Learning local substitutable CF languages from text in polynomial time and data by reduction

François Coste and Jacques Nicolas

Univ Rennes, Inria, CNRS, IRISA

ICGI 2018

Previously, on learning local substitutable languages...

ICGI'12–14, PhD thesis of Gaelle Garet 2014

- Learning substitutable[1] and k, l-substitutable[2] languages
  $\rightsquigarrow$ Learning k, l-local substitutable languages for proteins

L is k, l-local substitutable (k, l-LS) iff

$x_1, y_1, z_1, x_2, y_2, z_2, x_3, z_3 \in \Sigma^*, u \in \Sigma^k, v \in \Sigma^l, y_1, y_2 \neq \lambda$

$$x_1 u y_1 v z_1 \in L \wedge x_3 u y_2 v z_3 \in L \Rightarrow (x_2 y_1 z_2 \in L \Leftrightarrow x_2 y_2 z_2 \in L)$$

$$\text{i.e. } [y_1]_L = [y_2]_L$$

$k = 1, l = 2$: "the man gulped down a coffee in Rennes" $\in L$
"a man xertzed a coffee before leaving" $\in L$
$\Rightarrow [\text{gulped down}]_L = [\text{xertzed}]_L$:
If "a guy gulped down a beer in Wrocław" $\in L$
Then "a guy xertzed a beer in Wrocław" $\in L$

[1] A. Clark and R. Eyraud. "Polynomial Identification in the Limit of Substitutable Context-free Languages". In: *Journal of Machine Learning Research* 8 (2007).

[2] R. Yoshinaka. "Identification in the Limit of k,l-Substitutable Context-Free Languages". In: *ICGI*. vol. 5278. Lecture Notes in Computer Science. 2008.

Previously, on learning local substitutable languages...

ICGI'12–14, PhD thesis of Gaelle Garet 2014

- Learning substitutable[1] and k, l-substitutable[2] languages
  $\rightsquigarrow$ Learning k, l-local substitutable languages for proteins

L is k, l-local substitutable (k, l-LS) iff

$x_1, y_1, z_1, x_2, y_2, z_2, x_3, z_3 \in \Sigma^*, u \in \Sigma^k, v \in \Sigma^l, y_1, y_2 \neq \lambda$

$$x_1 u y_1 v z_1 \in L \wedge x_3 u y_2 v z_3 \in L \Rightarrow (x_2 y_1 z_2 \in L \Leftrightarrow x_2 y_2 z_2 \in L)$$

$$\text{i.e. } [y_1]_L = [y_2]_L$$

L is local substitutable (LS) if it is k, l-LS for some k, l

[1] A. Clark and R. Eyraud. "Polynomial Identification in the Limit of Substitutable Context-free Languages". In: *Journal of Machine Learning Research* 8 (2007).

[2] R. Yoshinaka. "Identification in the Limit of k,l-Substitutable Context-Free Languages". In: *ICGI*. vol. 5278. Lecture Notes in Computer Science. 2008.

Previously, on learning local substitutable languages...

ICGI'12–14, PhD thesis of Gaelle Garet 2014

- Learning substitutable[1] and k, l-substitutable[2] languages
  $\rightsquigarrow$ Learning k, l-local substitutable languages for proteins

L is k, l-local substitutable (k, l-LS) iff

$x_1, y_1, z_1, x_2, y_2, z_2, x_3, z_3 \in \Sigma^*, u \in \Sigma^k, v \in \Sigma^l, y_1, y_2 \neq \lambda$

$$x_1 u y_1 v z_1 \in L \wedge x_3 u y_2 v z_3 \in L \Rightarrow (x_2 y_1 z_2 \in L \Leftrightarrow x_2 y_2 z_2 \in L)$$

$$\text{i.e. } [y_1]_L = [y_2]_L$$

L is local substitutable (LS) if it is k, l-LS for some k, l

- Practical learning algorithm by reduction (ReGLiS)
  When training sample is good (*i.e.* includes a characteristic sample):
  - Runs in polynomial time wrt to training sample S
  - Returns CF grammar in canonical reduced form for k, l-LS target

[1] A. Clark and R. Eyraud. "Polynomial Identification in the Limit of Substitutable Context-free Languages". In: *Journal of Machine Learning Research* 8 (2007).

