Pumping, With or Without Choice

Aquinas Hobor[1,2] **Elaine Li[1,3]** Frank Stephan [2]

[1]Yale-NUS College

[2]National University of Singapore

[3]Runtime Verification, Inc

APLAS 2019

Regularity $\Rightarrow$ Rabin-Scott

Rabin-Scott's Pumping Lemma (Rabin, Scott 1959)

If a language L is regular, then there exists a k s.t. $\forall w \in L.\ |w| \geq k \ \Rightarrow\ \exists x, y, z \in \Sigma^*. \quad w = xyz\ \wedge |xy| \leq k \wedge\ y \neq \epsilon\ \wedge\ \forall n \in \mathbb{N}.\ xy^nz \subseteq L.$

- Non-regular languages satisfy it (Sommerhalder 1980)
- Modus tollens form to show irregularity, e.g. $L = 0^n1^n$

Regularity $\Leftrightarrow$ Jaffe

Jaffe's Pumping Lemma (Jaffe 1978)

A language L is regular iff there exists a k s.t.
$\forall w \in \Sigma^*.\ |w| = k \ \Rightarrow\ \exists x, y, z \in \Sigma^*.\quad w = xyz\ \wedge\ y \neq \epsilon\ \wedge\ \forall u \in \Sigma^*.\ n \in \mathbb{N},\ xyzu \in L\ \Leftrightarrow\ xy^n zu \in L.$

Regularity $\Leftrightarrow$ Jaffe

Definition (Derivative)

$L_x = \{y \in \Sigma^* : xy \in L\}$ is the derivative of L with respect to x.

- e.g. $L = 0^*1^*2^*$, L's derivatives are $L_0 = 0^*1^*2^*$, $L_{01} = 1^*2^*$, $L_{012} = 2^*$.
- Jaffe states that every derivative of language L is equivalent to some derivative with label length shorter than or equal to k.

Myhill-Nerode Theorem (Myhill, Nerode 1958)

A language L is regular iff it has finitely many derivatives.

Regularity $\Leftrightarrow$ EPR

The block pumping (cancellation) property (EPR, 1981)

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* (*omitted*) without affecting word membership.

Theorem of Ehrenfeucht, Parikh and Rozenberg (EPR, 1981)

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

Picture view

012101010210101

Picture view

0 | 1 2 1 | 0 1 | 0 | 1 0 2 | 1 0 | 1 0 | 1 |

Picture view

$$0 \mid 1\,2\,1 \mid 0\,1 \mid 0 \mid \underbrace{1\,0\,2} \mid 1\,0 \mid 1\,0 \mid 1 \mid$$

Picture view

0121010**102**10101

0121010**102102102...102**10101

012101010101

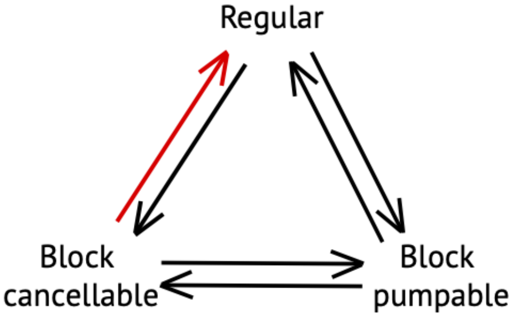
Contributions

- Coq formalization of Jaffe's and EPR's pumping lemmas, and block pumpable language closure properties;
- Clarification of a gap in EPR's proof that block cancelable languages are regular that implicitly uses the Axiom of Choice;
- New choice-free proof using an explicit construction of block cancelable languages from well-formed input sets.

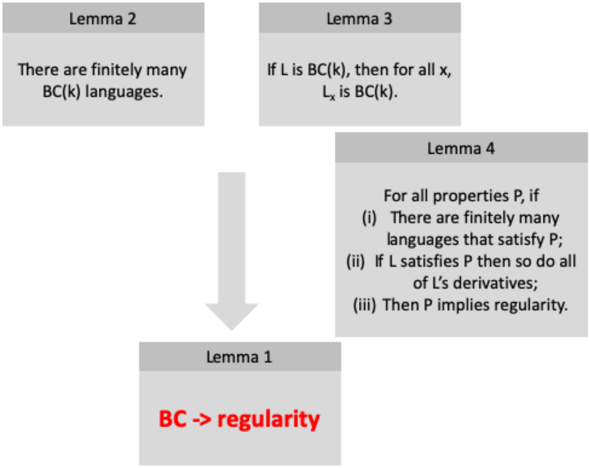
Roadmap

1. Pumping
2. With choice
   - Proof sketch
   - Ramsey's theory
3. Without choice
   - "unshear" function
   - Proof of correctness

