On Two Letters versus Three

Dexter Kozen
Department of Computer Science
Cornell University
Ithaca, New York 14853-7501, USA
kozen@cs.cornell.edu

February 4, 2002

Abstract

If A is a context-free language over a two-letter alphabet, then the set of all words obtained by sorting words in A and the set of all permutations of words in A are context-free. This is false over alphabets of three or more letters. Thus these problems illustrate a difference in behavior between two- and three-letter alphabets.

The following problem appeared on a recent exam at Cornell:

> Let Σ be a finite alphabet with a fixed total ordering on the letters. For a string $x \in \Sigma^*$, let $\mathsf{sort}\ x$ be the string obtained by sorting the letters in increasing order. For example, if $a < b < c$, then $\mathsf{sort}\ abacbaa = aaaabbc$. For $A \subseteq \Sigma^*$, let $\mathsf{sort}\ A = \{\mathsf{sort}\ x \mid x \in A\}$. Of the following three statements, two are false and one is true. Give counterexamples for the two false ones and a proof of the true one.
>
> (i) If A is regular, then so is $\mathsf{sort}\ A$.
>
> (ii) If A is context-free, then so is $\mathsf{sort}\ A$.
>
> (iii) If A is context-sensitive, then so is $\mathsf{sort}\ A$.

One might also ask the same questions about $\mathsf{perm}\ A$, the set of all permutations of words in A.

Of course, it is (i) and (ii) that are false, since

$$\mathsf{sort}\ (abc)^* = \mathsf{perm}\ (abc)^* \cap a^*b^*c^* = \{a^nb^nc^n \mid n \geq 0\}.$$

Interestingly, (ii) is true for both sort and perm over a two-letter alphabet. This is quite surprising: whereas a two-letter alphabet is exponentially more succinct than a one-letter alphabet, one does not normally think of a break in behavior between two- and three-letter alphabets. In many applications, three letters (or for that matter any fixed finite number of letters) can be coded into two with only a linear loss of efficiency. Not so, apparently, in this case.

In this short note we give an elementary proof of these facts. The proof for sort is a fairly straightforward construction relying on Parikh's theorem and Pilling normal form, but the proof for perm is somewhat more involved, requiring a bit of linear algebra over integer lattices.

Let $\Sigma = \{a_1, \ldots, a_d\}$, and let $\pi : \Sigma^* \to \mathbb{N}^d$ be the Parikh map

$$\pi(x) \stackrel{\text{def}}{=} (\#a_1(x), \ldots, \#a_d(x)),$$

where $\#a(x)$ is the number of a's in x. Define

$$\begin{aligned} \pi(A) &\stackrel{\text{def}}{=} \{\pi(x) \mid x \in A\} \\ \mathsf{perm}\ A &\stackrel{\text{def}}{=} \pi^{-1}(\pi(A)) \\ \mathsf{sort}\ A &\stackrel{\text{def}}{=} \mathsf{perm}\ A \cap a_1^* \cdots a_d^*. \end{aligned}$$

Theorem 1 *For $d \leq 2$, if A is a context-free language, then so are* $\mathsf{perm}\ A$ *and* $\mathsf{sort}\ A$.

This is trivial for $d = 1$ and false for $d \geq 3$. The interesting case is $d = 2$.

Lemma 1 *It suffices to prove Theorem 1 for A regular. When manipulating regular expressions, we can also use the commutativity axiom $xy = yx$.*

Proof. This is a consequence of Parikh's theorem (the commutative image of any context-free language is the commutative image of some regular set), observing that the definitions of $\mathsf{perm}\ A$ and $\mathsf{sort}\ A$ depend only on the commutative image $\pi(A)$ of A. □

Lemma 2 *It suffices to prove Theorem 1 for A of the form $xy_1^* \cdots y_k^*$, where $x, y_1, \ldots, y_k \in \Sigma^*$.*

Proof. Under commutativity, every regular expression is equivalent to a sum of expressions of this form. This is known as Pilling normal form (see [1]). □

Here is a direct construction for $\mathsf{sort}\ A$. This result will also follow from the result for $\mathsf{perm}\ A$ by intersecting with a^*b^*, but the proof for $\mathsf{perm}\ A$ is somewhat harder.