[2] R. Yoshinaka. "Identification in the Limit of k,l-Substitutable Context-Free Languages". In: *ICGI*. vol. 5278. Lecture Notes in Computer Science. 2008.

Previously, on learning local substitutable languages...

ICGI'12–14, PhD thesis of Gaelle Garet 2014

- Learning substitutable[1] and k, l-substitutable[2] languages
  $\rightsquigarrow$ Learning k, l-local substitutable languages for proteins

L is k, l-local substitutable (k, l-LS) iff

$x_1, y_1, z_1, x_2, y_2, z_2, x_3, z_3 \in \Sigma^*, u \in \Sigma^k, v \in \Sigma^l, y_1, y_2 \neq \lambda$

$$x_1 u y_1 v z_1 \in L \wedge x_3 u y_2 v z_3 \in L \Rightarrow (x_2 y_1 z_2 \in L \Leftrightarrow x_2 y_2 z_2 \in L)$$

$$\text{i.e. } [y_1]_L = [y_2]_L$$

L is local substitutable (LS) if it is k, l-LS for some k, l

- Practical learning algorithm by reduction (ReGLiS)
  When training sample is good (*i.e.* includes a characteristic sample):
  - Runs in polynomial time wrt to training sample S
  - Returns CF grammar in canonical reduced form for k, l-LS target

[1] A. Clark and R. Eyraud. "Polynomial Identification in the Limit of Substitutable Context-free Languages". In: *Journal of Machine Learning Research* 8 (2007).

[2] R. Yoshinaka. "Identification in the Limit of k,l-Substitutable Context-Free Languages". In: *ICGI*. vol. 5278. Lecture Notes in Computer Science. 2008.

Today: Learning in polynomial time and data by reduction

- Polynomial time and (thick) data identification in the limit
- Grammars in reduced normal form (RNF)
- ReGLiS$_{\text{core}}$:
  - *Single pass* algorithm
  - Build parsing graph and reduce rule *at once*
  - Returns in polynomial time a CF in RNF, by rejection of *"bad" training samples*
- Conclusion, perspectives and questions...

Polynomial time and (thick) data identification in the limit

IPTD

A class $\mathcal{L}$ of languages is *identifiable in the limit from polynomial time and data* (IPTD) using a class $\mathcal{R}$ of representation [3][4]

iff there exist two polynomials $p()$ and $q()$ and an algorithm $\mathcal{A}$ such that:

- Given a training sample S of size $\|S\|$,
  $\mathcal{A}$ returns a representation R in $\mathcal{R}$ consistent with S in $\mathcal{O}(p(\|S\|))$;
- For each representation R in $\mathcal{R}$ of size $\|R\|$,
  there exists a characteristic sample[a] CS of size at most $\mathcal{O}(q(\|R\|))$

[a] such that $\mathcal{A}$ returns a representation R' equivalent with R for any sample $S \supseteq CS$

- The learnability criterion to target for tractable Grammatical Inference

[3] C. de la Higuera. "Characteristic Sets for Polynomial Grammatical Inference". In: *Machine Learning* 27.2 (1997)

[4] R. Eyraud, J. Heinz, and R. Yoshinaka. "Efficiency in the Identification in the Limit Learning Paradigm". In: *Topics in Grammatical Inference*. 2016

IPTD

A class $\mathcal{L}$ of languages is *identifiable in the limit from polynomial time and data* (IPTD) using a class $\mathcal{R}$ of representation [3][4]

iff there exist two polynomials $p()$ and $q()$ and an algorithm $\mathcal{A}$ such that:

- Given a training sample S of size $\|S\|$,
  $\mathcal{A}$ returns a representation R in $\mathcal{R}$ consistent with S in $\mathcal{O}(p(\|S\|))$;
- For each representation R in $\mathcal{R}$ of size $\|R\|$,
  there exists a characteristic sample[a] CS of size at most $\mathcal{O}(q(\|R\|))$