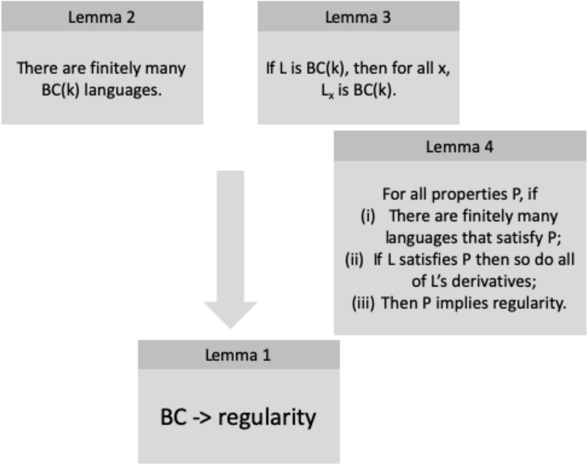
EPR's Theorem



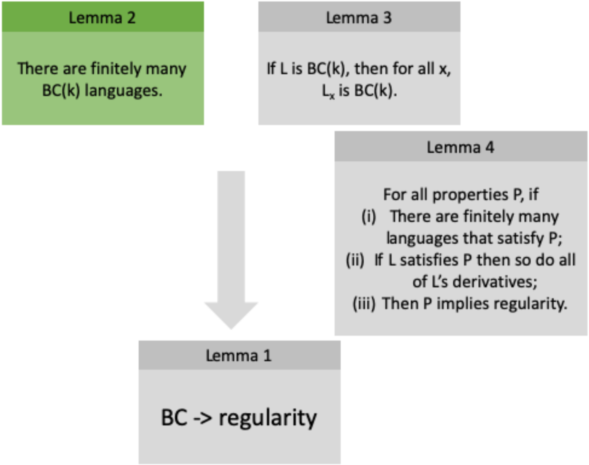
EPR's proof, in pictures



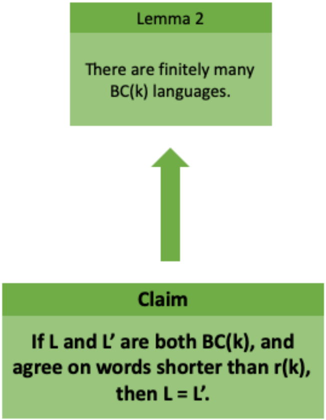
EPR's proof, in pictures



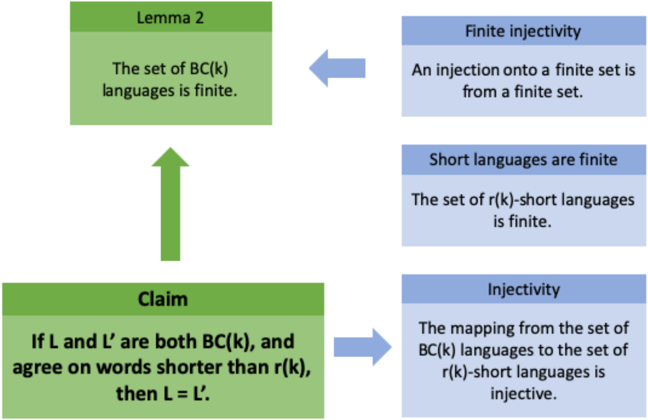
EPR's proof, in pictures



"It is sufficient to show that..."



"It is sufficient to show that..."



Finite injectivity, in Coq

Theorem (Finite injectivity)

```
: forall (P Q : X -> Prop) (f : {x | P 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` - BC(k)
- `Q` - r(k)-short
- `f` - mapping from BC(k) languages to r(k) short languages

Finitehood, in Coq

Definition (Dependent finitehood)

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

Definition (Finitehood)

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

Axiom of functional choice

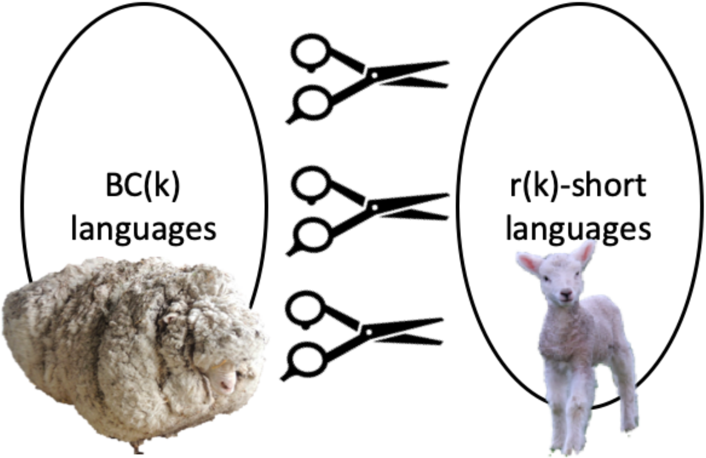
Definition (Functional choice)