Without loss of generality, assume A is of the form of Lemma 2. Let $m = \#a(x)$, $n = \#b(x)$, $m_i = \#a(y_i)$, and $n_i = \#b(y_i)$, $1 \leq i \leq k$. A context-free grammar for $\mathsf{sort}\ A$ is

$$\begin{aligned} S &\rightarrow a^m T_1 b^n \\ T_i &\rightarrow a^{m_i} T_i b^{n_i} \mid T_{i+1}, \quad 1 \leq i \leq k-1 \\ T_k &\rightarrow a^{m_k} T_k b^{n_k} \mid \varepsilon. \end{aligned}$$

For $\mathsf{perm}\ A$, we will need to use some linear algebra on integer lattices.

Lemma 3 *Let $y_1, \ldots, y_n$ be nontrivial. The following are equivalent:*

(i) $\pi(y_1), \ldots, \pi(y_n)$ *are linearly dependent over* $\mathbb{Q}$.

(ii) $\pi(y_1), \ldots, \pi(y_n)$ *are linearly dependent over* $\mathbb{Z}$.

(iii) *There exists a partition of $y_1, \ldots, y_n$ into two nonempty disjoint sets $y_1, \ldots, y_k$ and $y_{k+1}, \ldots, y_n$ (renumbering if necessary) and coefficients $a_i \in \mathbb{N}$, $1 \leq i \leq n$, such that not all $a_i = 0$, $1 \leq i \leq k$, not all $a_i = 0$, $k+1 \leq i \leq n$, and $\prod_{i=1}^{k} y_i^{a_i} = \prod_{i=k+1}^{n} y_i^{a_i}$.*

The property in (iii) regarding the vanishing of the coefficients follows from the observation that we cannot have $\prod_{i=1}^{k} y_i^{a_i} = 1$ with $a_i \in \mathbb{N}$ unless all $a_i = 0$.

The following lemma gives a stronger version of Pilling normal form.

Lemma 4 (Conway [1, Theorem 2, p. 92]) *Any regular subset of $\mathbb{N}^d$ can be written as a sum of terms of the form $xy_1^* \cdots y_n^*$ with $\pi(y_1), \ldots, \pi(y_n)$ linearly independent over $\mathbb{Q}$.*

Proof. Suppose $\pi(y_1), \ldots, \pi(y_n)$ are linearly dependent. Let $\prod_{i=1}^{k} y_i^{a_i} = \prod_{i=k+1}^{n} y_i^{a_i}$ with $a_i \in \mathbb{N}$, $1 \leq i \leq n$, not all $a_1, \ldots, a_k = 0$ and not all $a_{k+1}, \ldots, a_n = 0$. Using the Kleene algebra identities

$$\begin{aligned} y^* &= (\sum_{i=0}^{n-1} y^i)(y^n)^* \\ x_1^* \cdots x_n^* &= (x_1 \cdots x_n)^* (\sum_{i=1}^{k} \prod_{\substack{1 \leq j \leq k \\ j \neq i}} x_j^*) \end{aligned}$$

(the second one requires commutativity), rewrite $y_1^* \cdots y_k^*$ as $\alpha(y_1^{a_1})^* \cdots (y_k^{a_k})^*$, where

$$\alpha \;=\; \prod_{i=1}^{k} \sum_{j=0}^{a_i - 1} y_i^j,$$

and then $(y_1^{a_1})^* \cdots (y_k^{a_k})^*$ as

$$(y_1^{a_1} \cdots y_k^{a_k})^* (\sum_{i=1}^{k} \beta_i),$$

where

$$\beta_i \;=\; \prod_{j \neq i} (y_j^{a_j})^*, \quad 1 \leq i \leq k.$$

Note α contains no starred terms, so it can be expressed as a finite sum of products of the y_i. Then $y_1^* \cdots y_n^*$ can be written as a sum of terms of the form