[a] such that $\mathcal{A}$ returns a representation R' equivalent with R for any sample $S \supseteq CS$

- The learnability criterion to target for tractable Grammatical Inference
- Importance of representation: DFA are IPTD, NFA are not IPTD

[3] C. de la Higuera. "Characteristic Sets for Polynomial Grammatical Inference". In: *Machine Learning* 27.2 (1997)

[4] R. Eyraud, J. Heinz, and R. Yoshinaka. "Efficiency in the Identification in the Limit Learning Paradigm". In: *Topics in Grammatical Inference*. 2016

IPTD

A class $\mathcal{L}$ of languages is *identifiable in the limit from polynomial time and data* (IPTD) using a class $\mathcal{R}$ of representation [3][4]

iff there exist two polynomials $p()$ and $q()$ and an algorithm $\mathcal{A}$ such that:

- Given a training sample S of size $\|S\|$,
  $\mathcal{A}$ returns a representation R in $\mathcal{R}$ consistent with S in $\mathcal{O}\left(p(\|S\|)\right)$;
- For each representation R in $\mathcal{R}$ of size $\|R\|$,
  there exists a characteristic sample[a] CS of size at most $\mathcal{O}\left(q(\|R\|)\right)$

[a] such that $\mathcal{A}$ returns a representation R' equivalent with R for any sample $S \supseteq CS$

- The learnability criterion to target for tractable Grammatical Inference
- Importance of representation: DFA are IPTD, NFA are not IPTD
- Not fair for CFG? ...

[3] C. de la Higuera. "Characteristic Sets for Polynomial Grammatical Inference". In: *Machine Learning* 27.2 (1997)

[4] R. Eyraud, J. Heinz, and R. Yoshinaka. "Efficiency in the Identification in the Limit Learning Paradigm". In: *Topics in Grammatical Inference*. 2016

Compression is disadvantaged in IPTD

- NFA vs DFA: CS for DFA could ensure convergence to target language, but *NFA for the language can be exponentially smaller*
- CF grammars $G_n = \langle \Sigma, N, P, N_n \rangle$ for $\{a^{2^n}\}$:

  $$N_n \to N_{n-1} N_{n-1}$$
  $$N_i \to N_{i-1} N_{i-1} \text{ for } 1 < i < n$$
  $$N_1 \to aa$$

  representing each a unique string a^{2^n}, that has to be in CS but whose *sizes grow exponentially with* $\|G_n\|$.

Compression is disadvantaged in IPTD

- NFA vs DFA: CS for DFA could ensure convergence to target language, but *NFA for the language can be exponentially smaller*
- CF grammars $G_n = \langle \Sigma, N, P, N_n \rangle$ for $\{a^{2^n}\}$:

  $$N_n \to N_{n-1} N_{n-1}$$
  $$N_i \to N_{i-1} N_{i-1} \text{ for } 1 < i < n$$
  $$N_1 \to aa$$

  representing each a unique string a^{2^n}, that has to be in CS but whose *sizes grow exponentially with* $\|G_n\|$.

In contrast, reduced grammars produced by ReGLiS don't introduce non-terminals which are not needed for the language ($\sim$ DFA).

The reduced grammar for $\{a^{2^n}\}$ is $S \to a^{2^n}$.

And k, l parameters from k,l-LS should constrain the grammar space...

Compression is disadvantaged in IPTD

- NFA vs DFA: CS for DFA could ensure convergence to target language, but *NFA for the language can be exponentially smaller*
- CF grammars $G_n = \langle \Sigma, N, P, N_n \rangle$ for $\{a^{2^n}\}$:

  $$N_n \to N_{n-1} N_{n-1}$$
  $$N_i \to N_{i-1} N_{i-1} \text{ for } 1 < i < n$$
  $$N_1 \to aa$$

  representing each a unique string a^{2^n}, that has to be in CS but whose *sizes grow exponentially with* $\|G_n\|$.

