Open Problems Column
Edited by William Gasarch

This Issue's Column! This issue’s Open Problem Column is by Daniel Frishberg and William Gasarch. It is about Different Ways to Prove a Language is Not regular.

Request for Columns! I invite any reader who has knowledge of some area to contact me and arrange to write a column about open problems in that area. That area can be (1) broad or narrow or anywhere inbetween, and (2) really important or really unimportant or anywhere inbetween.

Different Ways to Prove a Language is Not Regular

By Daniel Frishberg and William Gasarch

1 Introduction

One semester when I (William Gasarch) was teaching Formal Language Theory a very bright math major was taking the class and said Why teach the pumping lemma when you an prove everything from the Myhill-Nerode Theorem? That statement might be correct mathematically though not pedagogically. However, it raises the question: there are many ways to prove languages not regular—how do they compares?

2 Reductions

Notation 2.1 Let $\Sigma$ be a finite alphabet, $\sigma \in \Sigma$, and $w \in \Sigma^*$. Then $\#_\sigma(w)$ is be the number of $\sigma$’s in $w$.

The following is a common exercise in a course in formal language theory.

1. Show that $X_1 = \{a^n b^n : n \in \mathbb{N}\}$ is not regular.
2. Show that $X_2 = \{w : \#_a(w) = \#_b(w)\}$ is not regular.
3. Show that $X_3 = \{w : \#_a(w) \neq \#_b(w)\}$ is not regular.

One can prove $X_1$ is not regular using the pumping lemma. One can prove $X_2$ is not regular either by using the pumping lemma (a version with bounds on the prefix) or by contradiction: if $X_2$ is regular than $X_2 \cap a^*b^* = X_1$ is regular. One cannot prove $X_3$ non-regular with the pumping theorem directly; however one can prove its regular by contradiction: if $X_3$ is regular than $\overline{X_3} = X_2$ is regular.

We will view the proofs by contradiction as reductions.

Def 2.2 Let $\Sigma$ be a finite alphabet.

1. For every regular $B \subseteq \Sigma^*$ let $f_B : 2^{\Sigma^*} \rightarrow 2^{\Sigma^*}$ be $f_B(A) = A \cap B$. Note that if $A$ is regular then $f_B(A)$ is regular. Let $FREG = \{g : (\exists B \text{ regular } g = f_B)\}$.
2. Let $COMP : 2^{\Sigma^*} \rightarrow 2^{\Sigma^*}$ be $COMP(A) = \overline{A}$. 

1
3. Let $RED = \{g_1 \circ g_2 \circ \cdots \circ g_k : (\forall i)[g_i \in FREG \text{ or } g_i = COMP]\}$

4. Let $X, Y \subseteq \Sigma^*$. $X \leq Y$ if there exists $h \in RED$ such that $h(Y) = X$.

**Example 2.3**

1. $X_1 \leq X_2$ via $h = f_{a^*b^*}$.
2. $X_2 \leq X_3$ via $h = COMP$.

The following theorem is easy and left to the reader.

**Theorem 2.4** If $A$ is not regular and $A \leq B$ then $B$ is not regular.

**Convention 2.5** When we have a technique to show languages are not regular we also include languages whose non-regularity is obtained by reduction. Hence we will say $X_3$ can be proven not-regular by the Pumping Lemma

We could expand the definition of $A \leq B$ by allowing more reductions based on other closure properties of regular languages. We have never found a case where we needed to do so. We have never even found a case where doing so made a proof of regularity easier.

### 3 The Pumping Lemma

There are many different pumping lemmas. We choose the most powerful one we know that is reasonable to present to a class of undergraduates.

**Theorem 3.1** If $L$ is regular than there exists $n_0$ such that, for all $w \in L$, for all prefixes $x'$ of $w$, there exists $x, y, z$ such that the following hold:

1. $w = x'xyz$
2. $|x| \leq n_0$
3. $y \neq e$
4. $(\forall i \geq 0)[xy^iz \in L]$.

As noted in Section 2 It is a standard exercise to show that $X_1, X_2, X_3$ are not regular using the pumping lemma. $\{a^{f(n)} : n \in \mathbb{N}\}$ is regular iff $f$ is a finite variant of a function of the form $f(n) = an + b$ where $a, b \in \mathbb{N}$.

Ehrenfeucht, et al [3] exhibit, for all languages $Z \subseteq \{1, 2\}^*$, a languages $L_Z$ (the mapping $Z$ goes to $L_Z$ is injective) such that $L_Z$ cannot be proven not regular by (an advanced version of) the Pumping Lemma. Since most of these $L_Z$ are not regular, this would seem show there are many non-regular languages that cannot be proven non-regular by the pumping lemma. However, in the appendix of this open problems column we show that $L_Z$ is regular iff $Z$ is regular, so this does not give an example.

The following candidates have been suggested; however, they can be proven non-regular by pumping and closure. We leave these proofs as an exercise.

1. $\{a^ib^j : i, j \text{ are relatively prime}\}$.
2. $\{xx^Rw : x, w \in \Sigma^* - \{e\}\}$. ($x^R$ is $x$ written backwards.)
4 Kolmogorov Complexity

**Def 4.1** The *Kolmogorov complexity of a string* $x$, denoted $KC(x)$, is the length of the shortest program that prints out $x$. For example, the $C(a^n) \leq \lg(n) + O(1)$ since the $n$ in binary takes $\lg(n)$ bits and the following program prints out $a^n$

For $i = 1$ to $n$ print($a$).