$$u(y_1^{a_1} \cdots y_k^{a_k})^* \beta_i y_{k+1}^* \cdots y_n^*.$$

Now we can replace $\prod_{i=1}^{k} y_i^{a_i}$ with $\prod_{i=k+1}^{n} y_i^{a_i}$ to get

$$u(y_{k+1}^{a_{k+1}} \cdots y_n^{a_n})^* \beta_i y_{k+1}^* \cdots y_n^*.$$

Since this is contained in $y_1^* \cdots y_n^*$, we have $u \in y_1^* \cdots y_n^*$, thus

$$u(y_{k+1}^{a_{k+1}} \cdots y_n^{a_n})^* \beta_i y_{k+1}^* \cdots y_n^* \quad \subseteq \quad u\beta_i' y_{k+1}^* \cdots y_n^*,$$

where

$$\beta_i' \;=\; \prod_{j \neq i} y_j^*, \quad 1 \leq i \leq k.$$

Thus the original term $xy_1^* \cdots y_n^*$ can be written as a sum of terms of the same form but with one fewer starred y_i.

We can continue decreasing the number of starred terms inductively until the y_i are linearly independent. $\square$

By this lemma, to prove Theorem 1 for the case $\mathsf{perm}\ A$, it suffices to consider A of the form xu^* or xu^*v^*, where $\pi(u)$ and $\pi(v)$ are linearly independent. Note that the dimension is at most two since we are over a two-letter alphabet. We can get rid of the x without loss of generality by $|x|$ applications of the following lemma:

Lemma 5 *Let $a \in \Sigma$. If A is context-free, then so is $\{xay \mid xy \in A\}$. It follows that if* $\mathsf{perm}\ A$ *is context-free, then so is* $\mathsf{perm}\ aA$*, since* $\mathsf{perm}\ aA = \{xay \mid xy \in \mathsf{perm}\ A\}$.

Proof. Consider a Chomsky normal form grammar for $\mathsf{perm}\ A$. For every nonterminal X, add a new nonterminal X_a, which is meant to generate all the strings that X generates but with an extra a somewhere. For every production $X \to YZ$, add the productions $X_a \to Y_a Z \mid Y Z_a$. For every production $X \to b$, add the productions $X_a \to ba \mid ab$. For every production $X \to \varepsilon$, add the production $X_a \to a$. The new start symbol is S_a, where S was the old start symbol. □

Now we show that $\mathsf{perm}\ u^* v^*$ is context-free. (We leave the easier case, $\mathsf{perm}\ u^*$, as an exercise for the interested reader.) Suppose $\#a(u) = u_1$, $\#b(u) = u_2$, $\#a(v) = v_1$, $\#a(v) = v_2$; thus $\pi(u) = (u_1, u_2)$ and $\pi(v) = (v_1, v_2)$. Arrange $\pi(u)$ and $\pi(v)$ in a 2×2 matrix

$$A \stackrel{\text{def}}{=} \begin{bmatrix} u_1 & v_1 \\ u_2 & v_2 \end{bmatrix}$$

with positive determinant $\Delta = u_1 v_2 - u_2 v_1 > 0$. (The sign of the determinant is determined by the orientation of u and v; exchange if necessary to make it positive.) The adjoint (pseudo-inverse) of A is

$$A' \stackrel{\text{def}}{=} \begin{bmatrix} v_2 & -v_1 \\ -u_2 & u_1 \end{bmatrix}$$

and satisfies the property

$$AA' = A'A = \begin{bmatrix} \Delta & 0 \\ 0 & \Delta \end{bmatrix}.$$

Now we give a nondeterministic one-way automaton with an integer counter accepting $\mathsf{perm}\ u^* v^*$. The machine actually keeps three counters, c_1, c_2, c_3, but the counters c_1 and c_3 hold only finitely many values and can be stored in the finite control. The counter c_2 holds an integer. We can simulate this with a pushdown automaton with a single-letter stack, keeping the sign in the finite control.

