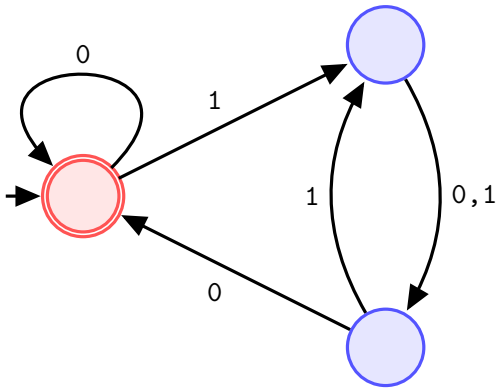
Basic

Recall the following language definitions from the lecture:

1. Acceptance problems
   1) $A_{\text{DFA}} = \{\langle B, w\rangle \mid B \text{ is a DFA that accepts input string } w\}$
   2) $A_{\text{NFA}} = \{\langle B, w\rangle \mid B \text{ is an NFA that accepts input string } w\}$
   3) $A_{\text{CFG}} = \{\langle G, w\rangle \mid G \text{ is a CFG that generates string } w\}$
   4) $A_{\text{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ accepts } w\}$
2. Language emptiness problems
   1) $E_{\text{DFA}} = \{\langle A\rangle \mid A \text{ is a DFA and } L(A) = \emptyset\}$
   2) $E_{\text{CFG}} = \{\langle G\rangle \mid G \text{ is a CFG and } L(G) = \emptyset\}$
   3) $E_{\text{TM}} = \{\langle M\rangle \mid M \text{ is a TM and } L(M) = \emptyset\}$
3. Language equality problems
   1) $EQ_{\text{DFA}} = \{\langle A, B\rangle \mid A \text{ and } B \text{ are DFAs and } L(A) = L(B)\}$
   2) $EQ_{\text{CFG}} = \{\langle G, H\rangle \mid G \text{ and } H \text{ are CFGs and } L(G) = L(H)\}$
   3) $EQ_{\text{TM}} = \{\langle M_1, M_2\rangle \mid M_1 \text{ and } M_2 \text{ are TMs and } L(M_1) = L(M_2)\}$

(1) This exercise is meant to help you get used to the *object encoding* notation.

Answer all parts for the following DFA M and give reasons for your answers.



   1) Is $\langle M, 0100\rangle \in A_{\text{DFA}}$?
   2) Is $\langle M, 011\rangle \in A_{\text{DFA}}$?
   3) Is $\langle M\rangle \in A_{\text{DFA}}$?
   4) Is $\langle M\rangle \in E_{\text{DFA}}$?
   5) Is $\langle M, M\rangle \in EQ_{\text{DFA}}$?

(2) Draw a Venn diagram to illustrate the relationship between the languages: *regular, context-free, decidable*, and *Turing-recognizable*; then indicate where the following problems belong to:

$$A_{\text{DFA}}, E_{\text{DFA}}, EQ_{\text{DFA}}, \quad A_{\text{NFA}}$$
$$A_{\text{CFG}}, E_{\text{CFG}}, EQ_{\text{CFG}}$$
$$A_{\text{TM}}, E_{\text{TM}}, EQ_{\text{TM}}$$

(3) Read about the *Post Correspondence Problem (PCP)* on Wikipedia: http://en.wikipedia.org/wiki/Post_correspondence_problem.

- Is it decidable?
- How about when the alphabet is simply {a}?
- Find some (easy) examples and try to solve them by hand, e.g.

$$\left\{ \left[\frac{\texttt{ab}}{\texttt{abab}}\right], \left[\frac{\texttt{b}}{\texttt{a}}\right], \left[\frac{\texttt{aba}}{\texttt{b}}\right], \left[\frac{\texttt{aa}}{\texttt{a}}\right] \right\}$$

- Try to write code to search for solutions using exhaustive search.

(4) Let $AMBIGCFG = \{\langle G \rangle \mid G \text{ is an ambiguous CFG}\}$.

Show that $AMBIGCFG$ is undecidable.