If you flip a coin $n$ times and record the heads and tails to obtain a string $x$ of length $n$ then the shortest program that prints $x$ is likely to be

print($x$).

Hence $C(x) = n + O(1)$.

For more on Kolmogorov complexity see the awesome book by Li and Vitanyi [7].

Li and Vitanyi have proven (see [6] or [7]) the following:

**Def 4.2** Let $L$ be a language. For all $x \in \Sigma^*$ let $L_x = \{y : xy \in L\}$.

**Theorem 4.3** *(The Li-Vitanyi Non-Regularit Theorem.)* Let $L$ be a language. The following are equivalent.

1. $L$ is regular.

2. For all $x$, if $y$ is the $n$th element of $L_x$ then $C(y) \leq C(n) + O(1)$.

We give four examples of showing languages non-regular using Theorem 4.3. They are all from Vitanyi and Li [7].

1) Let $f(i) : \mathbb{N} \to \mathbb{N}$ be any function such that $\lim \inf_{i \to \infty} f(i + 1) - f(i) = \infty$. Let $A$ be the image of $f$. Let $L_1 = \{1^i : i \in A\}$.

Assume $L_1$ is regular. Let $m$ be arbitrary but large. Let $i$ and $j$ be consecutive elements of $A$ such that $C(j - i) = \log(m) + O(1)$ (any nonconstant function will suffice). Let $x = 1^i$. The first $y$ (so $n = 1$ in Theorem 4.3) such that $xy \in L_1$ is $y = 1^{j-i}$. By Theorem 4.3.

$$C(y) = C(1^{j-i}) = C(j - i) + O(1) = \log(m) + O(1) \leq C(1) + O(1) = O(1).$$

Since $m$ is arbitrarily large this is a contradiction.

2) $L_2 = \{xx^R w : x, w \in \Sigma^* \setminus \{e\}\}$.

Assume $L_2$ is regular. Let $m$ be arbitrary but large. Let $x = (01)^m$ where $C(m) = \log m + O(1)$ (any nonconstant function will suffice). The first $y$ (so $n = 1$ in Theorem 4.3) such that $xy \in L_2$ is $y = (10)^m0$. By Theorem 4.3.

$$C(y) = C(10^m0) = C(m) + O(1) = \log(m) + O(1) \leq C(1) + O(1) = O(1).$$

Since $m$ is arbitrarily large this is a contradiction. This is a contradiction.
3) \( L_3 = \{0^i1^j : \gcd(i, j) = 1\} \).

Assume \( L_3 \) is regular. Let \( m \) be arbitrary but large. Let \( x = 0^{(p-1)!} \) where \( C(p) = \log m + O(1) \) (any nonconstant function will suffice). The first \( y \) (so \( n = 1 \) in Theorem 4.3) such that \( xy \in L_3 \) is \( y = 1^p \) By Theorem 4.3.

\[
C(y) = C(1^p) = C(p) + O(1) = \log(m) + O(1) \leq C(1) + O(1) = O(1).
\]

Since \( m \) is arbitrarily large this is a contradiction. This is a contradiction.

4) \( L_4 = \{ p : p \) is a prime expressed in binary \( \} \). We give two proofs

**Proof one:** If \( L_4 \) is regular then \( L_4 \cap 1^* \) is regular. This is the set of binary representations of primes of the form \( 2^n - 1 \). These are called Mersenne primes. It is well known that if \( 2^n - 1 \) is a Mersenne prime then \( n \) is prime. Hence the elements of \( L_4 \cap 1^* \) can be arbitrarily far apart. Hence they are a language of \( L_1 \)-type and is not regular.

**Proof two:** Assume \( L_4 \) is regular. Let \( p_i \) be the \( i \)th prime. Let \( m \) be arbitrary but large. Note that if \( x, y \in \{0, 1\}^* \) then \( xy \) is the number \( x2^{\left| y \right|} + y \).

We first present an approach that does not quite work. Let \( k \) be such that all numbers \( y \) larger than the first \( p_k \) have \( C(y) \geq \log m + O(1) \) (any nonconstant function will suffice). Let \( x \) be the binary representation of the product of the first \( k \) primes.

Let \( xy \) be prime. Then \( x2^{\left| y \right|} + y \) is a prime. Since \( x \) is the product of the first \( k \) primes \( y \) is not divisible by any of the first \( k \) primes. So it seems that \( y \) must be \( > p_k \). But that is not true. \( y = 1 \) could work. For example:

\[
2 \times 3 \times 5 \times 7 \times 11 = 2310 \text{ so } x = 100100000110
\]

\[
x1 = 100100000110:1 = 121441 \text{ which is prime.}
\]

Even if \( x1 \) is not prime, \( x01 \) could be prime. So we need to pre-plan what prime we want \( xy \) to be. The key is that we don’t want it to end in \( 0 \ast 1 \).

We now present the real proof. Let \( k \) be a number to be determined later. Let \( u \) be the binary representation of the product of the first \( k \) primes. **Claim:** There exists \( v \) such that \( u2^{\left| v \right|} + v \) is prime and \( v \) is not in \( 0^*1 \).

