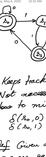


Keep track of minimizers not δ.
Not necessary: too many states.
How to minimize a DFA.

$\delta(s_1, a) = s_2 = \delta(s_1, a)$
$\delta(s_2, a) = s_1 = \delta(s_2, a)$

Def Given a DFA $M = (S, s_0, \delta, F)$, $s \in S$, p, q are **equivalent**, $p \approx_M q$ if
$\forall x \in \Sigma^*\ \delta^*(p, x) \in F \Leftrightarrow \delta^*(q, x) \in F$.

Remark $p \not\approx_M q$ means $\exists x \in \Sigma^*$ s.t.
$\delta^*(p, x) \in F \wedge \delta^*(q, x) \notin F$ or
$\delta^*(p, x) \notin F \wedge \delta^*(q, x) \in F$.

Observation $\approx$ is an equivalence rel.

Lemma A $p \approx_M q \Leftrightarrow \forall a \in \Sigma\ \delta(p, a) \approx \delta(q, a)$.

$[p] = \{s \mid s \approx_M p\}$ $[p] \approx [q] = [\delta(p, a)]$

Lemma A restated $[p] \to [q] \Leftrightarrow [\delta(p, a)] = [\delta(q, a)]$

We define a new machine

$M' = (S', s_0', \delta', F')$

$S' = S/\approx$ $s_0' = [s_0]$

$F' = \{[s] \mid s \in F\}$ $\delta'([s], a) = [\delta(s, a)]$

Lemma B $p \in F \Leftrightarrow [p] \in F'$

Lemma C $\forall x \in \Sigma^*\ \delta'^*([s], x) = [\delta^*(s, x)]$

Proof By induction on the length of x.

Thm $L(M) = L(M')$

Proof $x \in L(M) \Leftrightarrow \delta'^*([s_0], x) \in F'$
$\Leftrightarrow [\delta^*(s_0, x)] \in F' \Leftrightarrow \delta^*(s_0, x) \in F$
$\Leftrightarrow x \in L(M)$ ☺

SPLITTING ALGORITHM
$P \approx_M$ if $\exists w \in \Sigma^*$ s.t. $\delta^*(p, w) \in F$ & $\delta^*(q, w) \notin F$ or vice versa.

FACT if $\exists w \in \Sigma^*$ s.t. $\delta^*(p, w) \in F$ & $\delta^*(q, w) \notin F$
then $P \not\approx_M$.

ALG Define a $S \times S$ array of bool.
1. For every (p, q) s.t. $p \in F$ & $q \notin F$ put a 1 in the (p, q) cell of the matrix.
2. Repeat until no more changes
   For each pair (p, q) that is not already marked with a 1, check if $\exists a \in \Sigma$ s.t. $(\delta(p, a), \delta(q, a))$ is marked with a 1. If so mark (p, q) 1

Thm If all the and the states are not marked 0 then they are equivalent.

$O(n^4) \to$ naive
$O(n^2 k) \to$ using basic data structure
$O(k \log n) \to$ Hopcroft
$O(2^n) \to$ BRZOZOWSKI

There is a **unique** smallest DFA to recognize a regular language.

MYHILL-NERODE THEOREM (1957)

Def An eq. rel. $R \subseteq \Sigma^* \times \Sigma^*$ is said to be **right-invariant** if
$\forall x, y \in \Sigma^*, \forall z\ xRy \Leftrightarrow \forall y \in \Sigma^*\ xzRyz$.

Ex 1 DFA $M = (Q, s_0, \delta, F)$
$xRy \Leftrightarrow \delta^*(s_0, x) = \delta^*(s_0, y)$.

Ex 2 Given $L \subseteq \Sigma^*$, not necessarily regular, we define R_L as $x R_L y$
$\Leftrightarrow \forall z \in \Sigma^*\ xz \in L \Leftrightarrow yz \in L$.

Terminology If we have a u.b.S and an eq. rel. $\sim$ on the index and an eq. rel. $\sim$ on the number of equivalence classes.

