Marcus Kracht CompLing II, Spring 2007

Exercise Sheet 1

To be handed in on Thursday, April 19

Linear Context Free Rewrite Systems

Exercise 1.
Suppose that $L \subseteq A^*$ is a LCFRL and that $v : A \to B^*$ is a map that assigns a B-string to each letter of A. v is extended to A^* by putting $v(\varepsilon) := \varepsilon$, and $v(\vec{x}a) := v(\vec{x})v(a)$. Show that $h[L] := \{v(\vec{x}) : \vec{x} \in L\}$ is a LCFRL.

Exercise 2.
(Number names.) Assume that the largest primitive name for a number is `million`. How can we express numbers larger than a million? By iteration: we say `one million million` for `billion`, and `one million million million` for `trillion` and son on. Consider now the legal expressions for numbers in English. They are formed as follows. They are sequences

$$a_1\text{␣}(\texttt{million␣})^{i_1} a_2\text{␣}(\texttt{million␣})^{i_2} a_3\text{␣}(\texttt{million␣})^{i_3} \cdots \qquad (1)$$

where $i_1 > i_2 > i_3$ and so on, and the a_i are expressions for numbers less than a million (for example `seventeen␣thousand␣three␣hundred␣and␣sixty␣five`). Write a program that recognises the legal expressions, scanning them from left to right, and which returns the number expressed by the number name, if it is legal, and some predefined exception otherwise. *Hint.* The input to this function is a string and the output is a number (= `int`).

Exercise 3.
Show that the language of number names satisfies the pumping lemma for context free languages. (But the language is not context free!) *For wizards.* It does not, however, satisfy Ogden's Lemma (which is a strengthening).

Exercise 4.
Let $\vec{x}$ be a word of length n.

- How many substrings does $\vec{x}$ have?
- How many k-tuples of substrings does $\vec{x}$ have? (For wizards: what happens if we require the substrings to be disjoint?)

Exercise 5.
Here is a simple proof to show that languages generated by a simple k-LMG G can be recognised in **PTIME**. Let $\vec{x}$ be given. Step 1. Get all k-tuples of substrings of $\vec{x}$ that are in the lexicon of G. Step 2. Compute the k-tuples of $\vec{x}$ that are constituents in increasing length. Compute how many steps this algorithm takes.

Exercise 6.
Let G be a simple k-LMG whose lexicon does not contain the empty word. Let $\vec{x}$ be a string of length n. Estimate the maximum number of steps needed to generate $\vec{x}$. Give an estimate of parsing complexity obtained by simply enumerating all derivations of suitable length and see if they yield $\vec{x}$.