**Proof:** Consider the interval \( I = [u2^{\left| v \right|}, u2^{\left| v \right|} + (u2^{\left| v \right|})^{11/20}] \). Note that (1) \( u2^{\left| v \right|} \) in binary is \( u \) followed by \( |u| \) 0’s, and (2) \( u2^{\left| v \right|} + (u2^{\left| v \right|})^{11/20} \) in binary is \( u \) followed by some \( |u| \)-long sequence. Health-Brown and Iwaniec showed that, for all \( n \), there is a prime in \([n, n^{11/20}] \). The prime \( p \) in \( I \) is of the form \( u2^{\left| v \right|} + v \) where \( |v| = |u| \).

**End of Proof of Claim**

Let \( x \) be the binary representation of the product of the first \( k \) primes.

There is good new and bad news here:

1. Assume \( xy \) is a prime. Then \( x2^{\left| y \right|} + y \) is a prime. Since \( x \) is the product of the first \( k \) primes \( y \) is not divisible by any of the first \( k \) primes. Yeah!

2. \( y \) could be 1. Or 01. Or 001. Etc.

Let \( y \) be the first \( y \) (so \( n = 1 \) in Theorem 4.3) such that \( xy \in L_4 \). Then \( x2^{\left| y \right|} + y \) has to be prime. Since the first \( k \) primes divide \( x \), \( y \) has to have as a factor some prime that is not in the first \( k \) primes. Hence \( y \) is larger than any of the first \( k \) primes. Hence \( C(y) \geq \log m + O(1) \). By Theorem 4.3.

\[
C(y) \leq C(1) + O(1) = O(1).
\]
Since $m$ is arbitrarily large this is a contradiction.

Note that in the proofs that $L_1, L_2, L_3, L_4$ are not regular we did not need to use reductions.

5 The Myhill-Nerode Theorem

Def 5.1 Given $u, v \in \Sigma^*$, $u \equiv_R v$ if for all $w \in \Sigma^*$, $uw \in L$ iff $vw \in L$.

Easily, this is an equivalence relation.

Theorem 5.2 A language $L$ is regular iff $L$ is a finite union of $\equiv_R$ classes.

This theorem, known as the Myhill-Nerode theorem, is used to show that $X_1$ is not regular: If $i, j \geq 0, i \neq j$, then $a^i \not\equiv_R a^j$, because $a^ib^i \in X_1$, but $a^ib^j \notin X_1$. Therefore, there are $\omega$ distinct $\equiv_R$ classes, not finitely many. The same proof works for $X_2$. For $X_3$: $a^ib^i \in X_3$, but $a^ib^j \notin X_3$.

6 Monoids

Def 6.1 Given a language $L \subseteq \Sigma^*$ and words $u, v \in \Sigma^*$, define $u \equiv v$ if for all $x, y \in \Sigma^*$, $xuy \in L$ iff $xvy \in L$.

Note 6.2 $x \equiv y \Rightarrow x \equiv_R y$. One may verify that $\Sigma^*$ is a monoid under concatenation (with $\lambda$ as the identity), and that $\equiv$ is a congruence.

Def 6.3 Given $L \subseteq \Sigma^*$, let $M = \{[u] \mid u \in \Sigma^*\}$. Call the quotient monoid $\Sigma^*/L$, via the semigroup homomorphism $\phi(u) = [u]$, the syntactic monoid, and denote it as $M(L)$.

It is known that:

Theorem 6.4 If $L$ is a language, then $L$ is regular iff its syntactic monoid is finite.

Since the elements of $M(L)$ are precisely the $\equiv$ classes, this is identical to the statement of the Myhill-Nerode theorem (except that the latter uses only equivalence on the right).

7 Communication Complexity

The techniques in this paper are essentially due to Birget [1] and Galister and Shallit [4].

Def 7.1 Let $A \subseteq \{0, 1\}^n \times \{0, 1\}^n$. Imagine that Alice has $x \in \{0, 1\}^n$ and Bob has $y \in \{0, 1\}^n$. They want to determine if $(x, y) \in A$. The Communication Complexity of $A$ is the minimum number of bits they need to communicate in order for them both to know if $(x, y) \in A$.

Let

$EQ = \{(x, x) \in \{0, 1\}^n \times \{0, 1\}^n\}$,

$MAJ = \{(x, y) \in \{0, 1\}^n \times \{0, 1\}^n : \#1(xy) \geq n/2\}$

$EQL = \{(x, y) \in \{0, 1\}^n \times \{0, 1\}^n : \#_0(xy) = \#_1(xy)\}$

The following are well known.
Theorem 7.2 \(D(EQ) = n + 1, \ D(MAJ) = \log(n) + O(1), \ D(EQL) = \log(n) + 1.\)

Theorem 7.3 Let \(L\) be a language. Let \(n \in \mathbb{N}\). Let
\[L_n = \{(x, y) : |x| = |y| = n \land xy \in L\}.
If \(D(L_n)\) is not constant than \(L\) is not regular.

Proof: We show that if \(D(L_n)\) is regular via DFA \(M\) then \(D(L_n)\) is constant. Alice has \(x\), Bob has \(y\). Alice runs \(M(x)\) and sends the resulting state \(q\) to Bob (this is a constant number of bits). Bob then takes the state \(q\) and runs \(M\) from there with \(y\). If the final result is (is not) an accept state then \(xy \in L\) (\(xy \notin L\)), Bob knows this, and sends Alice a 1 (0).

