On the Formalization of Some Results of Context-Free Language Theory

Marcus Vinícius Midena Ramos
Ruy J. G. B. de Queiroz
Nelma Moreira
José Carlos Bacelar Almeida

August 18th, 2016

WoLLIC 2016

BUAP, Puebla, México

Background

- Mathematical Formalization;
- Coq Proof Assistant;
- Context-Free Language Theory.

Related Work

Formal languages

- Language and automata theory has been subject of formalization since the mid-1980s, when Kreitz used the Nuprl proof assistant to prove results about deterministic finite automata and the pumping lemma for regular languages;
- Since then, the theory of regular languages has been the subject of intense formalization by various researchers using many different proof assistants (e.g. Doczkal, Kaiser and Smolka);
- The formalization of context-free language theory, on the other hand, is more recent and includes fewer accomplishments in theory (Norrish and Barthwal, Firsov and Uustalu), being mostly concentrated in certified parser generation.

Related Work

Summary

	Norrish & Barthwal2010	Firsov & Uustalu
Proof assistant	HOL4	Agda
Closure	✓	×
Simplification	✓	*only empty and unit rules*
CNF	✓	✓
GNF	✓	×
PDA	✓	×
PL	✓	×

Related Work

Motivation

- Until 2015, the only comprehensive work is the one by Norrish and Barthwal (HOL4 in 2010);
- The Pumping Lemma has not been published;
- Firsov and Uustalu add a more limited formalization (Agda in 2015);
- No formalization in Coq.
- Formalization of the PL in HOL4 discovered only in november 2015.

Objectives

Formally state and prove:

- Closure properties of context-free languages and grammars;
- Context-free grammar simplification;
- Chomsky Normal Form (CNF);
- Pumping Lemma (PL) for context-free languages.

PL depends on CNF, which in turn depends on grammar simplification.

Steps

1. Selection of an underlying formal logic to express the theory and then a tool that supports it adequately;
2. Representation of the objects of the universe of discourse in this logic;
3. Implementation of a set of basic transformations and mappings over these objects;
4. Statement of the lemmas and theorems that describe the properties and the behaviour of these objects, and establish a consistent and complete theory;
5. Formal derivation of proofs of these lemmas and theorems, leading to proof objects that can confirm their validity.

Definitions

Main definitions in Coq:

- Symbols (including terminal and non-terminal);
- Sentential forms (strings of terminal and non-terminal symbols);
- Sentences (strings of terminal symbols);
- Context-free grammars;
- Derivations;
- Binary trees.

Definitions

Context-free grammar

(V, Σ, P, S)

```
Record cfg (non_terminal terminal : Type): Type:= {
start_symbol: non_terminal;
rules: non_terminal → sf → Prop;
rules_finite:
    ∃ n: nat,
    ∃ ntl: nlist,
    ∃ tl: tlist,
    rules_finite_def start_symbol rules n ntl tl }.
```

Definitions

Context-free grammar

```
Definition rules_finite_def
    (non_terminal terminal : Type)
    (ss: non_terminal)
    (rules: non_terminal → sf → Prop)
    (n: nat)
    (ntl: list non_terminal)
    (tl: list terminal) :=
In ss ntl ∧
(∀ left: non_terminal,
 ∀ right: list (non_terminal + terminal),
 rules left right →
 length right ≤ n ∧
 In left ntl ∧
 (∀ s : non_terminal, In (inl s) right → In s ntl) ∧
 (∀ s : terminal, In (inr s) right → In s tl)).
```

Definitions

Derivation

$s_1 \Rightarrow^* s_2$

```
Inductive derives
  (non_terminal terminal : Type)
  (g : cfg non_terminal terminal)
  : sf → sf → Prop :=
  | derives_refl :
    ∀ s : sf,
    derives g s s
  | derives_step :
    ∀ (s1 s2 s3 : sf)
    ∀ (left : non_terminal)
    ∀ (right : sf),
    derives g s1 (s2 ++inl left :: s3) →
    rules g left right → derives g s1 (s2 ++right ++s3)
```

Sequence

1. General purpose libraries;
   - Basic lemmas on arithmetic, lists and logic;
   - Basic lemmas on context-free languages and grammars;
   - Basic lemmas on binary trees and their relation to CNF grammars;
2. Closure properties;
3. Grammar simplification $\rightarrow$ Chomsky Normal Form $\rightarrow$ Pumping Lemma.