In contrast, reduced grammars produced by ReGLiS don't introduce non-terminals which are not needed for the language ($\sim$ DFA).

The reduced grammar for $\{a^{2^n}\}$ is $S \to a^{2^n}$.

And k, l parameters from k,l-LS should constrain the grammar space...

Open question: can we bind polynomially the size of the characteristic sets wrt the size of reduced grammars (or wrt the size of reduced grammars for k,l-LS languages)? Any ideas?

We will thus use here the classical criterion for CFG: IPT**t**D...

IPTtD[56]

Thickness of grammar G: $\tau_G = \max\{|w(\alpha)| : (A \to \alpha) \in P\}$

where $w(\alpha)$ is the smallest word derived from α

A class $\mathcal{L}$ of languages is identifiable in the limit from polynomial time and *thick* data (IPTtD) using a class $\mathcal{G}$ of grammars

iff there exist two polynomials $p()$ and $q()$ and an algorithm $\mathcal{A}$ such that:

- Given a training sample S of size $\|S\|$,
  $\mathcal{A}$ returns a grammar G in $\mathcal{G}$ consistent with S in $\mathcal{O}(p(\|S\|))$;
- For each grammar G in $\mathcal{G}$ of size $\|G\|$,
  there exists a characteristic sample[a] of size at most $\mathcal{O}(q(\|G\|, \tau_G))$

[a] such that $\mathcal{A}$ returns a grammar G' equivalent with G for any sample $S \supseteq CS$

[5] R. Yoshinaka. "Identification in the Limit of k,l-Substitutable Context-Free Languages". In: *ICGI*. vol. 5278. Lecture Notes in Computer Science. 2008.

[6] R. Eyraud, J. Heinz, and R. Yoshinaka. "Efficiency in the Identification in the Limit Learning Paradigm". In: *Topics in Grammatical Inference*. 2016.

IPTtD[56]

Thickness of grammar G: $\tau_G = \max\{|w(\alpha)| : (A \to \alpha) \in P\}$

where $w(\alpha)$ is the smallest word derived from α

A class $\mathcal{L}$ of languages is identifiable in the limit from polynomial time and *thick* data (IPTtD) using a class $\mathcal{G}$ of grammars

iff there exist two polynomials $p()$ and $q()$ and an algorithm $\mathcal{A}$ such that:

- Given a training sample S of size $\|S\|$,
  $\mathcal{A}$ returns a grammar G in $\mathcal{G}$ consistent with S in $\mathcal{O}(p(\|S\|))$;
- For each grammar G in $\mathcal{G}$ of size $\|G\|$,
  there exists a characteristic sample[a] of size at most $\mathcal{O}(q(\|G\|, \tau_G))$

[a] such that $\mathcal{A}$ returns a grammar G' equivalent with G for any sample $S \supseteq CS$

[5] R. Yoshinaka. "Identification in the Limit of k,l-Substitutable Context-Free Languages". In: *ICGI*. vol. 5278. Lecture Notes in Computer Science. 2008.

[6] R. Eyraud, J. Heinz, and R. Yoshinaka. "Efficiency in the Identification in the Limit Learning Paradigm". In: *Topics in Grammatical Inference*. 2016.

Class $\mathcal{G}$ of grammars learnt by reduction?

Learning by reduction principle (ReGLiS)

Example with $k = l = 1$ on training set: $S = \{$ ◁ I see a dog in the street ▷,
◁ I see a cat in the street ▷,
◁ I look at a bird ▷,
◁ We look at the dog ▷,
◁ I watch through the window ▷ $\}$

Step 1: build bottom grammar $G_0 = \langle \Sigma, N, P, N_0 \rangle$:

```
/* Axiom */
N0 -> I see a dog in the street | I see a cat in the street | I look at a bird
     | We look at the dog | I watch through the window
/* Substitutability classes */
N1 -> I | We
N2 -> I see | I look at
N3 -> see | look at
N4 -> I see a | We look at the
N5 -> see a
N6 -> a
N7 -> I see a dog | I see a cat
N8 -> see a dog | see a cat
N9 -> a dog | a cat
N28 -> dog | street | cat | window
N11 -> I see a dog in | I see a cat in | We look at | I watch through
N12 -> see a dog in | see a cat in | watch through
```