Theorem 7.3 can be used to show that (1) \(X_2\) is not regular since \(D(EQ)\) is not constant, and (2) \(\{w : \#_a(w) \geq \#_b(w)\}\) is not regular since \(D(MAJ)\) is not constant. But what about \(X_1\)? Here \(L_n = \{a^{n/2}b^{n/2}\}\) which does have \(D(L_n) = O(1)\). So we cannot use Comm Comp directly. We can use it a different way.

The following proof is due to Narad Rampersad and Marzio De Biasi independently (they both left comments on my blog post of October 16, Boss’s day!).

Theorem 7.4 If \(X_1\) is regular then \(D(EQ) = O(1)\). Hence \(X_1\) is not regular.

Proof: Assume \(X_1\) is regular via DFA \(M\). We give the protocol that shows \(D(EQL) = O(1)\).

1. Alice gets \(x \in \{0, 1\} n\), Bob gets \(y \in \{0, 1\} n\). They want to determine if \(#_0(xy) = #_1(xy)\).
2. Let \(s\) be the start state of \(M\). Alice runs \(M(s, 0#_0(x))\) and ends up at state \(p\). Alice sends \(p\) to Bob.
3. Bob runs \(M(p, 0#_0(y)1#_1(y))\) and ends up in state \(q\). Bob sends \(q\) to Alice.
4. Alice runs \(M(q, 1#_1(x))\) and ends up in state \(r\). If \(r\) is an accept state then transmit to bob YEAH! If \(r\) is a reject stat then transmit to bob BOO!

8 E-F Games

We define a set of formulas and their interpretations. They are interpreted over a string \(w \in \Sigma^*\). The first order quantifiers will range over positions in the string. The second order quantifiers will range over sets of positions in the string.

Def 8.1

1. Terms are used to refer to positions in the word. A term is (1) an expression of the form \(x + 1\) where \(x\) is a variable. (2) \(F, \ F + 1 \ (L, \ L + 1)\). This is the index of the first (last) symbol in the word, and the next one. Note that \(x + 2, \ F + 2, \ L + 2\) are not terms.
2. Let \( t_1, t_2, t \) be terms, \( \sigma \in \Sigma \), and \( X \) be a second order variable. The following are atomic formulas:

(a) \( t_1 = t_2 + 1 \). This conveys the obvious meaning.
(b) \( t \in X \). This conveys the obvious meaning.
(c) \( Q_\sigma(t) \). This is interpreted as saying the \( t \)th letter in \( w \) is \( \sigma \).
(d) \( \text{PART}_k(X_0, \ldots, X_k) \). The meaning is that \( X_0, \ldots, X_k \) are a partition of the indices of the word.

3. A formula \( \phi \) is defined recursively:

(a) Any atomic formula is a formula.
(b) If \( \phi_1, \phi_2, \phi \) are formulas then \( \phi_1 \land \phi_2, \phi_1 \lor \phi_2 \), and \( \neg \phi \) are formulas.
(c) If \( \phi(x) \) is a formula with a free variable \( x \) (either first or second order, and there could be other free variables as well) then \( (\exists x)[\phi(x)] \) and \( (\forall x)[\phi(x)] \) are formulas.

4. A sentence is a formula with no free variables. Note that if \( \phi \) is a sentence and \( w \in \Sigma^* \) then \( \phi \) is either true or false of \( w \).

5. If \( w \in \Sigma^* \) and \( \phi \) is a sentence then \( w \models \phi \) means that \( \phi \) is true when interpreted over \( w \).

6. Let \( n \) be an integer, and let \( m = (m_1, \ldots, m_k) \) be a sequence of integers. A formula \( \phi \) is in \( \Sigma_{n,m} \) if the prefix of \( \phi \) is the formula \( \text{PART}(X_0, \ldots, X_n) \), followed by \( k \) alternating blocks of first-order quantifiers (starting with either \( \exists \) or \( \forall \)). (This is not the standard use of \( \Sigma \) in logic, but it is close.)

7. \( L \in \Sigma_{n,m} \) if there is a sentence \( \phi \in \Sigma_{n,m} \) such that \( L = \{w : w \models \phi\} \).

**Note 8.2** For simplicity, our language does not include = or <. With additional first- and second-order quantifiers, these can both be derived from \( t_1 = t_2 + 1 \) and \( t \in X \).