Hint: Use a reduction from PCP (above) as follows:

Given an instance

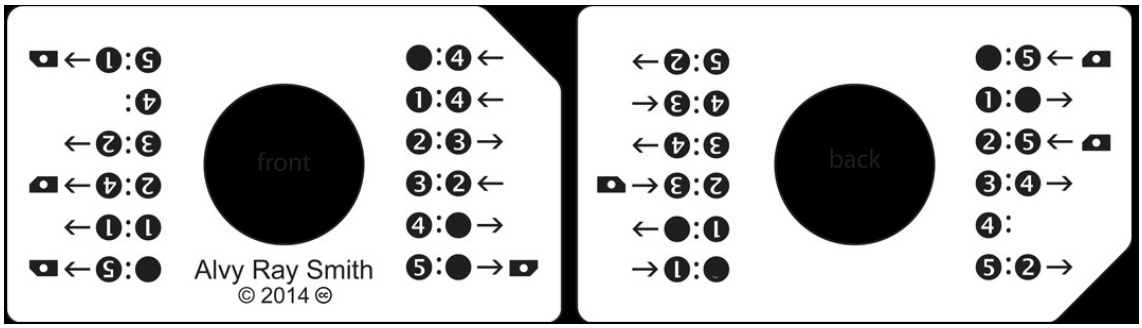
$$\left\{ \left[\frac{t_1}{b_1}\right], \left[\frac{t_2}{b_2}\right], \cdots, \left[\frac{t_k}{b_k}\right] \right\}$$

of the PCP, construct a CFG G with the rules:

$$\begin{array}{lcl} S & \to & T \mid B \\ T & \to & t_1 T a_1 \mid \cdots \mid t_k T a_k \mid \varepsilon \\ B & \to & b_1 B a_1 \mid \cdots \mid b_k B a_k \mid \varepsilon \end{array}$$

where $a_1, a_2, \ldots, a_k$ are new terminal symbols.

(1) Here is a **Universal Turing Machines** (UTM) as business card TM!



This TM is... **universal** with only 4 states and 6 symbols!

This TM can simulate any other TM!

→ a **computer** is a simple idea – it is programming it that is hard...

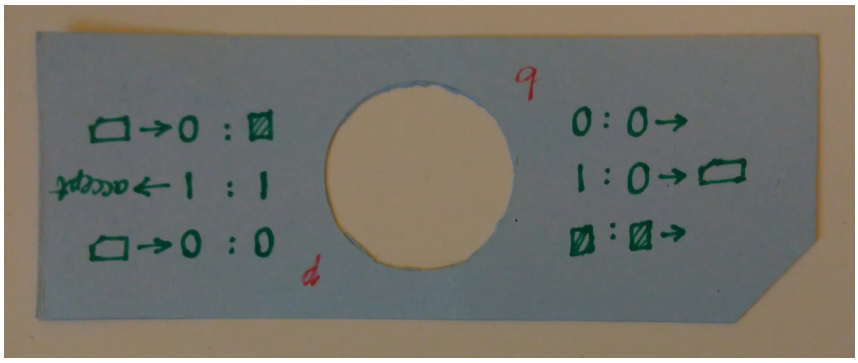
Explanation of how to use this machine can be found at: http://alvyray.com/CreativeCommons/TuringToys.htm

(2) **Business card TMs**

Recall the TM from last week over $\Sigma = \{0, 1\}$ with $Q = \{q, p, q_{\text{accept}}, q_{\text{reject}}\}$, $q_{\text{start}} = q$, $\Gamma = \{0, 1, \square\}$, and δ:

State	Tape symbol	Transition
q	0	$(q, 0, R)$
q	1	$(p, 0, R)$
q	$\square$	$(q, \square, R)$
p	0	$(q, 0, L)$
p	1	$(q_{\text{accept}}, 1, R)$
p	$\square$	$(q, 0, L)$

Use cardboard to make a "business card" representation of it as follows:



To use it, you would place it on the memory tape with the current cell visible through the circular opening and the current state showing on the right, etc. as in the previous question.

(3) "A *quine* is a computer program which takes no input and produces a copy of its own source code as its only output. The standard terms for these programs in the computability theory and computer science literature are *self-replicating programs*, *self-reproducing programs*, and *self-copying programs*." http://en.wikipedia.org/wiki/Quine_(computing)

Write a quine in your preferred programming language.

This is related to the concept of *re____ion*. (Find the missing letters.)