Generic CFG Library

General results on context-free gramars and languages:

- 4,393 lines of Coq script, ~18.3% of the total;
- 105 lemmas and theorems;
- Supports the whole formalization;
- Some examples:
  - Derivation transitivity:
    $\forall g, s_1, s_2, s_3, (s_1 \Rightarrow_g^* s_2) \rightarrow (s_2 \Rightarrow_g^* s_3) \rightarrow (s_1 \Rightarrow_g^* s_3)$
  - Context independence:
    $\forall g, s_1, s_2, s, s', (s_1 \Rightarrow_g^* s_2) \rightarrow (s \cdot s_1 \cdot s' \Rightarrow_g^* s \cdot s_2 \cdot s')$
  - Concatenation:
    $\forall g, s_1, s_2, s_3, s_4, (s_1 \Rightarrow_g^* s_2) \rightarrow (s_3 \Rightarrow_g^* s_4) \rightarrow (s_1 \cdot s_3 \Rightarrow_g^* s_2 \cdot s_4)$
  - Derivation independence:
    $\forall g, s_1, s_2, s_3, (s_1 \cdot s_2 \Rightarrow_g^* s_3) \rightarrow \exists s_1', s_2' \,|\, (s_3 = s_1' \cdot s_2') \wedge (s_1 \Rightarrow_g^* s_1') \wedge (s_2 \Rightarrow_g^* s_2')$

Method

Most of the work share a common objective: to construct a new grammar from an existing one (or two existing ones).

$$G \rightsquigarrow G'$$

$$\begin{aligned} P &\rightsquigarrow P' \\ \Sigma &\rightsquigarrow \Sigma' \\ V &\rightsquigarrow V' \\ S &\rightsquigarrow S' \end{aligned}$$

Method

This is the case of:

- Closure properties:
  - Union;
  - Concatenation;
  - Kleene star;
- Grammar simplification:
  - Elimination of empty rules;
  - Elimination of unit rules;
  - Elimination of useless symbols;
  - Elimination of inaccessible symbols;
- Chomsky Normal Form (CNF).

Thus, a common method to be used in all these cases has been devised.

Does not apply for the Pumping Lemma.

Method

For the first three objectives

1. Inductively define a new type of non-terminal symbols (if necessary):
   - Ensure that the start symbol of the grammar does not appear in the right-hand side of any rule;
   - New non-terminals are necessary;
   - Map previous;
   - Add new.
2. Inductively define the rules of the new grammar:
   - Exclude previous;
   - Map previous;
   - Add new (directly and/or indirectly).

Method

For the first three objectives

3. Define the new grammar:
   - Use the new non-terminal symbols and the new rules;
   - Start symbol (previous or new);
   - Prove the finiteness of the set of rules for this new grammar;
4. State and prove all the lemmas and theorems that will assert that the newly defined grammar has the desired properties;
5. Consolidate the results within the same scope and finally with the previously obtained results.

Closure Properties

Union

Given two arbitrary context-free grammars g_1 and g_2, the following definitions are used to construct g_3 such that:

$$L(g_3) = L(g_1) \cup L(g_2)$$

(that is, the language generated by g_3 is the union of the languages generated by g_1 and g_2).

Closure Properties

Union

$g_1,\ g_2 \rightsquigarrow g_3$

$$
\begin{array}{rcl}
P_1,\ P_2 & \underbrace{\rightsquigarrow}_{+2} & P_3 \\
\Sigma,\ \Sigma & \underbrace{\rightsquigarrow}_{same} & \Sigma \\
V_1,\ V_2 & \underbrace{\rightsquigarrow}_{+1} & V_3 \\
S_1,\ S_2 & \underbrace{\rightsquigarrow}_{new} & S_3
\end{array}
$$

Closure Properties

Union

- For the new set of non-terminals (`g_uni_nt`):
  - All the non-terminals of g_1;
  - All the non-terminals of g_2;
  - A fresh new non-terminal symbol (S_3).
- For the new set of rules (`g_uni_rules`):
  - All the rules of g_1;
  - All the rules of g_2;
  - Two new rules: $S_3 \rightarrow S_1$ and $S_3 \rightarrow S_2$.
- For the new grammar (`g_uni`):
  - The new set of non-terminals;
  - The new set of rules;
  - The new non-terminal (S_3) as the start symbol.

