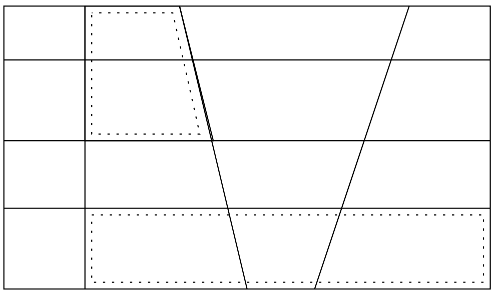
Lecture 1: Regular Languages and Monoids

We shall assume familiarity with the definitions and basic results regarding regular languages and finite automata (as presented in Hopcroft and Ullman [1] or Kozen [2]) and begin by recalling their connections to Myhill-Nerode relations.

1 Myhill-Nerode Characterization

An equivalence relation $\sim$ over Σ^*

- is a *right congruence* if $x \sim y$ implies $xz \sim yz$ for every $x, y, z \in \Sigma^*$
- is *of finite index* if $\Sigma^*/\sim$ is finite.
- *saturates* a language L if $x \sim y \;\Rightarrow\; (x \in L \text{ iff } y \in L)$. Or equivalently, L is the union of some of the equivalence classes of $\sim$, or equivalently for each $x \in \Sigma^*$, $[x]_\sim \cap L = \emptyset$ or $[x]_\sim \subseteq L$. This is illustrated by the following diagram. The entire rectangle corresponds to Σ^* and the individual regions inside are the equivalence classes under $\sim$ and the regions enclosed by the dotted lines are those that are contained in L. Note that every region is either entirely contained in L or is disjoint from L.



Theorem 1 *(Myhill-Nerode) A language L is regular if and only if there is a right congruence $\sim$ of finite index, that saturates L.*

From any finite automaton $A = (Q, \Sigma, \delta, s, F)$ recognising L it is easy to construct a right congruence $\sim_A$ of the desired kind.

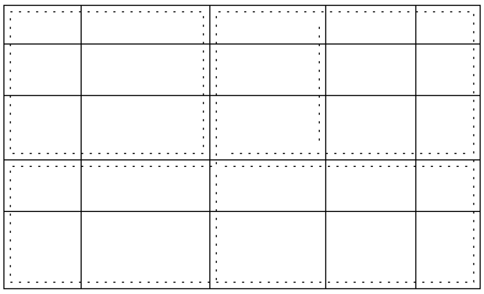
$$x \sim_A y \quad \Longleftrightarrow \quad \delta(s, x) = \delta(s, y)$$

For the converse, an automaton recognising L can be constructed as $A_\sim = (\Sigma^*/\sim, \Sigma, \delta, [\epsilon], F)$ where, $F = \{[x]_\sim \mid x \in L\}$ and $\delta([x]_\sim, a) \;=\; [xa]_\sim$.

With every language L (regular or otherwise) one can associate the "coarsest" right congruence that saturates L ($\sim_L$) as follows:

$$x \sim_L y \quad \Longleftrightarrow \quad \forall z.(xz \in L \iff yz \in L)$$

Quite evidently this relation is a right congruence that saturates L. It is also the coarsest because, if $\sim$ is any other right congruence that saturates L and $x \sim y$ then $x \sim_L y$ — suppose not, then there is an z such that (w.l.o.g.) $xz \in L$ and $yz \notin L$, contradicting the right congruent property of $\sim$. Thus, not only does $\sim_L$ saturate L, but further if $\sim$ is any right congruence saturing L then the equivalences classes of $\sim$ can be coalesced to form the equivalence classes of $\sim_L$.



In the above diagram the regions enclosed by the solid lines are the equivalence classes induced by $\sim$ and they are all entirely contained inside the equivalence classes induced by $\sim_L$ (the regions enclosed by the dotted lines).

If the language L is regular then $\sim_L$ is of finite index (since we can start with any finite index relation saturating L and coalesce its states to obtain $\sim_L$). The automaton obtained from $\sim_L$ is the *minimal* automaton for L.

