A Language's Canonical Automaton

Wim H. Hesselink, April 17, 2009

The theorem of Myhill-Nerode asserts that a formal langage L is regular if and only if its right-equivalence relation $\sim_L$ is of finite index. The theorem appeared in [RS59], and was based on unpublished work of Myhill, and [Ner58]. The theorem is closely related to the construction of the minimal automaton for a regular language. It has survived in modern text books as, e.g., [Sud06, Section 6.7].

Yet, there is a more elegant approach. Show that every language has a (not necessarily finite) minimal automaton. Because a language is defined to be regular iff it is the language of some finite automaton, it follows immediately that a language is regular if and only if its minimal automaton is finite. The theorem of Myhill-Nerode then follows from the relationship of this minimal automaton with the right-equivalence relation $\sim_L$.

After I had gained these insights, I searched the library, and found that they are contained in [Eil74, Section 3.5]. The aim of this note is thus to revive these insights and save them from oblivion.

1 The language of an automaton

Recall that an *alphabet* is a finite set (of symbols). If Σ is an alphabet, a *string* over Σ is a finite sequence of elements of Σ. The empty string is denoted by λ. The set of all string over Σ is denoted by Σ^*. A *language* is defined to be a set of strings over Σ, i.e., a subset of Σ^*.

We only consider deterministic, not necessarily finite automata. We therefore use the following definition, in which both *deterministic* and *not necessarily finite* are omitted from the terminology.

An *automaton* M is a tuple $(Q, \Sigma, \delta, q_0, F)$ where Q is an arbitrary set, Σ is an alphabet, δ is a function $Q \times \Sigma \to Q$, q_0 is an element of Q, and F is a subset of Q. The set Q is called the *state space* of the automaton, δ is the *transition function*, q_0 is the *start state*, and F is the set of *accepting* states.

The language $L(M)$ of an automaton M is defined by

$$L(M) = \{x \in \Sigma^* \mid \hat{\delta}(q_0, x) \in F\} \ ,$$

where $\hat{\delta}$ is the *extended* transition function $Q \times \Sigma^* \to Q$ defined recursively by

$$\begin{aligned}&\hat{\delta}(q, \lambda) = q \ \text{ for all } q \in Q,\\ &\hat{\delta}(q, xa) = \delta(\hat{\delta}(q, x), a) \ \text{ for all } q \in Q,\ x \in \Sigma^*,\ a \in \Sigma.\end{aligned}$$

An automaton is called *finite* if its state space is a finite set. A language L is defined to be *regular* if $L = L(M)$ for some *finite* automaton M.

2 The automaton of a language

Let L be a language over Σ. For any string $x \in \Sigma^*$, we define the language

$$L \ \textbf{after} \ x = \{z \in \Sigma^* \mid xz \in L\} \ .$$

It is easy to prove the following two rules

(0) $$\begin{aligned}&L \ \textbf{after} \ \lambda = L \ ,\\ &(L \ \textbf{after} \ x) \ \textbf{after} \ y = L \ \textbf{after} \ xy \quad \text{for all } x,\ y \in \Sigma^*.\end{aligned}$$

A language L' is called an *after-language* of L when $L' = (L\ \textbf{after}\ x)$ for some string x. By (0), an after-language of an after-language is itself an after-language.

For a given language L_0, we denote the set of all after-languages of L_0 by $\textbf{After}(L_0)$. The elements of $\textbf{After}(L_0)$ are now regarded as states of an automaton $\textbf{Can}(L_0)$, the *canonical automaton*, with the transition function δ given by

$$\delta(L, a) = (L\ \textbf{after}\ a) \quad \text{for } a \in \Sigma \text{ and } L \in \textbf{After}(L_0)\ .$$

We give $\textbf{Can}(L_0)$ the start state L_0, and the set of accepting states

$$\textbf{Accept}(L_0) = \{L \in \textbf{After}(L_0) \mid \lambda \in L\}\ .$$

We now prove that the language of the canonical automaton of language L_0 is L_0 itself. By induction, one first proves that $\hat{\delta}(L, x) = (L\ \textbf{after}\ x)$ for all $x \in \Sigma^*$. It follows that

$$\begin{array}{ll} & x \in L(\textbf{Can}(L_0)) \\ \equiv & \hat{\delta}(L_0, x) \in \textbf{Accept}(L_0) \\ \equiv & (L_0\ \textbf{after}\ x) \in \textbf{Accept}(L_0) \\ \equiv & \lambda \in (L_0\ \textbf{after}\ x) \\ \equiv & x\lambda \in L_0 \\ \equiv & x \in L_0\ . \end{array}$$

This proves that $L(\textbf{Can}(L_0)) = L_0$.