Learning by reduction principle (ReGLiS)

Example with $k = l = 1$ on training set: $S = \{$ ◁ I see a dog in the street ▷,
◁ I see a cat in the street ▷,
◁ I look at a bird ▷,
◁ We look at the dog ▷,
◁ I watch through the window ▷ $\}$

Step 1: build bottom grammar $G_0 = \langle \Sigma, N, P, N_0 \rangle$:

```
/* Axiom */
N0 -> I see a dog in the street | I see a cat in the street | I look at a bird
     | We look at the dog | I watch through the window
/* Substitutability classes */
N1 -> I | We
N2 -> I see | I look at
N3 -> see | look at
N4 -> I see a | We look at the
N5 -> see a
N6 -> a
N7 -> I see a dog | I see a cat
N8 -> see a dog | see a cat
N9 -> a dog | a cat
N28 -> dog | street | cat | window
N11 -> I see a dog in | I see a cat in | We look at | I watch through
N12 -> see a dog in | see a cat in | watch through
```

Learning by reduction principle (ReGLiS)

Step 2: generalize bottom grammar by reduction of rhs:

Main idea

Replace each $A \to \ldots \beta \ldots$ by $A \to \ldots B \ldots$
when $\exists B \in N, B \Rightarrow^* \beta$

- Simple: `N9 -> a dog` is reduced into `N9 -> a N28`
  since `N28 -> dog | street | cat | window`

Learning by reduction principle (ReGLiS)

Step 2: generalize bottom grammar by reduction of rhs:

Main idea

Replace each $A \to \ldots \beta \ldots$ by $A \to \ldots B \ldots$
when $\exists B \in N, B \Rightarrow^* \beta$

- Simple: `N9 -> a dog` is reduced into `N9 -> a N28` since `N28 -> dog | street | cat | window`
- Useless: Reducing `N9 -> a N28` into `N9 -> N6 N28` from `N6 -> a` (no generalization).

Learning by reduction principle (ReGLiS)

Step 2: generalize bottom grammar by reduction of rhs:

Main idea

Replace each $A \to \ldots \beta \ldots$ by $A \to \ldots B \ldots$
when $\exists B \in N, B \Rightarrow^* \beta$

- Simple: `N9 -> a dog` is reduced into `N9 -> a N28` since `N28 -> dog | street | cat | window`
- Useless: Reducing `N9 -> a N28` into `N9 -> N6 N28` from `N6 -> a` (no generalization). `N6` is useless...

Learning by reduction principle (ReGLiS)

Step 2: generalize bottom grammar by reduction of rhs:

Main idea

Replace each $A \to \dots \beta \dots$ by $A \to \dots B \dots$
when $\exists B \in N, B \Rightarrow^* \beta$

- Simple: `N9 -> a dog` is reduced into `N9 -> a N28` since `N28 -> dog | street | cat | window`
- Useless: Reducing `N9 -> a N28` into `N9 -> N6 N28` from `N6 -> a` (no generalization). `N6` is useless...
- Take care of overlaps: `N0 -> N1 N3 a N25`, the reduction of `N0 -> I look at a bird`, is reduced into `N0 -> N11 N36 | N4 N25 | N1 N23`
  when: `N11 -> N1 N3 | ...` ; `N36 -> a N25 | ...` ; `N4 -> N1 N3 a | ...` ; `N23 -> N3 a N25 | ...`