```
forall R : A -> B -> Prop,
(forall x : A, exists y : B, R x y) ->
(exists f : A -> B, forall x : A, R x (f x)).
```

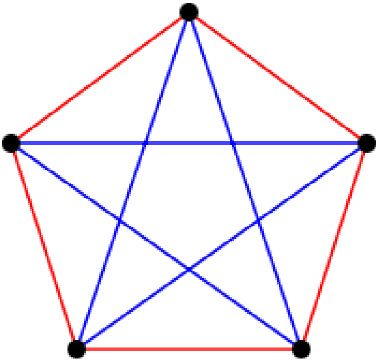
where:

- `A` - {`x | Q x`}, i.e. r(k)-short
- `B` - {`y | P y`}, i.e. BC(k)
- `R` - y "shears down" to x
- `f` - mapping from r(k)-short languages to BC(k) languages

Picture view



Ramsey theory



Ramsey on graphs

One can always find monochromatic cliques in any edge-coloring of a sufficiently large connected graph (Ramsey 1930).

Ramsey on sets

For every natural number k and finite set of colors Q, there exists a natural number $r(k)$ such that for every ordered set I with $r(k)$ elements and for every function mapping each pair (i, j) to a color $C(i, j)$, there exists a subset $J \subset I$ with k elements such that all pairs in J are mapped to the same color.

Ramsey on breakpoint sets

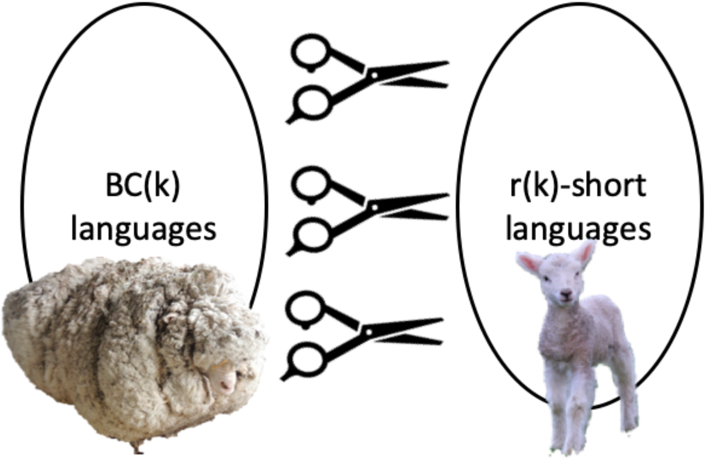
Theorem (Ramsey's theorem for breakpoint sets)

```
forall (k: block_pumping_constant),
exists (rk: block_pumping_constant), rk >= k /\
forall (w: word) (bps: breakpoint_set rk w)
(P: nat -> nat -> Prop),
exists (bps': breakpoint_set k w),
sublist bps' bps /\
((forall (bp1 bp2: breakpoint bps'), bp1<bp2->(P bp1 bp2))
\/ (forall (bp1 bp2: breakpoint bps'), bp1<bp2->~(P bp1 bp2)))
```

Roadmap

1. Pumping
2. With choice
   - Proof sketch
   - Ramsey's theory
3. Without choice
   - "unshear" function
   - Proof of correctness

Picture view



Unshear correctness

1. Shearing an unsheared list returns us the input list;
2. Unshearing a sheared language recovers us the BC(k) language.

Unshear, pictorially

k = 3, r(k) = 6

{ 0, 1, 001, 011, 0011, 01111, 000111, ...}

Unshear, pictorially

k = 3, r(k) = 6

{ 0, 1, 001, 011, 0011, 01111, 000111, ...}

0000000, 0000001, 0000010, 0000011, 0000111, 0000110, 0000101,
0001110, 0000100, 0001010, ..., 1011111, 0111111, 1111111

All words of length r(k)+1

Unshear, pictorially

k = 3, r(k) = 6

{ 0, 1, 001, 011, 0011, 01111, 000111, ...}

0000000, 0000001, 0000010, 0000011, 0000111, 0000110, 0000101, 0001110, 0000100, 0001010, ..., 1011111, 0111111, 1111111

All words of length r(k)+1

Unshear, pictorially

k = 3, r(k) = 6

{ 0, 1, 001, 011, 0011, 01111, 000111, ...}

0000000, 0000001, 0000010, 0000011, 0000111, 0000110, 0000101, 0001110, 0000100, 0001010, ..., 1011111, 0111111, 1111111

All words of length r(k)+1

{ 0, 1, 001, 011, 0011, 01111, 000111, ..., 0000000, 0000001, 0000011, 0000111, 0111111, 1111111}

The chucking function

0 | 1 2 1 | 0 1 | 0 | 1 0 2 | 1 0 | 1 0 | 1 |

The chucking function

$$0 \mid \underbrace{1\;2\;1} \mid 0\;1 \mid 0 \mid 1\;0\;2 \mid 1\;0 \mid 1\;0 \mid 1 \mid$$

The chucking function

$$0 \mid \underbrace{1\ 2\ 1 \mid 0\ 1} \mid 0 \mid 1\ 0\ 2 \mid 1\ 0 \mid 1\ 0 \mid 1 \mid$$

The chucking function

$$0\underbrace{|\,1\,2\,1\,|\,0\,1\,|\,0\,|}\,1\,0\,2\,|\,1\,0\,|\,1\,0\,|\,1\,|$$

Chucking condition

Definition (Chucking function)

