Computational Linguistics (Spring 2008)
— Sample Exam —

1 Finite-State Technology

(a) Draw the finite-state automaton (FSA) that corresponds to the regular expression `(a|b*)+`.

(b) For each of the following strings, say whether it is part of the language recognized by this automation or not: (i) `aaa`, (ii) `bab`, (iii) `abab`, (iv) `aabaab`.

(c) Recall the equivalence relation holding between non-deterministic and deterministic FSAs. What is the characteristic of a non-deterministic automaton, and is your solution to part (a) above deterministic or not?

(d) Assume an electronic corpus of running text. In a few sentences, summarize the tasks of tokenization and sentence segmentation. What is the relation between these two tasks, if any? Give at least two examples of common challenges (i.e. potential sources of errors) encountered in these tasks. Assuming a finite-state approach, sketch regular expressions that could be employed in identifying token and sentence boundaries.

2 Mirror English

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

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

(a) For each of the following items, identify the number of readings (distinct analyses) that the grammar of Mirror English assigns:

(i) *in oslo snow adores kim*

(ii) *kim adores snow in oslo*

(iii) *snow adores in oslo kim*

(b) Draw the constituent tree for each of the readings.

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

3 Unification-Based Grammar

(a) Show the feature structure representation of the following rule as it is used in our LKB grammars:

$$\textit{head-initial}\begin{bmatrix}\texttt{HEAD} & \boxed{1}\\ \texttt{SPR} & \boxed{2}\\ \texttt{COMPS} & \boxed{3}\end{bmatrix} \longrightarrow \textit{expression}\begin{bmatrix}\texttt{HEAD} & \boxed{1}\\ \texttt{SPR} & \boxed{2}\\ \texttt{COMPS} & \begin{bmatrix}\texttt{FIRST} & \boxed{4}\\ \texttt{REST} & \boxed{3}\end{bmatrix}\end{bmatrix}, \quad \boxed{4}\textit{phrase}\begin{bmatrix}\ \end{bmatrix}$$

(b) What is the (approximate) name of the above rule in our grammar? Sketch its functionality in a few sentences and provide two examples of types of phrases built using this rule.

(c) Why did we choose to implement rules as single feature structures?

4 Typed Feature Structures

(a) Draw the following feature structure in DAG notation, i.e. as a directed acyclic graph of labeled nodes and directed arcs:

$$\textit{phrase}\left[\begin{array}{ll}\texttt{HEAD}\ \boxed{1}\ \textit{verb} & \\ \texttt{ARGS} & \textit{*ne-list*}\left[\begin{array}{ll}\texttt{FIRST} & \textit{word}\left[\begin{array}{l}\texttt{ORTH}\ \textit{“chased”}\\ \texttt{HEAD}\ \boxed{1}\end{array}\right]\\ \texttt{REST} & \textit{*ne-list*}\left[\begin{array}{ll}\texttt{FIRST} & \textit{expression}\left[\texttt{HEAD}\ \textit{noun}\right]\\ \texttt{REST} & \textit{*null*}\end{array}\right]\end{array}\right]\end{array}\right]$$

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

5 Linguistic Concepts

(a) Name at least two syntactic properties that characterize (syntactic) heads in the formation of phrases. Use one or two examples.

(b) Sketch a constituent tree for the sentence *the fierce dog chased the cat near the aardvark*. On each node, provide information about its general category (using abbreviatory notions like ‘Det’, ‘N’, ‘NP’, ‘VP’, et al.) and for each branch of the tree indicate whether the constituent dominated by it acts as a *head*, *specifier*, *complement*, or *modifier*.

(c) What is the difference between the function of the prepositional phrases (marked by square brackets) in the following two sentences:

(i) *that cat gave the aardvark [to the dogs]*
(ii) *the cat chased the aardvark [near the dogs]*

6 Chart Parsing

(a) Which aspect(s) of natural language (grammars) make(s) parsing (using context-free grammars, say) a surprisingly hard problem? How is a naïve parser affected in its performance, and what is the worst-case complexity relative to input string length? Mention at least one example.

(b) How does our generalized chart parser avoid this worst-case complexity, in general and for the example you gave above? Sketch, in a few sentences, the central idea in chart-based parsing.

7 Common-Lisp Games

(a) For each of the following lists, make the underlying structure explicit by (i) showing the list in ‘box notation’, and (ii) writing a Common-Lisp 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))`

(b) 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))
```