The automaton starts in the state $c_1 = c_2 = c_3 = 0$ and takes the following actions on each input symbol: on input a,

$$\begin{aligned} c_1 &:= (c_1 + v_2) \bmod \Delta \\ c_2 &:= c_2 - u_2 \\ c_3 &:= \min(c_3 + 1, v_1) \end{aligned}$$

and on input b,

$$\begin{aligned} c_1 &:= (c_1 - v_1) \bmod \Delta \\ c_2 &:= c_2 + u_1. \end{aligned}$$

In addition, it may nondeterministically choose to take the following *reset step* whenever $c_3 = v_1$ without reading an input symbol.

$$\begin{aligned} c_2 &:= c_2 - \Delta \\ c_3 &:= 0. \end{aligned}$$

Thus after scanning a prefix y of the input string,

$$\begin{aligned} c_1 &= (v_2\#a(y) - v_1\#b(y)) \bmod \Delta \\ c_2 &= -u_2\#a(y) + u_1\#b(y) - \Delta q, \end{aligned} \tag{1}$$

where q is the number of resets that have occurred, and c_3 contains the number of a's seen since the last reset, up to a maximum of v_1. The automaton accepts if $c_1 = c_2 = 0$.

Now we show that the automaton accepts $\mathsf{perm}\ u^*v^*$. For $s, t \in \mathbb{Z}^2$, note that $As = t$ iff $A't = \Delta s$. Applying this with $s = (p, q)$ and $t = (\#a(x), \#b(x))$, we have

$$\begin{aligned} \#a(x) &= u_1 p + v_1 q \\ \#b(x) &= u_2 p + v_2 q \end{aligned} \tag{2}$$

iff

$$\begin{aligned} v_2\#a(x) - v_1\#b(x) &= \Delta p \\ -u_2\#a(x) + u_1\#b(x) &= \Delta q. \end{aligned} \tag{3}$$

This implies that the following are equivalent:

(i) $x \in \mathsf{perm}\ u^*v^*$

(ii) there exist $p, q \in \mathbb{N}$ such that $x \in \mathsf{perm}\ u^p v^q$

(iii) there exist $p, q \in \mathbb{N}$ satisfying either of the equivalent conditions (2) or (3).

Now suppose $x \in \mathsf{perm}\ u^*v^*$ and condition (iii) holds with $p, q \in \mathbb{N}$. Let the automaton choose to perform the reset step at its earliest opportunity while scanning x (i.e., as soon as the counter c_3 reaches v_1), but only q times. It has the opportunity to perform a reset at least q times, since by (2), $\#a(x) \geq v_1 q$. By (1), the final values of c_1 and c_2 are

$$\begin{aligned} (v_2\#a(x) - v_1\#b(x)) \bmod \Delta &= 0 \\ -u_2\#a(x) + u_1\#b(x) - \Delta q &= 0, \end{aligned}$$

respectively, so the machine accepts.

Conversely, suppose the machine accepts. Let q be the number of times the reset occurred. By (1), there exists $p \in \mathbb{Z}$ such that (3) holds, and we need only show that $p \geq 0$. Since the reset occurred q times, we have $\#a(x) \geq v_1 q$. Then

$$\begin{aligned} u_1 v_2 p &= \Delta p + u_2 v_1 p \\ &= v_2\#a(x) - v_1\#b(x) + u_2 v_1 p \\ &\geq v_2 v_1 q - v_1(u_2 p + v_2 q) + u_2 v_1 p \\ &= 0. \end{aligned}$$

But $u_1 v_2 = \Delta + u_2 v_1 > 0$, therefore $p \geq 0$.

Acknowledgements

Thanks to Juris Hartmanis, Jon Kleinberg, and Lillian Lee for valuable comments. This work was supported in part by NSF grant CCR-0105586 and by ONR Grant N00014-01-1-0968. The views and conclusions contained herein are those of the author and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of these organizations or the US Government.

References

[1] John Horton Conway. *Regular Algebra and Finite Machines*. Chapman and Hall, London, 1971.