Problem Set Six: Formal Languages

Carl Pollard
The Ohio State University
Linguistics 680

October 31, 2011

These problems are due Tue. Nov. 8.

Problem 1

Suppose A is finite and nonempty. Give an informal but persuasive argument that A^* is countably infinite. [Hints: (a) this means showing there is a bijection from ω to A^*. So what you are asked to do is to describe *informally* a way of listing all the A-strings without repetitions. It is a lot of work to make this technically precise, but you are not asked to do that. (b) Look at a (paper) phone directory. If you're uncertain what that is, ask your parents.]

Problem 2

Suppose A is finite and nonempty. Prove that the set of A-languages is nondenumerable. You can use any of the theorems or corollaries stated in Chapter Five or Problem Set Five, even if their proofs were not given. Also, you can take the countable infinitude of A^* to have been established in Problem 1.

Problem 3

Given an A-language L, use RT to formally define $\mathbf{kl}(L)$. More specifically, tell how X, x, and F in the statement of RT must be instantiated so that the function h whose existence is guaranteed by RT has the property that $\mathbf{kl}(L)$ is the union of all the languages $h(n)$, where $h(0)$ is x and for each $k \in \omega$, $h(k+1)$ is $F(h(k))$. [Hints: (a) The base clause in the informal definition should tell you what x has to be, and the recursion clause should

tell you what F has to be. (b) Another informal, but perhaps more helpful way to define $\mathbf{kl}(L)$ is as the union of all the languages $\mathbf{kl}_n(L)$, where

1. $\mathbf{kl}_0(L) = \mathbf{1}_A$; and
2. $\mathbf{kl}_{k+1}(L) = L \bullet \mathbf{kl}_k(L)$ for all $k \in \omega$.]

Problem 4

A **logic** consists of (1) a formal language whose members are called **formulas**; (2) a **proof theory**, i.e. a purely syntactic way of defining a relation of **provability** between collections of formulas and formulas (this will be made more precise in the weeks to come); and (3) a **semantics**, i.e. a way of assigning **meanings** to formulas that 'works in harmony' with the proof theory (this too will be made precise). Here we are concerned only with PL, the formal language associated with one particularly simple logic called **propositional logic**.

[*Note:* In propositional logic, the formulas can be thought of as (extremely) crude representations of natural language sentences; the proof theory as representing our ability to reason to conclusions from premisses based only on the forms of sentences (disregarding their meanings); and the meaning assigned by the semantics to a sentence as representing the proposition that the sentence expresses. However, in principle you don't need to know about any of this background to do the problem.]

Informally, the language PL is usually recursively defined in the following way. We start by assuming we have a set Let of things called 'propositional letters'. To keep things simple, let's suppose Let has three members, called X, Y, and Z. (Intuitively, the propositional letters are thought of as standing for arbitrary *atomic* natural language sentences, where by 'atomic' we mean sentences which do not themselves contain 'logic words' such as *and*, *or*, *implies*, and *not*, or their counterparts in non-English languages). Then the informal recursive definition goes like this:

1. (The length-one string corresponding to) each propositional letter is a formula;
2. (The length-one strings corresponding to) $\top$ and $\bot$ are formulas;
3. if P and Q are formulas, so are $(\sim P)$, $(P \wedge Q)$, $(P \vee Q)$, $(P \rightarrow Q)$, and $(P \leftrightarrow Q)$; and
4. nothing else is a formula.

It's important to be aware that what we are talking about here are formulas—which are strings of 'symbols' (propositional letters, parentheses,$\top$, $\bot$, $\sim$, $\wedge$, $\vee$, $\rightarrow$, and $\leftrightarrow$), and *not* the propositions (whatever *they* are) that the formulas express. For example, in the informal definition, when we write "$(\sim P)$" we mean the string obtained by stringing together the following symbols in the specified order: (1) the left-paren symbol (; (2) the negation symbol $\sim$; (3) the symbols in the string P; and finally (4) the right paren symbol). In short, PL is an A-language where A is a set with 12 distinct members X, Y, Z, $\sim$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$, $\top$, $\bot$, (, and). The members of A are called 'symbols', but it doesn't really matter what they are.

The problem is to define PL formally as an A-language, using RT. As you should realize by now, this means telling how X, x, and F in the statement of RT should be instantiated. (Careful: 'X' is now being used in two distinct ways!)

[Hints: (a) clauses (1) and (2) are the base clauses and clause (3) is the recursion clauses. (b) After you think you have the right definition, check to make sure it makes $((X \wedge Y) \wedge Z)$ a formula. If it doesn't, you'll need to revise your definition.]

Problem 5

In *autosegmental-metrical (AM)* theory, the phonological structure of the intonation of an English expression is represented by a string of **tones**.[1] The question of how such a string is realized as a "tune" (pitch contour) is a matter of phonetics, which doesn't concern us here.

Simplifying slightly, in one version of this theory, the set A of tones has ten members:

$$A = \{\mathbf{H}^*, \mathbf{L}^*, \mathbf{H}^* + \mathbf{L}, \mathbf{L}^* + \mathbf{H}, \mathbf{H} + \mathbf{L}^*, \mathbf{L} + \mathbf{H}^*, \mathbf{H}^-, \mathbf{L}^-, \mathbf{H}\%, \mathbf{L}\%\}$$

Note that each of these ten elements is to be thought of as a single member of the tonal "alphabet", no matter how many symbols it is written with. Of these, the first six are called **pitch accents** (PAs)[2], the next two are called

[1] The sense of the term 'tone' here is "minimal unit of intonational phonology". This is distinct from the notion of tone in the sense of pitch levels or contours within words that function as part of the word phonology and distinguish words from each other, e.g. Mandarin *ma* [high level] 'mother' vs. *ma* [high-falling-to-low] 'scold'.

[2] The asterisks in the notation for the pitch accents have nothing to do with the asterisk of formal language theory. Instead, they designate a tone (or part of a tone) that has to be associated with a syllable that is specified by the word containing it as capable of bearing stress.

intermediate-phrase final boundary tones (IFBTs); and the last two are called **(intonation phrase) final boundary tones** (FBTs).

Certain sets of tone strings are defined informally as follows:

1. an **intermediate phrase** consists of one or more PAs followed by an IFBT.
2. an **intonation phrase** consists of one or more intermediate phrases, followed by an FBT.
3. an **utterance** consists of one or more intonation phrases.

The problem is to *formally* define each of the following as regular languages A-languages:

1. the language **IP** of intermediate phrases;
2. the language **IN** of intonation phrases; and
3. the language **UT** of utterances.

You do not need to prove anything, you do not need to use RT, and you do not need to use PMI. All you need to do is express each of the these three languages in terms of (1) the ten tones; the underscore (that maps each tone to the singleton language whose only string is the langth-one string of that tone), the empty language 0_A, the singleton language 1_A whose only member is the null string ϵ; and the operations on languages of union ($\cup$), concatenation ($\bullet$), Kleene closure (**kl**), and positive Kleene closure (**kl**$^+$). Note: the fact that this is possible means that these languages are all regular. [Hint: You will not necessarily need to use all the available operations.]

Problem 6

Let T be a finite set, and recall that T-language L is said to be **context-free** if there exists a CFG $\langle T, N, D, P\rangle$ such that L is one of its syntactic categories, i.e. $L = C_A$ for some $A \in N$.

Prove that $L = \mathbf{Mir}(2)$ is context-free by presenting a CFG which has L as one of its syntactic categories.

Problem 7

Same problem as the preceding, but with $L = \mathrm{PL}$, the language of propositional logic with three propositional letters as in Problem 4.