Closure Properties

Correctness and Completeness

Union (correctness)

Considering that g_3 is the union of g_1 and g_2 and S_3, S_1 and S_2 are, respectively, the start symbols of g_3, g_1 and g_2):

$$\forall g_1,\ g_2,\ s_1,\ s_2, (S_1 \Rightarrow^*_{g_1} s_1 \rightarrow S_3 \Rightarrow^*_{g_3} s_1) \wedge (S_2 \Rightarrow^*_{g_2} s_2 \rightarrow S_3 \Rightarrow^*_{g_3} s_2)$$

Union (completeness)

$$\forall s_3, (S_3 \Rightarrow^*_{g_3} s_3) \rightarrow (S_1 \Rightarrow^*_{g_1} s_3) \vee (S_2 \Rightarrow^*_{g_2} s_3)$$

Grammar Simplification

Objective

For all G, if G is non-empty, then there exists G' such that:

- $L(G) = L(G')$;
- G' has no empty rules (except for one, if G generates the empty string);
- G' has no unit rules;
- G' has no useless symbols;
- G' has no inaccessible symbols;
- The start symbol of G' does not appear on the right-hand side of any other rule of G'.

Grammar Simplification

Empty rule

- An *empty rule* $r \in P$ is a rule whose right-hand side β is empty (e.g. $X \to \epsilon$).

We formalize that for all G, there exists G' such that:

- $L(G) = L(G')$ and
- G' has no empty rules, except for a single rule $S' \to \epsilon$ if $\epsilon \in L(G)$ (S' is the initial symbol of G');
- S' does not appear on the right-hand side of any rule of G'.

Grammar Simplification

Empty rules elimination

From g_1:

1. Construct g_2 such that $L(g_2) = L(g_1) - \epsilon$;
2. Construct g_3 (using g_2) such that:
   - $L(g_3) = L(g_1) \cup \{\epsilon\}$ if $\epsilon \in L(g_1)$ or
   - $L(g_3) = L(g_1)$ if $\epsilon \notin L(g_1)$.

Grammar Simplification

Empty rules elimination

Step 1:

$g_1 \rightsquigarrow g_2$

$$
\begin{array}{lcl}
P_1 & \underbrace{\rightsquigarrow}_{\leq,\geq,+1} & P_2 \\
\Sigma & \underbrace{\rightsquigarrow}_{same} & \Sigma \\
V_1 & \underbrace{\rightsquigarrow}_{+1} & V_2 \\
S_1 & \underbrace{\rightsquigarrow}_{new} & S_2
\end{array}
$$

Grammar Simplification

Empty rules elimination

Step 1:

- For the new set of non-terminals (`non_terminal'`):
  - All the non-terminals of g_1;
  - A fresh new non-terminal symbol (S_2).
- For the new set of rules (`g_emp_rules`):
  - All non-empty rules of g_1;
  - All rules of g_1 with every combination on nullable symbols in the right-hand side removed, except if empty;
  - One new rule: $S_2 \rightarrow S_1$.
- For the new grammar (`g_emp`):
  - The new set of non-terminals;
  - The new set of rules;
  - The new non-terminal (S_2) as the start symbol.

Grammar Simplification

Empty rules elimination

Step 2:

$g_1 \rightsquigarrow g_3$

$$P_1 \underbrace{\rightsquigarrow}_{same\ of\ g_2\ or\ +1} P_3$$

$$\Sigma \underbrace{\rightsquigarrow}_{same\ of\ g_2} \Sigma$$

$$V_1 \underbrace{\rightsquigarrow}_{same\ of\ g_2} V_3$$

$$S_1 \underbrace{\rightsquigarrow}_{same\ of\ g_2} S_3$$

Grammar Simplification

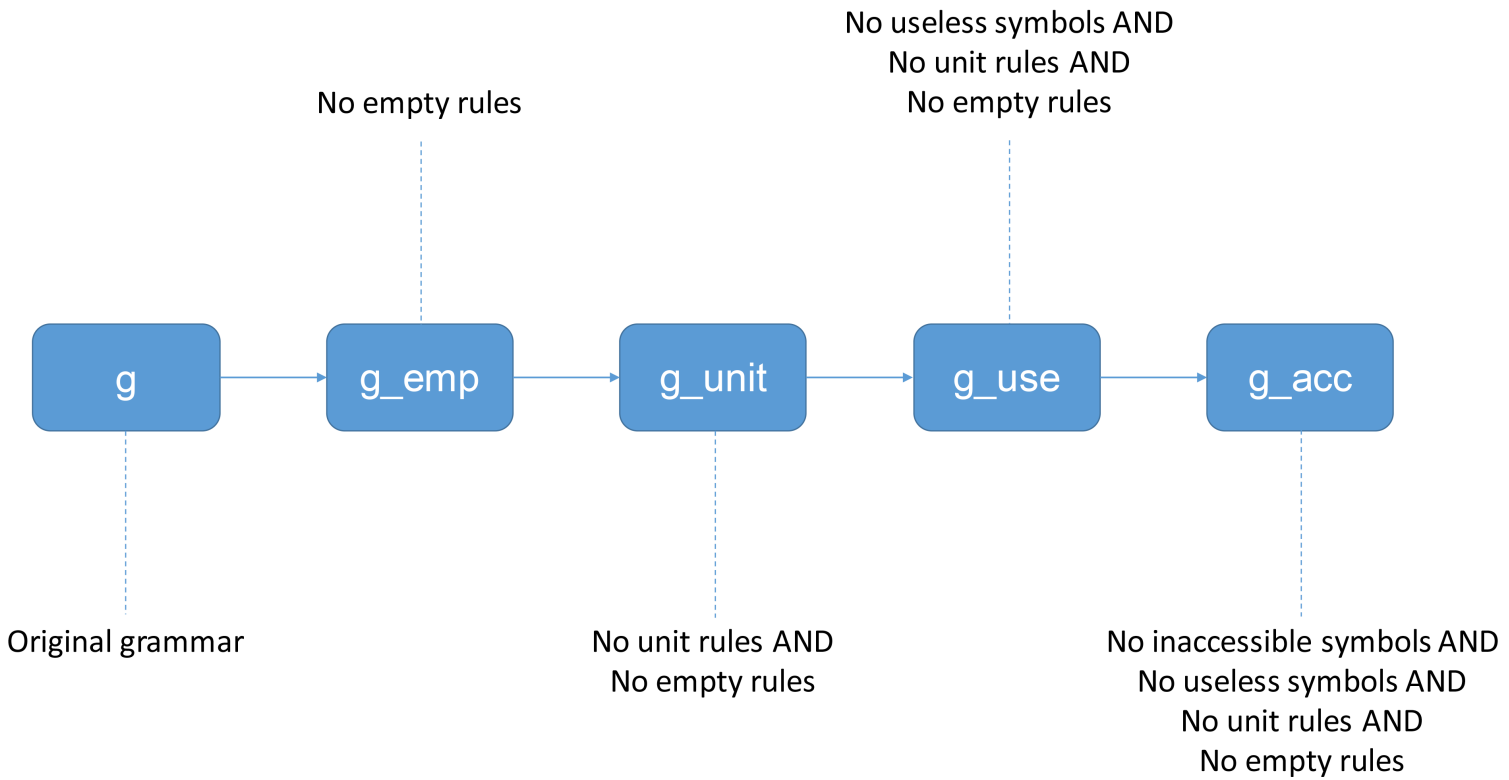
Empty rules elimination

Step 2:

- For the new set of non-terminals (`*`):
  - All the non-terminals of Step 1.
- For the new set of rules (`g_emp'_rules`):
  - All the rules of Step 1;
  - One new rule: $S_2 \rightarrow \epsilon$ if $\epsilon \in L(g_1)$.
- For the new grammar (`g_emp'`):
  - The same set of non-terminals;
  - The new set of rules;
  - The same start symbol (S_2).

Grammar Simplification

Unification



Chomsky Normal Form

Statement

$$\forall\, G = (V, \Sigma, P, S),$$

$$\exists\, G' = (V', \Sigma, P', S') \mid L(G) = L(G') \wedge \forall\, (\alpha \rightarrow_{G'} \beta) \in P',$$

$$(\beta \in \Sigma) \vee (\beta \in N \cdot N) \vee ((\alpha = S') \wedge (\beta = \epsilon))$$

From g_1:

1. Construct g_2 such that $L(g_2) = L(g_1) - \epsilon$ and g_2 is in CNF;
2. Construct g_3 (using g_2) such that $L(g_3) = L(g_2) \cup \{\epsilon\}$.

Chomsky Normal Form

Strategy

From g_1 to g_2:

- For the new set of non-terminals:
  - One for every possibile (non-empty) sequence of terminal and non-terminal symbols of g_1: $[...]$
- For the new set of rules:
  - One for every terminal symbol t of g_1: $[t] \to t$;
  - One for every rule $X \to t$ of g_1: $[X] \to t$;
  - One for every rule $left \to s_1 s_2 \beta$ of g_1: $[left] \to [s1][s_2\beta]$;
  - One for every rule $[left] \to [s_1][s_2 s_3 \beta]$ of g_2: $[s_2 s_3 \beta] \to [s_2][s_3 \beta]$
- For the new grammar:
  - The new set of non-terminals;
  - The new set of rules;
  - The mapped start symbol ($[S_1]$).