```
Definition chuck (k rk n : nat) (lref : list word) :=
filter (fun w => exists_all_pumps w lref k)
(generate_words_of_length (n + rk))
++ lref.
```

Definition (Chucking condition)

```
Definition chuck_prop (k rk n : nat) (lref : list word)
(w : word) :=
(exists_all_pumps_bps_prop w lref k /\ length w = n + rk)
\/ In w lref.
```

Unshear, pseudocode

1. Input: well-formed list of short words lw and candidate word w
2. Output: membership for w in $BC(k)$ language that lw agrees with on short words
3. Algorithm: incrementally consider sets of all words from length $r(k)$ to length $|w|$
    1. For every word, check the existence of a "full" set of k breakpoints such that every pair of breakpoints pumps the word down into lw
    2. If such a set exists, chuck, i.e. add, word into reference list lw
    3. If such a set does not exist, ignore
    4. Repeat with updated lw and set of words of length plus one until $|w|$ is reached
    5. Check membership of w in lw

Unshear correctness

1. Shearing an unsheared list returns us the input list;
2. Unshearing a sheared language recovers us the language.

```
Theorem unshear_correctness:
forall (k: block_pumping_constant),
exists (rk: block_pumping_constant),
forall (l: bc_language_dec k) (lw: list word),
agreement_upto k rk l (chuck_length k rk lw 0) 0 ->
(forall w, In w lw <-> (shear_language rk (unshear k rk lw)) w
/\ unshear k rk lw = (bc_language_dec_proj1 l).
```

shear (unshear)

Trivial!

unshear (shear)

`chuck` is chucking in the right words!

```
Lemma IH_chuck_step: forall (k: block_pumping_constant),
exists (rk: block_pumping_constant),
forall (l: bc_language_dec k) (lw: list word) (m: nat),
agreement_upto k rk l lw m ->
agreement_upto k rk l (chuck k rk (S m) lw) (S m).

Definition agreement_upto (k rk: block_pumping_constant)
(l: bc_language_dec k)
(lw: list word) (m : nat) :=
forall w, In w lw <-> (length w <= m + rk
/\ bc_language_dec_proj1 l w).
```

Unshear correctness

Induction step for the leftward direction:

- Given a word w that is in the new lw, we must show that:
  - $|w| \leq S\ m + k$
  - $L\ w$
- In the case that w was already in lw, we are done.
- In the case that w is newly chucked in,
  - The length requirement is satisfied by `chuck`'s specification
  - By our `chucking` condition, there exists a "full" breakpoint set lp. We specialize L's block cancelation property with w and lp to obtain a canceled word w', which agrees with w on membership.

Unshear correctness

Induction step for the rightward direction:

- Given a word w that satisfies:
  - $|w| \leq S\ m + k$
  - $L\ w$
- We must show that `chuck` actually does chuck it into lw, i.e. that it satisfies the `chuck`ing condition of having a "full" breakpoint set into lw.
- From Ramsey, we know that for any $r(k)$-size breakpoint set, we can find a monochromatic k-size breakpoint set lp.
  - In the case that all pairs in lp are cancelable pumps into lw, we satisfy the `chuck`able condition;
  - In the case that all pairs in lp are not cancelable pumps into lw, we reach a contradiction from the fact that L is $BC(k)$.

unshear (shear)

```
Lemma IH_chuck:
forall (k: block_pumping_constant),
exists (rk: block_pumping_constant),
forall (l: bc_language_dec k) (lw: list word) (m: nat),
agreement_upto k rk l (chuck_length k rk 0 lw) 0 ->
agreement_upto k rk l (chuck_length k rk m lw) m.
```

chuck is chucking in the right words!

New proof, pictorially

Lemma 2

There are finitely many BC(k) languages.

Surjectivity

unshear from well-formed short languages is surjective.

Short languages are finite

The set of r(k)-short languages is finite.

New proof, pictorially

Lemma 2

There are finitely many BC(k) languages.

Surjectivity

unshear from well-formed short languages is surjective.

Short languages are finite

The set of r(k)-short languages is finite.

Thank you!