The following is essentially due to Büchi [2] (see also [8])

**Theorem 8.3** A language \( L \subseteq \Sigma^* \) is regular iff \( L \in \Sigma_{n,(1)} \). The sentence defining \( L \) is of the form

\[
(\exists X_0) \cdots (\exists X_k)(\forall x)[\text{PART}(X_0, \ldots, X_k) \land \psi(X_0, \ldots, X_k, x)].
\]

**Example 8.4**

1. \( \Sigma = \{a, b\} \). Let \( L = \{w : \#_a(w) \equiv 0 \pmod{3}\} \). If \( \phi \) is as below then \( L = \{w : w \models \phi\} \).

\[
(\exists X_0, X_1, X_2)(\forall x) [ Q_a(F) \rightarrow F + 1 \in X_1 \land \neg Q_b(F) \rightarrow F + 1 \in X_0 \\
((x \in X_0 \land Q_a(x + 1)) \rightarrow x + 1 \in X_1) \land ((x \in X_0 \land Q_b(x + 1)) \rightarrow x + 1 \in X_0) \land \\
((x \in X_1 \land Q_a(x + 1)) \rightarrow x + 1 \in X_2) \land ((x \in X_1 \land Q_b(x + 1)) \rightarrow x + 1 \in X_1) \land \\
((x \in X_2 \land Q_a(x + 1)) \rightarrow x + 1 \in X_0) \land ((x \in X_2 \land Q_b(x + 1)) \rightarrow x + 1 \in X_2) 
\]
2. Let the alphabet be \{a, b\}. Consider all \(B(\Sigma_{n,(1)})\) sentences. They have 0 second order variables and 2 first order variables. Informally, all they can express is the presence or absence of various combinations of \(a, b, aa, ab, ba, bb\) (and in particular, if all \(a\) are followed by an \(a\) (or \(b\)), and if all \(b\) are followed by an \(a\) (or \(b\))). Hence if two strings agree on all of those properties they cannot be distinguished by a \(\Sigma_{n,(1)}\) sentence. Therefore the strings

\[aaabbbaaa, aaabbbbaaa\]

satisfy the same \(\Sigma_{(0),(1)}\) sentences.

Ehrenfeucht-Fraïssé games are a way to show that a set of structures is not definable by a particular logical language. We adapt a version of such games, based on work of Ladner [5] and Thomas [9], to show that a set of strings is not regular.

The intuition behind the game is that there are two strings \(u \neq v\). Spoiler wants to prove to Duplicator (henceforth Dup) that these strings are different. Spoiler chooses a subset of positions in \(u\) (or \(v\)) or a position in \(u\) (or \(v\)) and in effect challenges Dup to come up with a subset of positions or a position in the other string that is analogous.

We define two notions of strings being equivalent and later state that these notions are equivalent. One involves truth; one involves games.

**Notation 8.5** For \(u, v \in \Sigma^*\), if for all \(\phi \in \Sigma_{n,m}\), \(u \models \phi\) iff \(v \models \phi\), then write \(u \approx_{n,m,T} v\). (T stands for Truth.)

**Def 8.6** Let \(G_{n,m}(u, v)\) be the following game played by Spoiler and Dup.

1. Set up: There are two strings \(u, v \in \Sigma^*\). \(n \in \mathbb{N}\) and \(m = (m_1, \ldots, m_k)\).

2. Spoiler \(n\)-colors the positions in \(u\) (or \(v\)) which we express as a partition denoted \((S_{u,1}, \ldots, S_{u,n})\) (denoted \((S_{v,1}, \ldots, S_{v,n})\)). Dup \(n\)-colors the positions in \(v\) (or \(u\)), denoted \((S_{v,1}, \ldots, S_{v,n})\) (denoted \((S_{u,1}, \ldots, S_{u,n})\)). (Dup must \(n\)-color the string that Spoiler does not.)

3. For \(1 \leq i \leq k\), Spoiler chooses a position in \(u\) (or \(v\)) denoted \(i_u\) (denoted \(i_v\)). Dup chooses a position in \(v\) (or \(u\)) denoted \(i_v\) (denoted \(i_u\)).

4. At the end we have two tuples \((S_{u,1}, \ldots, S_{u,n}, 1_u, \ldots, k_u)\) and \((S_{v,1}, \ldots, S_{v,n}, 1_v, \ldots, k_v)\). Dup wins if the following hold

   (a) For all \(1 \leq i \leq k\) \(i_u = F\) iff \(i_v = F\). \(i_u = L\) iff \(i_v = L\).
   (b) For all \(1 \leq i \leq k\), for all \(\sigma \in \Sigma\), \(Q_\sigma(i_u) = Q_\sigma(i_v)\) and \(Q_\sigma(i_u + 1) = Q_\sigma(i_v + 1)\) (or they both do not exist).
   (c) For all \(1 \leq i, j \leq k\) \(i_u = j_u + 1\) iff \(i_v = j_v + 1\).
   (d) For all \(1 \leq i \leq k\), \(1 \leq j \leq n\) \(i_u \in S_{u,j}\) iff \(i_v \in S_{v,j}\).
   (e) Dup wins \(G_{n,m}(u, v)\) means that Dup has a winning strategy in that game. Similar for Spoiler.
   (f) If Dup wins \(G_{n,m}(u, v)\) then we write \(u \approx_{n,m,G} v\). (G stands for Game.)
Theorem 8.7

1. For all \( n, m \in \mathbb{N} \) for all \( u, v \in \Sigma^* \), \( u \approx_{n,m,T} v \) iff \( u \approx_{n,m,G} v \).

2. Let \( L \subseteq \Sigma^* \). Assume that, for all \( n \), there exists \( u, v \) with \( u \in L \) and \( v \notin L \) such that Dup wins \( G_{n,(1)}(u,v) \). Then \( L \) is not regular. This follows from Part 1 and Theorem 8.3.

