A formalization of block pumpable language theory in Coq

Elaine Li

Yale-NUS College
Advised by Aquinas Hobor, Frank Stephan

Wednesday, 3rd April 2019

Roadmap

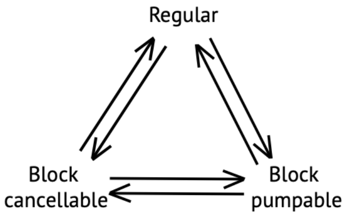
1. Motivation
2. Background
   - Formal languages
   - Regular languages and their properties
3. A proof nugget
   - Informal proof
   - Formal proof
   - Proof in numbers
4. Conclusion

Motivation

An observation:

- Formal languages are mathematical objects that enjoy a diversity of representations.
- Representations are costly in formal, mechanized proofs.

Goal: a illustrative, self-contained formalization of results related to block pumping in Coq that includes:



- Closure properties of positively block pumpable languages under union, intersection and concatenation.

Formal languages: an overview

1. Definition: a set of **strings** over an **alphabet** with a **mechanism to decide membership**
   - alphabet Σ: a finite set of symbols
     e.g. [a-z], $\{0,1,2\}$
   - string/word: concatenation of symbols
     e.g. "meringue", 00101
   - mechanisms to decide membership:
     - via checking, i.e. abstract machines
     - via generating, i.e. derivation rules
2. Applications: compiler design, formal linguistics, text processing, model checking etc.

Regular languages: a class of formal languages

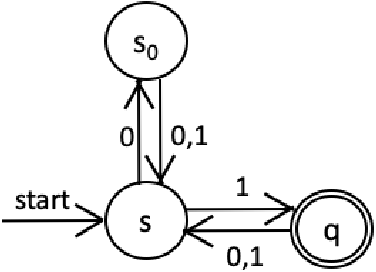
Regular languages are a class of formal languages recognizable by **finite automata**.

Regular languages: a class of formal languages

A language is regular iff there exists a **finite automaton** that accepts all words in the language, and rejects all words not in the language.

An example

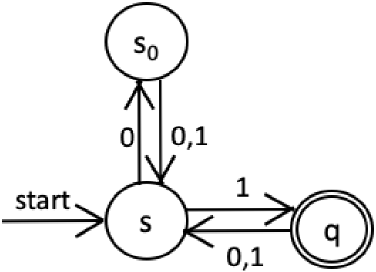
Let $\Sigma = \{0, 1\}$ and
$L = \{w : w$ is of odd length and ends in $1\}$.



$w_1 = 10$
$w_2 = 00101$

An example

Let $\Sigma = \{0, 1\}$ and
$L = \{w : w \text{ is of odd length and ends in } 1\}$.



$w_2 = 00\textcolor{red}{10}1$
$w_3 = 001$
$w_4 = 00\textcolor{red}{10101010}1$

Regular languages: pumping properties

Regular languages admit **pumping properties**.

Regular languages: pumping properties

Do pumping properties **characterize** regular languages?

The quest to find a necessary and sufficient pumping condition

Ehrenfeucht, Parikh and Rozenberg (1981)

The block pumping property

L $\subseteq \Sigma^*$ has the block pumping property iff there exists a k such that for all w $\in \Sigma^*$ and all ways of inserting k breakpoints into the word, there exist two breakpoints such that the word part in between them can be *repeated* or *omitted* without affecting word membership.

Mr. Pumping Lemma

Given a language $L \subseteq \Sigma^\star$,

1. You pick a pumping constant k.
2. Mr. Pumping Lemma picks a word w, and a set of breakpoints bps of size k.
3. You pick two breakpoints bp_1, bp_2 from the breakpoint set bps.
4. Mr. Pumping Lemma pumps the word part between bp_1 and bp_2 any number of times.

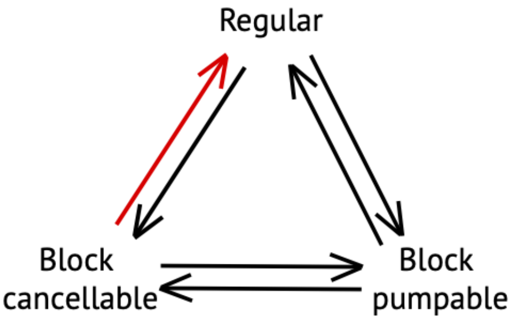
If the membership of the resulting word w' remains the same as the membership of the original word w in L, then you win. Otherwise, Mr. Pumping Lemma wins.