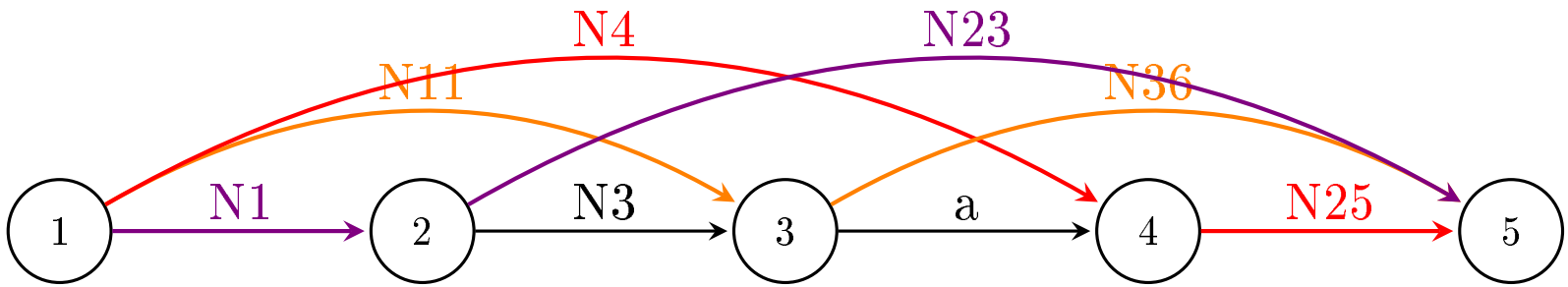
Learning by reduction principle (ReGLiS)

Step 2: generalize bottom grammar by reduction of rhs:

Main idea

Replace each $A \to \ldots \beta \ldots$ by $A \to \ldots B \ldots$
when $\exists B \in N, B \Rightarrow^* \beta$

- Simple: `N9 -> a dog` is reduced into `N9 -> a N28` since `N28 -> dog | street | cat | window`
- Useless: Reducing `N9 -> a N28` into `N9 -> N6 N28` from `N6 -> a` (no generalization). `N6` is useless...
- Take care of overlaps: `N0 -> N1 N3 a N25`, the reduction of `N0 -> I look at a bird`, is reduced into `N0 -> N11 N36 | N4 N25 | N1 N23`
  when: `N11 -> N1 N3 | ...` ; `N36 -> a N25 | ...` ; `N4 -> N1 N3 a | ...` ; `N23 -> N3 a N25 | ...`



A first contribution: characterization of reduction's result

A CFG $\langle \Sigma, N, P, N_0 \rangle$ is in *reduced normal form* (RNF) if:

1. Each non-terminal represents exactly one congruence class
$$\forall A \in N,\ L(A) = [L(A)]_L$$
2. Each non-terminal, other than the axiom, has alternative derivations
$$\forall A \in N, A \neq N_0 :\ |\{(A \to \alpha) \in P\}| > 1$$

3,4. P is the set of fully reduced rules

$\forall A, B \in N$:

3. $(B \Rightarrow_G^* \delta\alpha\gamma \wedge A \Rightarrow_G^* \alpha) \Longrightarrow (\exists (B \to \delta' A \gamma') \in P, \delta' \Rightarrow_G^* \delta \wedge \gamma' \Rightarrow_G^* \gamma)$
(existence of reduced rules)
4. $\forall (A \to \alpha), (B \to \beta) \in P$: $L(\alpha) \subseteq L(\beta) \implies A = B \wedge \alpha = \beta$
(only fully reduced rules)

- No assumption on the class of languages
- Canonical form for local substitutable languages
  - Substitutable languages with a finite set of prime [Cla13] are LS?
  - Interesting for strongly-congruential grammars [Sci14]?

$\mathcal{G}$ = CFG in RNF

Polynomial running time?

ReGLiS
Generalize grammar

/* Detect substitutability classes */
$\mathcal{C}_S \leftarrow$ `substitutability_classes`(S, k, l)
/* Discard classes $\rightsquigarrow$ NT with unique derivation */
$\mathcal{P}_S \leftarrow \{C \in \mathcal{C}_S : \forall C_1, C_2 \in \mathcal{C}_S, C \not\subseteq C_1\, C_2\}$ /* keep only $\mathcal{C}_S$-Prime */
/* Build bottom grammar */
$N \leftarrow \{N_C : C \in \mathcal{P}_S\}$; $P \leftarrow \{N_C \rightarrow y : N_C \in N, y \in C\}$
/* **Generalize grammar by reduction** */
$R \leftarrow P$;
repeat
  $P \leftarrow R$; $R \leftarrow \emptyset$;
  foreach $(N_C \rightarrow \alpha) \in P$, ordered by increasing$|\alpha|$ **do**
    /* Reduce rule */
    $PG \leftarrow$ `build_parsing_graph`(α, P)
    foreach $\beta \in$ `non_redundant_rhs`(PG) **do**
      if $\beta \notin N$ **then**
        $R \leftarrow R \cup \{N_C \rightarrow \beta\}$
until $R{=}P$
return $\langle \Sigma, N, R, N_{[S]} \rangle$