Example 8.8 Let \( u = a^9 \) and \( v = a^7 \). Consider the game \( G_{n,m}(u,v) \), where \( n = 3, m = (1) \). Certainly \( u \not\approx_{n,m,T} v \), as the sentence from Example 8.4.2 shows. We use this formula to guide Spoiler to victory. In the case where Spoiler plays first on \( u \), we examine what Dup can do and how Spoiler can then counter it.

1. On the first set move, Spoiler colors \( u \) via \( RWBRWBRW \)

2. Clearly Dup has to color \( v \) beginning \( RW \). Since \( W \) is always followed by \( B \), the next color has to be \( B \). Keep going this way and we have that \( v \) must be colored \( RWBRWBR \). But then the colors of the \( L \)'s differ and Spoiler wins.

The example points to the following definition and lemma.

Def 8.9 If \( COL \) is a \( k \)-coloring of \( u \in \Sigma^* \) then the induced coloring is the coloring \( COL'(i) = (COL(i), u_i) \). We will refer to the induced colored strings as \( u', v' \).

Def 8.10 Let \( u, v \in \Sigma^* \). Assume that \( u \) and \( v \) have been \( k \)-colored. Let \( u', v' \) be the induced \( |\Sigma|k \)-colorings. Let \( u' \approx_2 v' \) if \( u' \) and \( v' \) share a prefix and suffix of length 2, and for every substring \( w \) of one word, if \( |w| \leq 2 \), then \( w \) occurs in the other word. (This definition can be applied to any strings and we can also define \( \approx_3 \), etc; however, we do not do that so we can cut down on notation.)

Lemma 8.11 For all \( n > 0 \), let \( k = 2n^2, l = k + 2, i = (k!)lnl, j = i + k! \). Then, for every word \( w_1 \in \Sigma^i \), where \( n = |\Sigma| \), there exists a word \( w_2 \in \Sigma^j \) such that \( w_1 \approx_2 w_2 \). In particular, there exist \( x, v, u, y \in \Sigma^* \) such that \( w_2 = xv'uy \) for some \( r > 0 \).

Proof: \( n^2 \) is the number of distinct words of length 2. Let \( k = 2n^2, l = k!, i = (k!)lnl, j = i + k! \). Let \( |w_1| = i \). In \( w_1 \), some subword \( |u| = 2 \) must occur more than once, since \( i > 2n^2 \). I.e., \( w_1 = xuzuy \), for some \( x, y, u, z \). Let \( v = uz, w_2 = x(uz)'uy = xv'uy \), for any \( r > 0 \). Every sequence of length 2 in \( w_1 \) occurs in either \( x, u, z, y \), or at the boundary of two or more of these. If it occurs within \( x, y, u, \) or \( z \), it also occurs in \( w_2 \) since \( x, y, u, z \) occur in \( w_2 \). If it occurs at the boundary of \( x \) and \( u \), or of \( u \) and \( y \) or \( u \) and \( z \), it will occur at the same boundary in \( w_2 \). Similar reasoning shows that every such sequence in \( w_2 \) occurs in \( w_1 \), and that the prefixes and suffixes of length 2 are identical.

Lastly, \( |v| = k! = j - i \), since \( |v| \leq k \). Let \( r = \frac{k!}{|v|} + 1 \). Then \( |w_2| = j \), and \( w_1 \approx_2 w_2 \).

Before proving our main result, we illustrate the mechanism in Lemma 8.11 with an example.
**Example 8.12** Let $\Sigma = \{R, B\}$. Let $n = |\Sigma| = 2$. For this example, we can actually do a little better than the extremely large values for $i$ and $j$. Let $i = 10, j = 34 = i + 24$. (For all $1 \leq m \leq 4, 2m|24$.) Let $w_1 = xuzu = (BB)(RR)(RBBR)(RR)$. We can “pump” the substring $uzu$ to obtain $w_2 = x(uz)^5u = (BB)(RBBR)(RR)^5$, which is of length $34$.

Note that we could have chosen any $|w_1| = 10$, and we could have found a substring to pump, of length $2, 4, 6, \text{or } 8$. These all divide $24$, so we would always be able to produce $w_2 \approx w_1$.

**Lemma 8.13** Let $n = |\Sigma|, i, k, l, j$ be as in Lemma 8.11. Then given $w_2 \in \Sigma^j$, there exists $w_1 \in \Sigma^i$ such that $w_1 \approx w_2$.

**Proof:** First, note that $n^l = n^{2n^2 + 2}$ is the number of all possible strings of length $2n^2 + 2$ over $\Sigma$. Thus if a word $|w_2| = k\ln n^i + k!$, then at least one string of length $l = 2n^2 + 2$ appears at least $(2n^2)!$ times in $w_2$. Let $|w_1| = i = k!(n^i)$, and recall that $|w_2| = |w_1| = j - i = (2n^2)!$. Then recall that any string $s$ of length $2n^2 + 2$, is of the form $xuzu$, where $|u| = 2$ and $|z| > 0$. Note that if some $s = xuzyu$ occurs in $w_2$ more than once, we may in every occurrence but one replace $s$ with $t = xuy$: in this way, the only substrings of length 2 deleted from $w_2$ occur in $z$ or at the boundary of $u$ and $z$. But these certainly appear in the one remaining occurrence of $s$. Thus, since a substring $s$ of length $2n^2 + 2$ must occur at least $(2n^2)!$ times in $w_2$, we can replace $uzu$ with $u$ in any number of occurrences of $s$ from 1 to $(2n^2)! - 1$. Since $2 \leq |zu| \leq 2n^2$, there exists some integer $1 \leq q \leq (2n^2)!/2 < (2n^2)! - 1$ such that $q|zu| = (2n^2)!$. Therefore, we may cut $q$ occurrences of $|zu|$, obtaining $w_1 \approx w_2$. 