EPR's Theorem

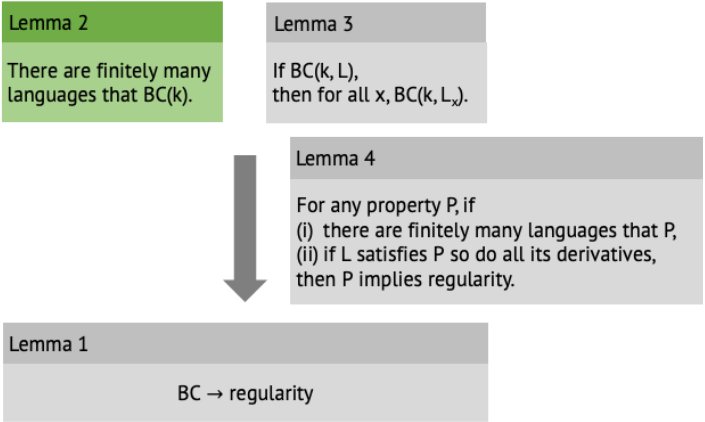
Theorem of Ehrenfeucht, Parikh and Rozenberg

Regularity, the block pumping property, and the block cancellation property are equivalent.

EPR's Theorem



EPR's proof, in pictures



EPR's proof, in pictures

Lemma 2

There are finitely many languages that BC(k).

EPR's proof, in pictures

Lemma 2

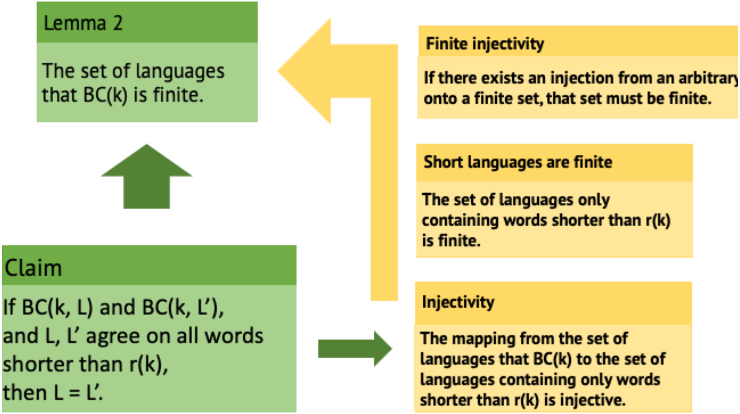
There are finitely many languages that BC(k).

↑

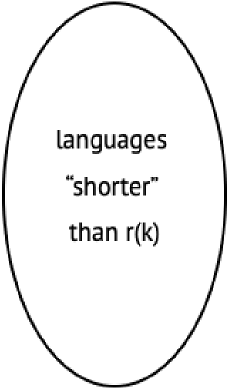
Claim

If BC(k, L) and BC(k, L'), and L, L' agree on all words shorter than r(k), then L = L'.

