Assignment 6

Regular Expressions from DFAs
Due. 11:59 p.m., Friday, 17 Oct. 2008

This is the last assignment before fall break. After the break, there is a **mid-term exam**
over the course material thus far. The exam is closed-book, except that you may use one
two-sided 8.5” x 11” sheet of notes that you have *prepared in advance* of the exam (i.e.
before the class at which the exams are distributed.) The exam may be taken at home, in
any **two-hour** contiguous period between end of class on Wed. 10/22 and start of class on
Mon. 10/27. **Contiguous** means that once you start the exam, you must finish it without
interrupting the process. The exam conduct is governed by the HMC Honor Code.

**Problem:**

Construct a Scheme function `dfa2re` that has one argument: a Moore machine using the S
expression representation for machines from the previous assignment. By convention, a
state is accepting iff it has an output of 1, and the first state listed is the initial state of the
machine.

The output of `dfa2re` is regular expression in a prescribed S expression representation:

<table>
<thead>
<tr>
<th>Representation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>lambda</code></td>
<td>The set consisting of just the <em>empty string</em></td>
</tr>
<tr>
<td><code>empty</code></td>
<td>The <em>empty set</em></td>
</tr>
<tr>
<td><code>any single letter</code></td>
<td>The set consisting of one string of just that letter.</td>
</tr>
<tr>
<td><code>( + R S . . . )</code></td>
<td>The set that is the <strong>union</strong> of the sets represented by the regular expressions <code>R, S, ...</code></td>
</tr>
<tr>
<td><code>( ^ R S . . . )</code></td>
<td>The set consisting of <strong>concatenation</strong> of strings, the first from the set represented by <code>R</code>, the second from the set represented by <code>S</code>, ...</td>
</tr>
<tr>
<td><code>(* R)</code></td>
<td>The set consisting of any number (including 0) of concatenations of elements of the set represented by <code>R</code></td>
</tr>
</tbody>
</table>

You may restrict the `+` and `^` operators to be binary if you wish, or allow them to be of
*arbitrary* arity.

You may produce the result in simplified form for extra credit if you wish, but it is not
required. For example, `( + lambda (* 0))` can be simplified to `(* 0)`.

Examples will be provided in the test file on the website.