Chomsky Normal Form

Strategy

From g_1 to g_3:

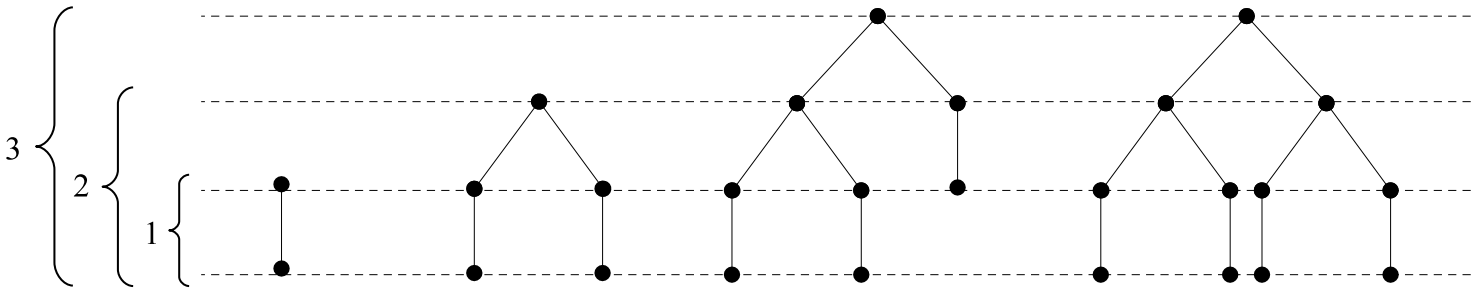
- For the new set of non-terminals:
  - The same of g_2.
- For the new set of rules:
  - The same of g_2;
  - One extra rule: $[S_1] \rightarrow \epsilon$
- For the new grammar:
  - The new set of non-terminals;
  - The new set of rules;
  - The mapped start symbol ($[S_1]$).

Generic Binary Trees Library

General results on binary trees and their relation to CNF grammars:

- 4,539 lines of Coq script, ~18.9% of the total;
- 84 lemmas that support the formalization of the Pumping Lemma.
- Based on the definition of `btree`:

```
Inductive btree (non_terminal terminal: Type): Type:=
| bnode_1: non_terminal → terminal → btree
| bnode_2: non_terminal → btree → btree → btree.
```



Pumping Lemma

Statement

$$\forall\ \mathcal{L}, (\text{cfl}\ \mathcal{L}) \rightarrow \exists\ n\ |$$

$$\forall\ \alpha, (\alpha \in \mathcal{L}) \wedge (|\alpha| \geq n) \rightarrow$$

$$\exists\ u, v, w, x, y \in \Sigma^*\ |\ (\alpha = uvwxy) \wedge (|vx| \geq 1) \wedge (|vwx| \leq n) \wedge$$

$$\forall\ i, uv^i wx^i y \in \mathcal{L}$$

- Used to prove a language not context-free;
- Can not be used to prove a language context-free;
- First proved by Bar-Hillel, Perles and Shamir in 1961;
- Formalization depends heavily on grammar simplification, CNF and binary trees, and uses the Pigeonhole Principle together with an inductive argument.

Discussion

Pumping Lemma

$$\forall\ \mathcal{L}, (\mathsf{cfl}\ \mathcal{L}) \rightarrow \exists\ n\ |$$

$$\forall\ \alpha, (\alpha \in \mathcal{L}) \wedge (|\alpha| \geq n) \rightarrow$$

$$\exists\ u, v, w, x, y \in \Sigma^*\ |\ (\alpha = uvwxy) \wedge (|vx| \geq 1) \wedge (|uy| \geq 1) \wedge (|vwx| \leq n) \wedge$$

$$\forall\ i, uv^i wx^i y \in \mathcal{L}$$

Discussion

Pumping Lemma

A variant of the Pumping Lemma, using a smaller value of n, has also been proved. This result uses $n = 2^{k-1} + 1$ instead of $n = 2^k$ (k is the number of non-terminal symbols in the CNF grammar). Since the proof needs a binary tree of height at least $k + 1$ in order to proceed, and since trees of height i have as frontier strings of length maximum 2^{i-1}, it is possible to consider strings of length equal to or greater than $2^{k-1} + 1$ (and not only of length equal to or greater than 2^k) in order to have the corresponding binary tree with height equal to or higher than $k + 1$. This way, two slightly different proofs of the Pumping Lemma have been produced: one with $n = 2^k$ (`pumping_lemma`) and the other with $n = 2^{k-1} + 1$ (`pumping_lemma_v2`).