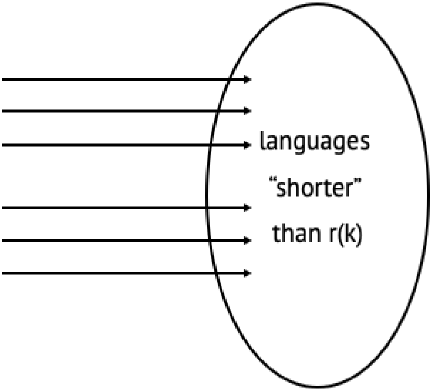
Formal proof, in pictures



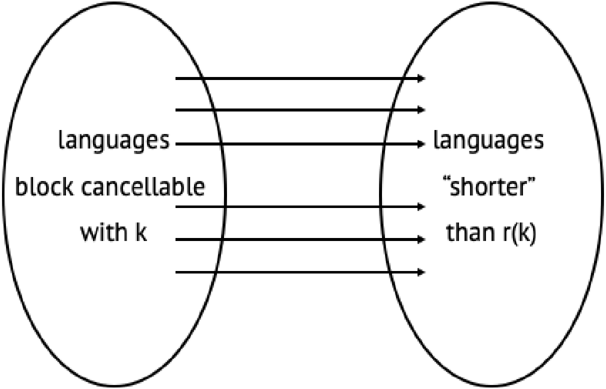
Picture view



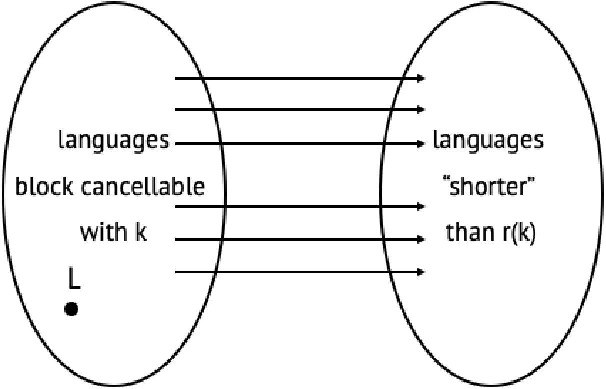
Picture view



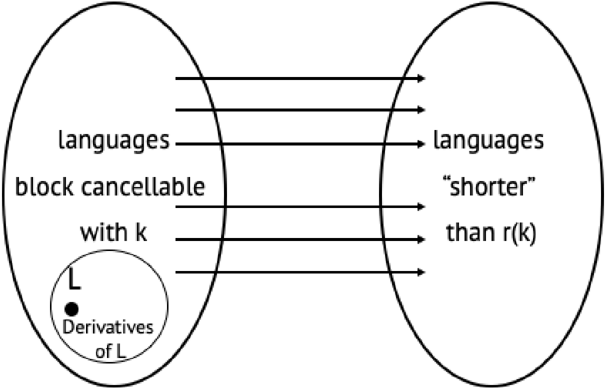
Picture view



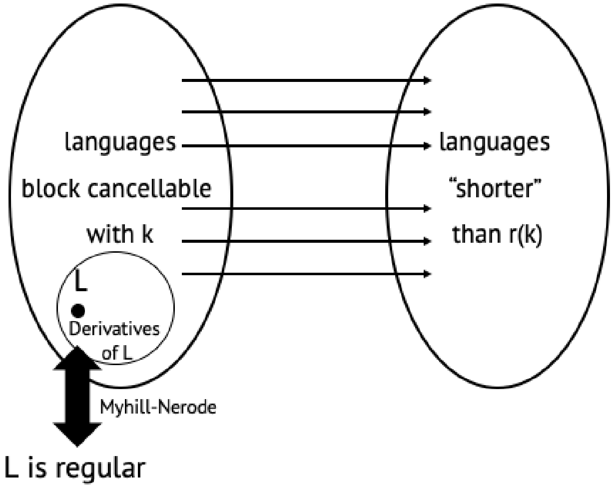
Picture view



Picture view



Picture view



Finite injectivity, in Coq

Theorem (Finite injectivity)

```
inj_finite
 : forall (P Q : X -> Prop)
          (f : {x : X | P x} -> {x : X | Q x}),
        inhabited {x : X | P x} ->
        injective P Q f ->
        is_finite_dep Q ->
        is_finite_dep P.
```

where:

- `X` - language
- `P` - property of being block cancellable with k
- `Q` - property of only containing words shorter than r(k)
- `f` - mapping from BC(k) languages to short languages

Finiteness, in Coq

Definition (Dependent finite)

```
is_finite_dep {X : Type}
              (P : X -> Prop) : Prop :=
  exists (L : list {x | P x}),
  forall (x : {x | P x}), In x L.
```

Definition (Finite)

```
is_finite {X : Type}
          (P : X -> Prop) :=
  exists (L : list X), forall (x : X),
  In x L <-> P x.
```

A comparison

	EPR proof	Coq proof
Definitions	22	400
Lemma 2	26	800
Lemma 3	5	160
Lemma 4	14	20

```
lib.v 1000
finite.v 400
triangle.v 1000
closure.v 650
```

Conclusion

One question: how to characterize the expressive power of dependent types in Coq's type theory?
One takeaway: automata theory through the lens of functional manipulation of inductive data types

Formal definitions : Injective

Definition (Injective)

```
injective {X Y : Type}
          (P : X -> Prop) (Q : Y -> Prop)
          (f : {x | P x}  -> {y | Q y}) :=
  forall (x1 x2 : {x | P x}),
  f x1 = f x2 -> x1 = x2.
```

Two views of propositional information

"All numbers greater than 2 are greater than 1."

On the set-theoretic view:

1. For all $n \in \mathbb{N}$, if $n > 2$ then $n > 1$.
2. For all $n \in \{n : n > 2\}$, $n > 1$.

On the type-theoretic view:

1. $\forall\ n : \texttt{nat}, n > 2 \rightarrow n > 1$.
2. $\forall\ n : \{n : \texttt{nat} \mid n > 2\}, n > 1$.

Dependent types in Coq

Dependent types are types that **carry propositional information**.

Definition (Dependent types)

```
Inductive sig (A : Type)
              (P : A -> Prop) : Type :=
    exist : forall x : A, P x -> {x : A | P x}.
```

Ramsey's Theorem

For all $k \in \mathbb{N}$ and finite set Q of colors, there exists an $r(k) \in \mathbb{N}$ such that for every ordered set I of size $r(k)$ and every coloring function C mapping ordered pairs $i, j \in I$ to a color $C(i, j) \in Q$, there exists a monochromatic subset of I of size k.

EPR's Proof

To prove:

Lemma 2

There are only finitely many languages block cancellable with k.

it is sufficient to show that:

Claim

If L, L' are block cancellable with k and for all strings x with $|x| < r(k)$, $x \in L$ iff $x \in L'$, then $L = L'$.

Formal definitions: dependent finiteness

```
Fixpoint build_dep_impl_list' {X: Type} (P: X → ℙ) (L: list X)
         (Hfin: ∀ x, In x L → P x): list {x | P x} :=
  match L as l return (l = L → list {x | P x}) with
  | nil ⇒ λ _ ⇒ nil
  | hd :: tl ⇒
    λ h ⇒ cons (eq_rect (hd :: tl) _
                    (λ Hfin0 : ∀ x, In x (hd :: tl) → P x ⇒
                       exist _ hd (Hfin0 hd (in_eq hd tl))) L h Hfin)
             (build_dep_impl_list' P tl (rest_fin P L hd tl Hfin h))
  end (eq_refl L).
```