Focus on core of ReGLiS

Generalize grammar

/* Detect substitutability classes */

$\mathcal{C}_S \leftarrow$ `substitutability_classes`(S, k, l)

/* Discard classes $\rightsquigarrow$ NT with unique derivation */

$\mathcal{P}_S \leftarrow \{C \in \mathcal{C}_S : \forall C_1, C_2 \in \mathcal{C}_S, C \not\subseteq C_1\, C_2\}$ /* keep only $\mathcal{C}_S$-Prime */

/* Build bottom grammar */

$N \leftarrow \{N_C : C \in \mathcal{P}_S\}$; $P \leftarrow \{N_C \rightarrow y : N_C \in N, y \in C\}$

/* **Generalize grammar by reduction** */

$R \leftarrow \emptyset$;

foreach $(N_C \rightarrow \alpha) \in P$, ordered by increasing$|\alpha|$ **do**

  /* Reduce rule */

  $PG \leftarrow$ `build_parsing_graph`(α, P)

  foreach $\beta \in$ `non_redundant_rhs`(PG) **do**

    if $\beta \notin N$ **then**

      $R \leftarrow R \cup \{N_C \rightarrow \beta\}$

return $\langle \Sigma, N, R, N_{[S]} \rangle$

Focus on core of ReGLiS

Generalize grammar

/* Detect substitutability classes */

$\mathcal{C}_S \leftarrow$ `substitutability_classes`(S, k, l)

/* Discard classes $\rightsquigarrow$ NT with unique derivation */

$\mathcal{P}_S \leftarrow \{C \in \mathcal{C}_S : \forall C_1, C_2 \in \mathcal{C}_S, C \not\subseteq C_1\, C_2\}$ /* keep only $\mathcal{C}_S$-Prime */

/* Build bottom grammar */

$N \leftarrow \{N_C : C \in \mathcal{P}_S\}$; $P \leftarrow \{N_C \rightarrow y : N_C \in N, y \in C\}$

/* **Generalize grammar by reduction** */

$R \leftarrow \emptyset$;

foreach $(N_C \rightarrow \alpha) \in P$, ordered by increasing$|\alpha|$ **do**

  /* Reduce rule */

  foreach $\beta \in$ `reduce_rhs`(α, R) **do**

    if $\beta \notin N$ **then**

      $R \leftarrow R \cup \{N_C \rightarrow \beta\}$

return $\langle \Sigma, N, R, N_{[S]} \rangle$

Focus on core of ReGLiS

Generalize grammar

/* Detect substitutability classes */

$\mathcal{C}_S \leftarrow$ `substitutability_classes`(S, k, l)

/* Discard classes $\rightsquigarrow$ NT with unique derivation */

$\mathcal{P}_S \leftarrow \{C \in \mathcal{C}_S : \forall C_1, C_2 \in \mathcal{C}_S, C \not\subseteq C_1\, C_2\}$ /* keep only $\mathcal{C}_S$-Prime */

/* Build bottom grammar */

$N \leftarrow \{N_C : C \in \mathcal{P}_S\}$; $P \leftarrow \{N_C \rightarrow y : N_C \in N, y \in C\}$

