ROSETTA PAPERS 2: CATEGORIES AND AUTOMATA

D. E. STEVENSON

Abstract.

1. Automata

1.1. **Free Automata.** Let Σ be a finite alphabet. A **Σ-automaton $\mathcal{A}$** is a category with with a finite set of objects called *states* and denoted Q. The arrows can be thought of as triples (p, σ, q) in $Q \times \Sigma \times Q$ or as a relation $\sigma : Q \to Q$. In either case, σ is called a *label*. We will frequently employ the graphic $p \xrightarrow{\sigma} q$ to indicate these relations. The only restriction on a Σ-automaton is the finiteness of Q.

There are a wide variety of texts available that explore automata from the classical viewpoint. Some of these texts are **give a list**.

A **path** in a Σ-automaton is a finite sequence

$$c = (q_0, \sigma_1, q_1)(q_1, \sigma_2, q_2) \cdots (q_{k-1}, \sigma_k, q_k)$$

An alternative notation would be

$$q_0 \xrightarrow{\sigma_1} q_1 \xrightarrow{\sigma_2} q_2$$

Let the element $s = \sigma_1 \cdots \sigma_k \in \Sigma^*$. s is called a *label* of the path c and is denoted by $|c|$. Since $|s|$ is already used for the length of strings, use $||c||$ to be the length of the path.

We would like to make automata be monoids. To do so, we need to have the unit path, denoted 1_q, and defined for every $q \in Q$ as (q, Λ, q), where Λ is the unit for Σ^*. So the unit for the path is $|\Lambda|$ and $||1_q||$ is zero.

1.2. **Defining Recognition.** The intent is that automata serve as recognition devices for the character strings associated with the path. So consider a simple automaton that recognizes exactly one, finite string s. How should we define the automaton?

Since the automaton recognizes only one string and it is finite, it must have the form $s = \sigma_1\sigma_2 \cdots \sigma_k$. Therefore, we could guess that a path in the automaton would look like

$$c = (x_0, \sigma_1, x_1)(x_1, \sigma_2, x_2) \cdots (x_{k-1}, \sigma_k, x_k),$$

where the x_i's are unknown. One way to solve the problem is to inspect the arrows (it is a finite set) and attempt to find a sequence of arrows to construct c. In other words, c is defined not only by the label but by the sequence of states $x_0x_1 \cdots x_k$.

Aside on usual definition of an automaton. This leads to the usual definition of Σ-automaton as having five components (Σ, Q, I, F, R) where I is a set containing the initial states (possible x_0s), F containing the final states (possible x_ks) and R is the allowed moves. We have accounted for all but I and F.

We call a path c *successful* if there exists a path starting in a state $x_0 \in I$ and terminating in state $x_k \in F$ such that each transition between states is such that the label on the transition matches the corresponding character in the string. □

The equivalent categorical definition would be in terms of compositions of arrows. It is clear that concatentation of paths is composition of arrows. Let FP_x be the set of *free paths starting at* x defined to be the set of all paths that have the form $(x_0, \sigma_1, x_1) \cdots$. let FS_{σ_1} be the set freely generated strings that have their initial character as σ_1 The correspondance we are looking for is that for the automaton to recognize the string $s \in FS_{\sigma_1}$ there must be a path in FP_{x_0} such that the label of the path equals s.

The labels on all the successful paths on an automaton $\mathtt{A}$ is said to be the **language of the automaton** $\mathtt{A}$ or the **behavior of** $\mathtt{A}$**. In another abuse of notation, we define $|\mathtt{A}|$ to be the behavior of** $\mathtt{A}$**.**

The class of recognizable subsets of Σ^* is denoted $Rec\Sigma$.

1.3. **Discourse on Effective Recognizabilty. A subset A of Σ^* is said to be recognizable if there exists a Σ-automaton $\mathtt{A}$ such that $|\mathtt{A}| = A$.**

This brings up the issue of how to understand the definition.

- **If the statement is "a recognizable subset A of Σ^* is given" it will be understood that the automaton has been effectively supplied. That is, we have in had a definition of $\mathtt{A}$ such that $|\mathtt{A}| = A$.**
- **If the set A is supplied some other way, then we are required to prove that it is recognizable by constructing an automaton that does recognize A. The demonstration must use the data concerning A to be used to develop the automaton. *Effective* is a notion....**

2. Examples

Example. The null automaton $\emptyset$ has the null language; i. e., $|\emptyset| = \emptyset$. It is clear that this could be done by having the set of states empty. What about the minimal arrow approach? The minimal arrow set would have to have at least $|1_q|$ for each $q \in Q$ and would therefore accept the unit string Λ. Since ther are no states, there are no initial or terminal states.

Example. By the above, the automaton $\mathtt{\Lambda}$ with one state q and only the identity morphism 1_q accepts the language $\{\Lambda\}$. The one state is both initial and terminal.

Added during rosetta3. There has to be a $\mathtt{\Lambda}$ for every situation. I. e., $\mathtt{A} \times \mathtt{\Lambda}$ the states must match up. If necessary, have $\mathtt{\Lambda_Q}$ to indicate that the state set is Q but the identity arrows are the only arrows.

Example. Now we want to generate Σ^*. We need one state q and the arrows have the form (q, σ_i, q) for each $\sigma_i \in \Sigma$. The one state is both initial and terminal.

Example. For a *given* n and Σ, there is an automaton that recognizes $\{a^n b^n\}$. How would we synthesize such a machine? The final machine must have one path

$$c = (q_0, a, q_1)(q_1, a, q_2) \cdots (q_{n-1}, a, q_n)(q_n, b, q_{n+1}) \cdots (q_{2n-1}, b, q_{2n})$$

There can't be less than $2n + 1$ states because to produce $2n$ characters you need to pass through $2n$ transitions and this means $2n + 1$ states in the path. There could be more than $2n + 1$ states, but the extra transitions must be unit transitions, otherwise more than $2n$ characters are needed. The initial state is q_0 and terminal state is q_{2n}.

Example. The language $\{a^n b^n \| n \geq 0\}$ is not recognizable by any automaton.

Now, you need to show how the evaluation goes with the limits, etc.

442 R. C. Edwards Hall Department of Computer Science, Clemson University, PO Box 341906, Clemson, SC 29634-1906

***E-mail address*: `steve@cs.clemson.edu`**