**Example 8.14** We illustrate the mechanism in Lemma 8.13. To simplify our illustration (and use smaller words), we take the liberty of choosing a particular $w_2$. Let $i = 26, j = 50$. Let $w_2 = ((BB)(RR)(RBBR)(RR)^5).

Cut four occurrences of $(RR)(RBBR)$ to obtain $w_2 = ((BB)(RR))45((BB)(RR))(RBBR)(RR)) \approx w_1$.

**Lemma 8.15** If $u \approx_{n,m,G} v$, then for all $w$, $uw \approx_{n,m,G} vw$.

**Note 8.16** This lemma is known; its proof is simply to combine Dup’s strategy in the E-F games on $u, v$ and on $u, w$.

**Lemma 8.17** Let $w_1 \approx w_2$. Dup has a winning strategy in $G_{0,(1)}(w_1, w_2)$.

**Proof:** Let Spoiler select position $s$ in either word. Since $w_1 \approx w_2$, Dup can select $t$ in the other word s.t. $s = L (s = F)$ iff $t = L (t = F)$, and for all $\sigma \in \Sigma$, $Q_\sigma(s) = Q_\sigma(t)$, and, if $s \neq L$, then $Q_\sigma(s + 1) = Q_\sigma(t + 1)$. I.e., Dup wins.

**Theorem 8.18** $X_1$ is not regular.
Proof: For all $n > 0$, let $i$ and $j$ be as in Lemma 8.11. Let $\Sigma = \{a, b\}$, and let $w_1 = a^i$, $w_2 = a^j$. Let $G_{n,(1)}$ be played on $w_1, w_2$. Suppose that in the first move, Spoiler $n$-colors $w_1$, inducing the colored string $w'_1$ over the induced alphabet $\Sigma'$. By Lemma 8.11, we may pump some substring of $w'_1$, obtaining a string $w'_2 \approx 2w'_1$, with $|w'_2| = j = |w_2|$. Let Dup color $w_2$ so as to induce $w'_2$.

Now suppose that in the first move, Spoiler $n$-colors $w_2$, inducing $w'_2 \in \Sigma'$. By Lemma 8.13, we may cut substrings of $w'_2$ to obtain $w'_1 \approx 2w'_2$, with $|w'_1| = i = |w_1|$. Let Dup induce $w'_1$.

Now the remainder of the game is equivalent to the game $G_{n,(1)}(w'_1, w'_2)$. By Lemma 8.17, Dup has a winning strategy in this game, so Dup has a winning strategy in $G_{n,(1)}(w_1, w_2)$. Therefore $a^i \approx_{n,m,G} a^j$. By Lemma 8.15, $a^ib^i \approx_{n,m,T} a^ib^i$. Therefore, by Theorem 8.7, $a^ib^i \approx_{n,m,T} a^ib^i$. But $a^ib^i \in X_1$ and $a^ib^i \notin X_1$. Therefore $X_1 \notin \Sigma_{n,(1)}$, and by Theorem 8.3, $X_1$ is not regular.

9 Compare and Contrast

We have presented several techniques to prove a language is not regular. How do they compare?

Open Problem:

1. For each technique $T$ above determine if there is a non-regular language that cannot be shown non-regular using that technique (and reductions).

2. For all ordered pairs of techniques $(T_1, T_2)$ determine if there is a non-regular language that can be shown non-regular using $T_1$ but not $T_2$.

3. For each technique $T$ define a notion of length-of-proof-of-non-regularity. Let $LEN_T(L)$ be the length of the shortest proof that $L$ is not regular using technique $T$. For all ordered pairs of techniques $(T_1, T_2)$ determine if there is a family of non-regular language $\{L_n\}_{n=1}^\infty$ such that $LEN_{T_1}(L) \ll LEN_{T_2}(L)$.

A Showing The Ehrenfeucht-Parikh-Rozenberg Language Not Regular by Closure

Ehrenfeucht, et al [3] exhibit, for all languages $Z \subseteq \{1, 2\}^*$ a languages $L_Z$ (the mapping $Z$ goes to $L_Z$ is injective) such that $L_Z$ cannot be proven not regular by the Pumping Lemma (they show this for a rather advanced version of the pumping lemma). Since most of these $L_Z$ are not regular, this would seem show there are many non-regular languages that cannot be proven non-regular by the pumping lemma. In this note we show that, using closure properties and a simple form of the pumping lemma, the languages $L_Z$ that are non-regular can be proven to be non-regular.

Notation A.1

$\Sigma$ is the 16-letter alphabet $\{(i, j) : 0 \leq i, j \leq 3\}$.

$f_1 : \Sigma \rightarrow \Sigma$ is defined by

$$f_1((i, j)) = (i + 1 \pmod{4}, j)$$

$f_2 : \Sigma \rightarrow \Sigma$ be defined by

$$f_2((i, j)) = (i, j + 1 \pmod{4})$$
Note that $f_1(f_2(\sigma)) \neq f_2(f_1(\sigma))$.