Note that the states of $\textbf{Can}(L_0)$ are languages. They may therefore be not very manageable. In particular it may be difficult to decide whether two states are equal or not. In some cases, however, it is doable, or even easy.

Example. Let L_0 be the language of the regular expression $(\textbf{abc})^*$. This language has precisely four different after-languages: $L_1 = (L_0\ \textbf{after}\ a)$ equals the language of $\textbf{bc}(\textbf{abc})^*$; $L_2 = (L_1\ \textbf{after}\ b)$ equals the language of $\textbf{c}(\textbf{abc})^*$; at this point we are done because of $(L_2\ \textbf{after}\ c) = L_0$. The only other after-language is $\emptyset = (L_0\ \textbf{after}\ b)$ (e.g.). The automaton can be sketched as a wheel with three spokes. L_0 is the only accepting state.

3 Own-languages

Let us again consider an automaton $M = (Q, \Sigma, \delta, q_0, F)$, with its language $L(M)$. For every state $q \in Q$, we define the *own-language* $T(q)$ as the language that would be accepted by the automaton when taking q as the start state. This means:

$$T(q) = \{z \in \Sigma^* \mid \hat{\delta}(q, z) \in F\}\ .$$

Note that $T(q_0) = L(M)$.

It is a matter of straightforward induction to prove that

$$\hat{\delta}(q, xy) = \hat{\delta}(\hat{\delta}(q, x), y) \ \text{ for all } q \in Q,\ x,\ y \in \Sigma^*.$$

An after-language of an own-language is the own-language of the state reached, in the sense that

(1) $$(T(q)\ \textbf{after}\ x) = T(\hat{\delta}(q, x)) \quad \text{for all } x \in \Sigma^*,\ q \in Q.$$

This is proved by observing that, for all $y \in \Sigma^*$,

$$\begin{array}{ll} & y \in T(\hat{\delta}(q, x)) \\ \equiv & \hat{\delta}(\hat{\delta}(q, x), y) \in F \\ \equiv & \hat{\delta}(q, xy) \in F \\ \equiv & xy \in T(q) \\ \equiv & y \in (T(q)\ \textbf{after}\ x)\ . \end{array}$$

Let automaton M be called *reachable* iff, for every $q \in Q$, there is a string $x \in \Sigma^*$ with $q = \hat{\delta}(q_0, x)$. For an automaton $M = (Q, \Sigma, \delta, q_0, F)$, its *reachable* automaton M^o is defined by $M^o = (Q^o, \Sigma, \delta, q_0, F)$ with $Q^o = \{\hat{\delta}(q_0, x) \mid x \in \Sigma^*\}$. It is easy to see that they have the same languages: $L(M^o) = L(M)$.

Theorem 1 *Let M be a reachable automaton with language L. Then T is a surjective function $Q \to$ **After**(L).*

Proof. By formula (1), we have $T(\hat{\delta}(q_0, x)) = (L \textbf{ after } x)$ for every $x \in \Sigma^*$. As M is reachable, this implies that $T(q) \in \textbf{After}(L)$ for all $q \in Q$. It also implies that every element of **After**(L) occurs as an image of T. □

In view of this result, we may regard **Can**(L) as the *minimal automaton* for the language L.

Two states q and $q' \in Q$ are called *equivalent* [Sud06, 5.7.1], when they have the same own-language, i.e., when $T(q) = T(q')$. The theorem implies that function $T : Q \to \textbf{After}(L)$ is bijective if and only if every pair of equivalent states q and q' in Q is equal. It thus follows from the theorem that, given a reachable automaton M for language L, we can obtain an automaton isomorphic to the canonical automaton by identifying equivalent states.