An equivalence relation $\equiv$ is said to be a *congruence* if $x \equiv y$ implies $uxv \equiv uyv$ for all $u, v, x, y \in \Sigma^*$. It is quite easy to show that a language is regular if and only if there is congruence of finite index that saturates L. The construction of the automaton recognising L from the congruence is identical to the one described above. For the other direction, starting with an automaton $A = (Q, \Sigma, \delta, s, F)$ with no unreachable states, define $x \equiv_A y$ iff for all $q \in Q$, $\delta(q, x) = \delta(q, y)$. (Check that this relation is indeed a congruence and that it saturates L.)

One can also define a canonical congruence for each language L, given by $x \equiv_L y$ if and only if for all $u, v \in \Sigma^*$ $uxv \in L$ if and only if $uyv \in L$. Understandably, this is the "coarsest" congruence saturating L (verify this), so that starting with any other congruence saturating L, one may obtain this congruence by coalescing some of the equivalence classes together.

Exercise: Verify that if A is the minimal automaton for L then $\equiv_A$ is $\equiv_L$.

2 Monoids

A monoid is set M along with a associative binary operation . and a special element $e \in M$ which acts as the identity element w.r.t to .. We write $(M, ., e)$ to describe a monoid, but very often we shall write M instead. $(\mathbb{N}, +, 0)$ is a monoid and so is $(\Sigma^*, ., \epsilon)$ where . is the concatenation operation. These monoids are "infinite" as the underlying set is an infinite set. An example of a finite monoid is $(\mathbb{Z}_n, +, 0)$. Another class of finite monoids comes from functions over a finite set. Let S be a set and let F be the set of functions from S to S and let Id_S be the identity function. Define $f \circ g$ to be the composition of g with f, i.e., $f \circ g(x) = g(f(x))$. Then $(F, \circ, Id_S)$ forms a monoid.

Given a finite automaton $A = (Q, \Sigma, \delta, s, F)$, the set of functions on Q defines a finite monoid. But there is a second and significantly more interesting monoid that one can associate with A. Let $M_A = (\{\hat{\delta}_x \mid x \in \Sigma^*\}, \circ, \hat{\delta}_\epsilon = Id_Q)$ where, $\hat{\delta}_x$ is the function from Q to Q defined by $\hat{\delta}_x(q) = \hat{\delta}(q, x)$. This monoid consists of those functions over Q that are defined as transition functions of words over Σ^*. Thus, it forms a *submonoid* of the set of functions over Q (any subset of a monoid containing the identity and closed w.r.t. the operation of the monoid is called a submonoid). This monoid associated with the automaton A is called the *transition monoid* of A and will play a critical role in the following developments.

A (homo)morphism from a monoid $(M, ., e)$ to a monoid $(N, *, f)$ is a function $h : M \longrightarrow N$ such that $h(x.y) = h(x) * h(y)$ and $h(e) = f$. For example, $\text{len} : \Sigma^* \longrightarrow \mathbb{N}$ with $\text{len}(x) = |x|$ is a morphism. The monoid $(\Sigma^*, ., \epsilon)$ is also called as the *free monoid* over Σ because, given any monoid $(N, *, f)$ and a function $f : \Sigma \longrightarrow N$, we can define a morphism $\hat{f}$ from $(\Sigma^*, ., \epsilon)$ to $(N, *, f)$ such that $\hat{f}(a) = f(a)$ for each $a \in \Sigma$ (the definition of $\hat{f}$ is quite obvious).

2.1 Monoids as Recognizers

We shall use monoids as recognizers of languages. Given a monoid $(M, ., e)$, a subset X of M and a morphism h from Σ^* to M, *the language defined by X w.r.t. to the morphism h* is $h^{-1}(X)$. We say that a language L is recognised by a monoid M if there is a morphism h and $X \subseteq M$ such that $L = h^{-1}(X)$. The interesting case is when M is a finite monoid.

Theorem 2 *L is a regular language if and only if it is recognised by some finite monoid.*

Proof: Suppose L is recognised by the monoid M via the morphism h and the subset X. Define the automaton $A_M = (M, \Sigma, \delta, e, X)$ where $\delta(m, a) = m.h(a)$. Then, $\hat{\delta}(m, a_1 a_2 \ldots a_n) = m.h(a_1).h(a_2).\ldots.h(a_n)$ and therefore $\hat{\delta}(e, a_1 a_2 \ldots a_n) = e.h(a_1).h(a_2)\ldots h(a_n) = h(a_1 a_2 \ldots a_n)$. Thus, $L(A_M) = \{x \mid h(x) \in X\} = L(A_M) = L$.