**Def A.2** A string $x$ is **legal** if

1. $x = (\sigma_1)^{n_1}(\sigma_2)^{n_2}\cdots(\sigma_m)^{n_m}$ where $n_1, n_2, \ldots, n_m \geq 1$.
2. $\sigma_1 = (0, 0)$.
3. For all $2 \leq i \leq m$, either $\sigma_i = f_1(\sigma_{i-1})$ or $\sigma_i = f_2(\sigma_{i-1})$.

**Example:**

$(0, 0)(1, 0)(1, 0)(1, 0)(2, 0)(2, 1)(3, 1)(0, 1)$

We associate to every legal string the sequence of transitions that cause $\sigma_i$ to go to $\sigma_{i+1}$, called the code string. Note that above:

- $f_1(0, 0) = (1, 0)$
- $f_1(1, 0) = (2, 0)$
- $f_2(2, 0) = (2, 1)$
- $f_1(2, 1) = (3, 1)$
- $f_1(3, 1) = (0, 1)$.

So we associate code string $11211$.

Let's go in the other direction: We give legal strings with code string $11211$:

$(0, 0)\{(1, 0)\}^{\geq 1}\{(2, 0)\}^{\geq 1}\{(2, 1)\}^{\geq 1}\{(3, 1)\}^{\geq 1}\{(0, 1)\}^{\geq 1}$

**Def A.3** Let $x \in \Sigma^*$. The **parity of** $x$ is the parity of the sum of all of the components of $x$.

**Example:** The parity of

$(0, 0)(1, 0)(1, 0)(1, 0)(2, 0)(2, 1)(3, 1)(0, 1)$

is

$$0 + 0 + 1 + 0 + 1 + 0 + 1 + 0 + 2 + 0 + 2 + 1 + 3 + 1 + 0 + 1 \pmod{2} = 1.$$  

**Def A.4** Let $Z \subseteq \{1, 2\}^*$. Let

$L_Z = \{x : x \text{ is legal and } (\exists z \in Z)[x \text{ has code strings } z]\} \cup \{x : x \text{ is not legal and } \text{parity}(x)=0\}$

We leave the following easy theorem to the reader.

**Theorem A.5** If $Z$ is regular than $L_Z$ is regular.

Ehrenfeucht, et al [3] prove that, for all $Z$, $L_Z$ cannot be proven non-regular using the pumping lemma. Since there are an uncountable number of $Z$, and each $Z$ gives a different $L_Z$, there are an uncountable number of non-regular languages that cannot be proven not-regular by the pumping lemma.

We use closure properties to show that if $L_Z$ is regular than $Z$ is regular.
**Def A.6** Let $\Sigma_1$ and $\Sigma_2$ be finite alphabets. Let $F : \Sigma_1 \times \Sigma_1 \to \Sigma_2$. We extend $F$, first to $\Sigma_1^*$, second to all subsets of $\Sigma_1^*$.

1. Let $F : \Sigma_1^* \to \Sigma_2^*$ be defined by

\[ F(\sigma_1\sigma_2\sigma_3\cdots\sigma_n) = f(\sigma_1\sigma_2)f(\sigma_2\sigma_3)\cdots f(\sigma_{n-2}\sigma_{n-1})f(\sigma_{n-1}\sigma_n). \]

2. Let $F : 2^{\Sigma_1^*} \to 2^{\Sigma_2^*}$ be defined by

\[ F(L) = \{ f(x) : x \in L \}. \]

**Lemma A.7** Let $\Sigma_1$ and $\Sigma_2$ be finite alphabets. Let $f : \Sigma_1 \times \Sigma_1 \to \Sigma_2$. Let $F$ be as in definition A.6. Let $L \subset \Sigma_1^*$ such that If $L$ is regular then $F(L)$ is regular.

**Theorem A.8** Let $Z \subseteq \{0,1\}^*$. If $L_Z$ is regular then $Z$ is regular.

**Proof:** Assume $L = L_Z$ is regular. Note that

\[ PAR_1 = \{ x : x \text{ has parity 1} \} \]

is regular. Hence

\[ L' = L \cap PAR_1 = \{ x : x \text{ is legal and } x \text{ has parity 1 and } (\exists z \in Z)[x \text{ has code strings } z] \} \]

is regular. Let

\[ NOD = \{ x = \sigma_1 \cdots \sigma_n : (\forall i \leq n-1)[\sigma_i \neq \sigma_{i+1}] \} \]

($NOD$ stands for NO Doubles.)

Note that $NOD$ is regular. Hence $L' \cap NOD$ is regular. If $x \in L' \cap NOD$ then the following hold:

1. $x = \sigma_1\sigma_2\cdots\sigma_m$ where, for all $1 \leq i \leq m-1$, $\sigma_i \neq \sigma_{i+1}$.
2. $\sigma_1 = (0,0)$.
3. For all $2 \leq i \leq m$, either $\sigma_i = f_1(\sigma_{i-1})$ or $\sigma_i = f_2(\sigma_{i-1})$.
4. $x$ has parity 1.
5. $x$ codes $z$.

One can easily construct a DFA for $Z$ from a DFA for $L' \cap NOD$. Hence $Z$ is regular.

\[ \blacksquare \]
References