If L is regular, it has a finite automaton M with $L = L(M)$; we may assume that M is reachable; then the surjection $T : M \to \textbf{After}(L)$ implies that **After**(L) is finite. Conversely, if **After**(L) is finite, **Can**(L) serves as a finite automaton for L, so that L is regular. This proves:

Corollary 2 *A language is regular if and only if it has only finitely many different after-languages.*

Remark. For a language L, the right equivalence relation $\sim_L$ on Σ^* is defined by

$$x \sim_L y \quad \equiv \quad (\forall\, w \in \Sigma^* : xw \in L \equiv yw \in L)\ .$$

The classical Theorem of Myhill-Nerode, [RS59, Thm. 2] or [Sud06, Thm. 6.7.4], states that language L is regular if and only if relation $\sim_L$ has finitely many equivalence classes. The definition of **after** implies that

$$x \sim_L y \quad \equiv \quad L \textbf{ after } x = L \textbf{ after } y\ .$$

We have therefore a bijection between the equivalence classes of $\sim_L$ and the after-languages of L. This shows that Corollary 2 is nothing but a rephrasing of the Theorem of Myhill-Nerode. □

Corollary 3 *If a language is regular, all its after-languages are regular as well.*

Proof. Let L_0 be a regular language and let $L_1 = (L_0 \textbf{ after } x)$ for some string x. All after-languages of L_1 are after-languages of L_0. Since L_0 has only finitely many different after-languages, language L_1 also has only finitely many different after-languages. Therefore L_1 is regular. □

Remark. If a language L is context-free, all its after-languages are also context-free. This is easily proved by means of the Greibach normal form, e.g., [Sud06, 4.8]. Alternatively, one can define an operation **after** on context-free grammars such that the language generated by an after-grammar is an after-language of the language generated by the grammar. □

4 Reversal of languages and automata

We now show that, if a language is given by an automaton, the reversal of the automaton is a minimal automaton for the reversed language, modulo reachability, see, e.g., [vdS85].

If x is a string, the *reversed string* is denoted by x^R. If L is a language, the *reversed language* is $R(L) = \{x \mid x^R \in L\}$.

Let $M = (Q, \Sigma, \delta, q_0, F)$ be an automaton. The *reversed automaton* is defined as $R(M) = (\mathbb{P}(Q), \Sigma, \delta', F, F')$ where δ' and F' are given by

$$\begin{array}{l} \delta'(S, a) = \{q \in Q \mid \delta(q, a) \in S\} \ , \\ S \in F' \quad \equiv \quad q_0 \in S \ . \end{array}$$

By induction one proves that $\hat{\delta}'(S, x) = \{q \in Q \mid \hat{\delta}(q, x^R) \in S\}$ for all $S \in \mathbb{P}(Q)$ and $x \in \Sigma^*$. We use this to determine the own-language $T(S)$ of a state $S \in \mathbb{P}(Q)$ of automaton $R(M)$ in:

$$\begin{array}{ll} & x \in T(S) \\ \equiv & \hat{\delta}'(S, x) \in F' \\ \equiv & q_0 \in \hat{\delta}'(S, x) \\ \equiv & \hat{\delta}(q_0, x^R) \in S \ . \end{array}$$

This proves $T(S) = \{x \mid \hat{\delta}(q_0, x^R) \in S\}$. In particular, we have

$$\begin{array}{ll} & L(R(M)) \\ = & T(F) \\ = & \{x \mid \hat{\delta}(q_0, x^R) \in F\} \\ = & \{x \mid x^R \in L(M)\} \\ = & R(L(M)) \ . \end{array}$$

This proves that the language of the reversed automaton is the reversal of the language of the automaton.

Theorem 4 *Let M be an automaton. Then the reachable automaton $R(M^o)^o$ is minimal.*

Proof. It suffices to prove that reachable states S and S' of $R(M^o)$ are equal when their own-languages are equal. So, assume that $T(S) = T(S')$. The above result implies

$$\forall\, x : \ \hat{\delta}(q_0, x^R) \in S \equiv \hat{\delta}(q_0, x^R) \in S' \ .$$

Since automaton M^o is reachable, this implies $S = S'$. □

References

[Eil74] S. Eilenberg. *Automata, Languages, and Machines, volume A.* Academic Press, 1974.

[Ner58] A. Nerode. Linear automaton transformations. *Proc. AMS*, 9:541–544, 1958.

[RS59] M.O. Rabin and D. Scott. Finite automata and their decision problems. *IBM J. Res.*, 3:115–125, 1959.

[Sud06] Th.A. Sudkamp. *Languages and Machines.* Pearson, 3d edition, 2006.

[vdS85] J. van de Snepscheut. *Trace theory and VLSI design*, volume 200 of *LNCS.* Springer V., 1985.