Thm (Myhill-Nerode) TFAE
(1) L is regular (i.e. $\exists$ a DFA)
(2) L is the union of the equivalence classes of a right-invariant equivalence relation R of finite index.
(3) Any equivalence relation R with the properties described in (2) has to refine R_L.

Proof (1) $\Rightarrow$ (2) We have M, $L(M) = L$.
Construct R_M as in Ex 1.
$S_f = \{s \mid \delta^*(s_0, x) \in F\} - [\varepsilon]$
$L = \bigcup_{s \in S_f} S_f$

(2) $\Rightarrow$ (3) We have R, right-invariant. R has finite index, L is the union of some of the equivalence classes of R.
WTS $xRy \Rightarrow x R_L y$.

Suppose $x \not\approx_L y$. Then $\exists z$ such that one of $xz \in L$ & $yz \notin L$. Now suppose that xRy & $xz \in L$. Then yz must be in the same eq. class as xz & R & the whole in same eq. class has to be in L. Since R is right-invariant xRy then $\forall z\ xz R yz$ & so the same R eq. class as yz. So $yz \in L$. Similarly with xz instead of yz.
$\forall z\ xz \in L \Leftrightarrow yz \in L$ i.e. $x \approx_L y$.

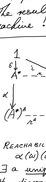
(3) $\Rightarrow$ (1) Construct an automaton from the equivalence classes of R_L.
(details in the notes)

---

(2) $\Rightarrow$ (3) shows that the automaton constructed in the proof of (2) $\Rightarrow$ (1) is the **UNIQUE** smallest DFA to recognize L.

BRZOZOWSKI'S ALGORITHM:
1. Reverse all the arrows, change the initial state to an accept state & make all the accept states that states.
2. Determinize it
3. Remove unreachable states
4. Repeat 1
5. Repeat 2
6. Repeat 3

The result is the minimal machine!!!



REACHABILITY & OBSERVABILITY
$\alpha(a)(a) = a \cdot a$.

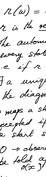
$\exists!$ a **unique** map $\alpha : A^* \to X$ s.t. the diagram above **commutes**.

$f : V \to W$
$f^* : V^* \to W^*$
$q \in V^*$
$f^*(q)(a) = f(q(a))$
$f^*(q) = a \mapsto f(q(a))$

$r(w) = \delta^*(s_0, w)$

r is the **reachability** map.
The automaton is reachable if every state can be reached, i.e. if r is **surjective**.

$\exists!$ a unique map $o : X \to 2^{A^*}$ s.t.



The diagram commutes.
o maps a state in X to the language accepted if that state happened to be the start state.

o is observability map. 2 states can be told apart exactly when $o(x) \neq o(y)$.
We say an automaton is **minimal** if any 2 states can be distinguished, i.e. if o is **injective**.

A minimal automaton is reachable & observable.

Thm (1) If (X, δ, s, f) is reachable then $(2^X, \delta^*, f^*, f^*)$ is observable.
(2) If (X, δ, s, f) accept L then $(2^X, \delta^*, f, 2^L)$ accepts L.

This minimization can be done for automata with vector space structure (weighted automata), topological structure, convex structure, lattice structure...

Set

$X \xrightarrow{f} Y$

$2^{(-)} : 2^Y \xrightarrow{f^*} 2^X$

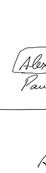
$2^f(B \subseteq Y) = f^{-1}(B) \subseteq X$

$1 \xrightarrow{i} X$

$2^X \xrightarrow{i^*} 2$

Alexandra Silva, Helle Hansen, Bart Jacobs, Ranzato, Rajamani, Bonchi, Pous, Rutten, Kupke

---



$(S, A, s_0 \in S, Obs, T : S \to Obs)$
$w \models Obs$ is a modal formula
if $a \in A$ & $\varphi \in$ formula
$a \cdot \varphi$ is a formula $\langle a \cdot \varphi \rangle$

Chris Heunen, P, Jacobs, Doina, Monica Dinculescu