Discussion

Pumping Lemma

The statement of (`pumping_lemma_v2`) becomes:

$$\forall\ \mathcal{L}, (\text{cfl}\ \mathcal{L}) \rightarrow \exists\ n\ |$$

$$\forall\ \alpha, (\alpha \in \mathcal{L}) \wedge (|\alpha| \geq n) \rightarrow$$

$$\exists\ u, v, w, x, y \in \Sigma^*\ |\ (\alpha = uvwxy) \wedge (|vx| \geq 1) \wedge (|vwx| \leq (n-1) * 2) \wedge$$

$$\forall\ i, uv^i wx^i y \in \mathcal{L}$$

Summary

Facts

- 23,985 lines of Coq script spread in 18 libraries;
- Eight auxiliary libraries contain 11,781 lines of Coq script and correspond to almost half of the formalization (49.1%);
- Two of these auxiliary libraries (`cfg.v` and `trees.v`) sum, alone, 8,932 lines or more than one third (37.2%) of the total;
- 533 lemmas and theorems, 83 definitions and 40 inductive definitions among 1,067 declared names;
- Created and compiled with the Coq Proof Assistant, version 8.4pl4 (June 2014), using CoqIDE for Windows;
- $2\frac{1}{2}$ years of work.

Summary

Comparison

	Norrish & Barthwal	Firsov & Uustalu	Ramos
Proof assistant	HOL4	Agda	Coq
Closure	✓	×	✓
Simplification	✓	*empty and unit rules*	✓
CNF	✓	✓	✓
GNF	✓	×	×
PDA	✓	×	×
PL	✓	×	✓

Insights

- Do not formalize a theory that you are not familiar with;
- Start with a easy and small set of results;
- Not every simple informal proof is easily formalizable;
- Formal proofs grow in number and size very rapidly;
- Most of the proofs are by induction on inductive definitions and case analysis;
- Mappings from one type to another creates a big burden in the formal proofs that is hidden in informal proofs;
- Make sure the lemma is provable;
- Work hard on inductive definitions;
- Experiment with the definitions (use examples);
- Separate generic from specific results (create libraries);
- Define and adopt a consistent naming policy.

Decidable Problems for CFLs

- Emptyness:
  Check if the start symbol is useful;
- Existence of an empty string:
  Check if the start symbol is nullable;
- Existence of a non-empty string:
  Check if the start symbol is useful in a grammar without the empty rule;
- Membership:
  Search through a finite number of derivations sequences (for a grammar in CNF).

This will eliminate unnecessary hypotheses in the statement of some main theorems.

New Devices and Results

- Pushdown automata, including: definition, equivalence of pushdown automata and context-free grammars; equivalence of empty stack and final state acceptance criteria; non-equivalence of the deterministic and the non-deterministic models;
- Elimination of left recursion in context-free grammars and Greibach Normal Form;
- Derivation trees, ambiguity and inherent ambiguity;
- Closure with regular languages;
- Decidable problems for context-free languages (finiteness and infiniteness);
- Odgen's Lemma.

Achievements and Results

- Formalization of an area which has relied so far mostly in informal arguments;
- A set of libraries that formalizes an important subset of context-free language theory;
- A kit that can be further used for teaching and research (Kreitz).
- First formalization of a coherent and complete subset of context-free language theory in the Coq proof assistant;
- First formalization of the Pumping Lemma for context-free languages in Coq;
- Second formalization ever (in any proof assistant) of the Pumping Lemma for context-free languages;
- Second most comprehensive formalization of an important subset of the context-free language theory in any proof assistant.

More Information

- "On the Formalization of Some Results of Context-Free Language Theory" (in WoLLIC 2016 Proceedings):
  `http://www.springer.com/br/book/9783662529201`
- PhD Thesis:
  `http://www.univasf.edu.br/~marcus.ramos/tese.pdf`
- Source files:
  `https://github.com/mvmramos/v1`
- These slides:
  `http://www.univasf.edu.br/~marcus.ramos/slides-wollic-2016.pdf`

The End

Thank You!