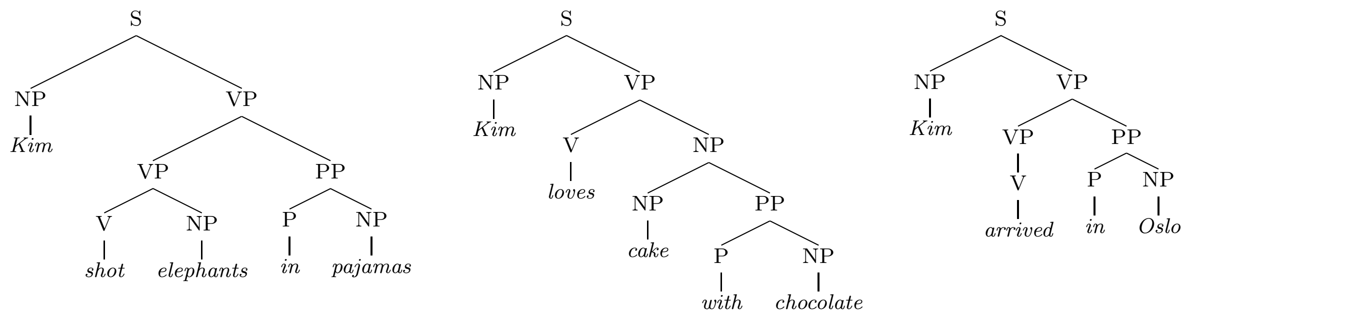
(b) Discuss the properties of this grammar that either support or prevent the use of a CKY parser for parsing sentences of Symmetric English. The CKY parser is limited to grammars in so-called Chomsky Normal Form (CNF); what does this limitation entail for the maximum number of elements in the right-hand sides of grammar rules, and how could this restriction be linguistically inadequate in analyzing sentences like *Kim gave Sandy the book.*

(c) In a few sentences, characterize what we mean by *local ambiguity* in parsing. Why is the run-time complexity of the parsing problem for natural languages (e.g. for standard English) exponential in the worst case? How do the CKY parser and our generalized chart parser work around this computationally rather undesirable complexity?

(d) Recall how we generalized the CKY parser to support arbitrary context-free grammars; what is the function of active edges, and how do they help to work around the CNF limitations? What is the so-called fundamental rule in chart parsing? In generalized chart parsing, there are no restrictions on the order of computation, i.e. edges can be added to the chart in any possible sequence. In a few sentences, explain how the algorithm can maintain correctness and completeness (i.e. find all valid solutions) no matter the order of computation.

3 Probabilistic CFGs and ParsEval (20 + 30 + 30 + 20 Points)

For the following questions, assume a mini-treebank of three annotated sentences:



(a) In a few sentences, sketch the procedure for extracting a context-free grammar from a treebank and show the formula for the estimation of rule probabilities. Given our limited training data, how many rules in the corresponding grammar will have VP as their left-hand side? Calculate the probabilities for these rules.

(b) In your own words, discuss whether the sentence *Kim loves elephants in Oslo.* is ambiguous; for each syntactic reading, provide a short paraphrase (i.e. free-form explanation) of the corresponding situation. In case there is ambiguity, which reading will be assigned the highest probability by the PCFG derived from our mini-treebank? Will a PCFG always be able to fully disambiguate, or could there be ties among different readings, in principle?

(c) Thinking about prepositional phrases (PPs) in our grammar, how many non-terminal categories can be combined with a PP as their sister? How do the probabilities of rules involving PPs (on their right-hand side) compare to each other, i.e. does our treebank provide a bias for attaching PPs to some category over others? If so, explain which property of the treebank is responsible for this effect.

(d) In a few sentences, recall the basic principles of the ParsEval metric to quantify the similarity of two parse trees; what are the elementary units used in the comparison, and how is the metric defined quantitatively? Ignoring the differences in terminal symbols, what would be the ParsEval score(s) for comparing the first and second trees of our mini-treebank, assuming the first tree was the gold standard, and the second tree was the top-ranked hypothesis of a statistical parser.

4 Common-Lisp (20 + 20 + 10 + 50 Points)

(a) What do we mean by the term *binding* when talking about Lisp symbols? Is it possible for a symbol to have multiple bindings simultaneously? Name at least two Lisp constructs that establish new value bindings (excluding `defparameter()`, `setf()`, and `setq()`).

(b) For each of the following lists, make the underlying structure explicit by either showing the list in 'box notation' or (ii) providing a Common-Lisp s-expression to construct the list, using exclusively the `cons()` function, the atoms that are elements of the list, and `nil`.

   (i) `(1 2 3)`

   (ii) `((1) (2 3))`

(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 (top-level) 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 found 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.

(d) Develop a revision of `ditch()` that does not generate new `cons()` cells, i.e. avoids all calls to functions like `cons()`, `append()`, or `list()`. In proposing your revised implementation, give a brief informal summary of the basic principles (e.g. in terms of base case(s) vs. recursive cases) and general approach.