For the converse, let A be any automaton recogising L. Consider the transition monoid $M_A = (\{\hat{\delta}_x \mid x \in \Sigma^*\}, \circ, Id_Q)$ and the morphism h from Σ^* to M_A defined by $h(x) = \hat{\delta}_x$. The pre-image under h of $X = \{\hat{\delta}_x \mid \hat{\delta}(s, x) \in F\}$ is easily seen to be L. Thus, A is recognised by a finite monoid. ■

2.2 The Syntactic Monoid

With each regular language L we can associate a canonical (in a manner to be explained soon) monoid that recognizes L. We associate a monoid structure on $\Sigma^*/\equiv_L$ by $[x]_{\equiv_L}.[y]_{\equiv_L} = [xy]_{\equiv_L}$. It is easy to check that with this operation $\Sigma^*/\equiv_L$ forms a monoid with $[\epsilon]_{\equiv_L}$ as the identity. The natural morphism η_L defined by $\eta_L(x) = [x]_{\equiv_L}$ recognises L as the pre-image of $X = \{[x]_{\equiv_L} \mid x \in L\}$. This monoid, denoted $\mathsf{Syn}(L)$, is called the *syntactic monoid* of L.

Exercise: Show by an example that $\sim_L$ is not a congruence in general. Thus, there is no monoid structure on $\Sigma^*/\sim_L$.

Exercise: What is the syntactic monoid of the language $(aa)^*$?

This monoid is canonical because, first of all, this is the smallest monoid that recognises L, and more importantly, η_L factors via every homomorphism (to any monoid M) that recognises L. This is the import of the following theorem

Theorem 3 *Let L be a regular language and suppose that L is recognised by the M via the morphism h. Then there is a morphism h_L from $h(M)$ (where $h(M)$ is the submonoid of M consisting of all the elements in the image of h) to $\mathsf{Syn}(L)$ such that $\eta_L = h \circ h_L$.*

$$\begin{tikzcd}
\Sigma^* \arrow[r, "h"] \arrow[dr, "\eta_L"'] & h(\Sigma)^* \arrow[r, hook] \arrow[d, dashed, "h_L"'] & M \\
& \mathsf{Syn}(L) &
\end{tikzcd}$$

Proof: Note that $\equiv_h$ defined by $x \equiv_h y$ if and only if $h(x) = h(y)$ is a congruence that saturates L: If $h(x) = h(y)$ then $h(uxv) = h(u)h(x)h(v) = h(u)h(y)h(v) = h(uyv)$. Thus, $\equiv_h$ is a congruence. Further, if $x \equiv_h y$ and $h(x) \in X$ then $h(y) \in X$. Hence it also saturates L. Thus, $\equiv_h$ refines $\equiv_L$ (i.e. each equivalence class of $\equiv_h$ is completely contained in some equivalence class of $\equiv_L$.)

Note that $\equiv_{\eta_L}$ is the same as $\equiv_L$. Hence, we may define the function h_L from $h(M)$ to $\mathsf{Syn}(L)$ as $h_L(h(x)) = \eta_L(x)$. This function is well-defined since we know that $h(x) = h(y)$ implies $\eta_L(x) = \eta_L(y)$. Clearly this map h_L is a morphism and by construction $h_L \circ h(x) = \eta_L(x)$. ■

Exercise: Prove that the syntactic monoid of a regular language L is isomorphic to the transition monoid of the minimal automaton for L.

We say that a monoid M *divides* a monoid N (written $M \prec N$) if M is the homomorphic image of a submonoid of N. In this language, the above theorem can be restated as

Theorem 4 *A monoid M recognises a regular language L only if $\mathsf{Syn}(L) \prec M$.*

We shall return to the study of regular languages via monoids after a couple of lectures. We shall see how we can use the structure of syntactic monoids to characterise subclasses of regular languages.

References

[1] John E. Hopcroft and Jeffrey D. Ullman: *Introduction to automata theory, languages and computation*, Addison-Wesley, 1979.

[2] Dexter Kozen: *Automata and Computability*, Springer-Verlag, 1997.