/* **Generalize grammar by reduction** */

$R \leftarrow \emptyset$;

foreach $(N_C \rightarrow \alpha) \in P$, ordered by increasing$|\alpha|$ **do**

  /* Reduce rule */

  foreach $\beta \in$ `reduce_rhs`(α, R) **do**

    if $\beta \notin N$ **then**

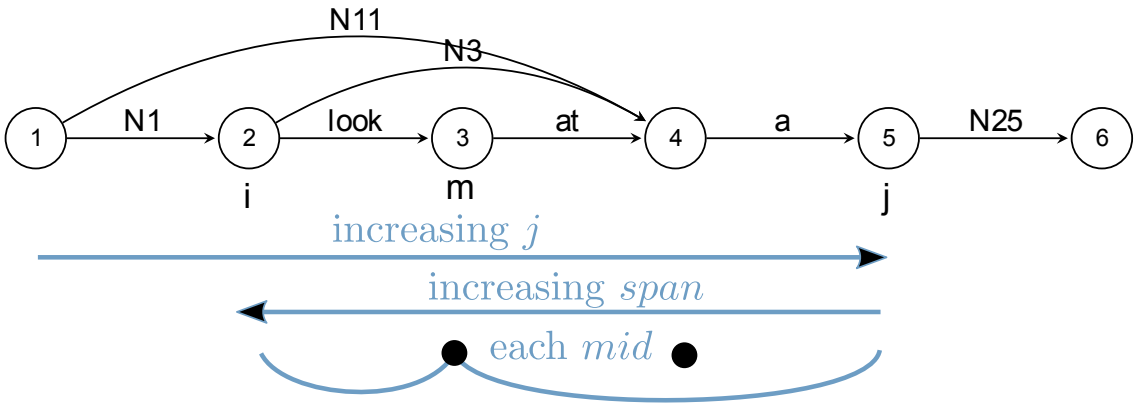
      $R \leftarrow R \cup \{N_C \rightarrow \beta\}$

return $\langle \Sigma, N, R, N_{[S]} \rangle$

Build parsing graph and reduce rule at once

`reduce_rhs`(α, R)

$\alpha =$ `I look at a bird`



IPaths$[i, j]$:

irreducible paths from i to j

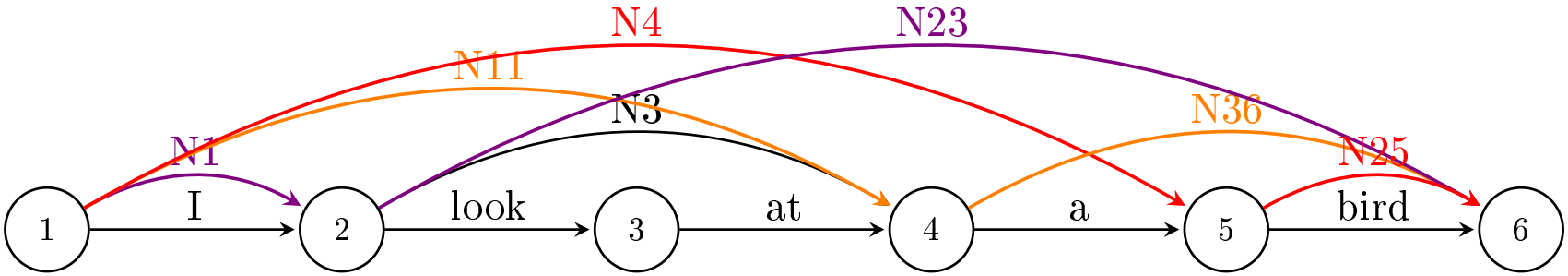
j \ i	1	2	3	4	5
2	(1,2)				
3	(1,2,3)	(2,3)			
4	(1,4)	(2,4)	(3,4)		
5	-	(2,4,5)	(3,4,5)	(4,5)	
6	-	-	-	-	(5,6)

- Dynamic programming algorithm ($\sim$ chart parsing)
  Parsing edges and irreducible paths[7] up to j from:
  - Parsing edges and irreducible paths up to $j - 1$
  - Smaller reduced rules

[7] path encoding a fully reduced rule

Build parsing graph and reduce rule at once

`reduce_rhs`(α, R)



Irreducible paths from 1 to 6 are $(1,2,6)$, $(1,4,6)$ and $(1,5,6)$

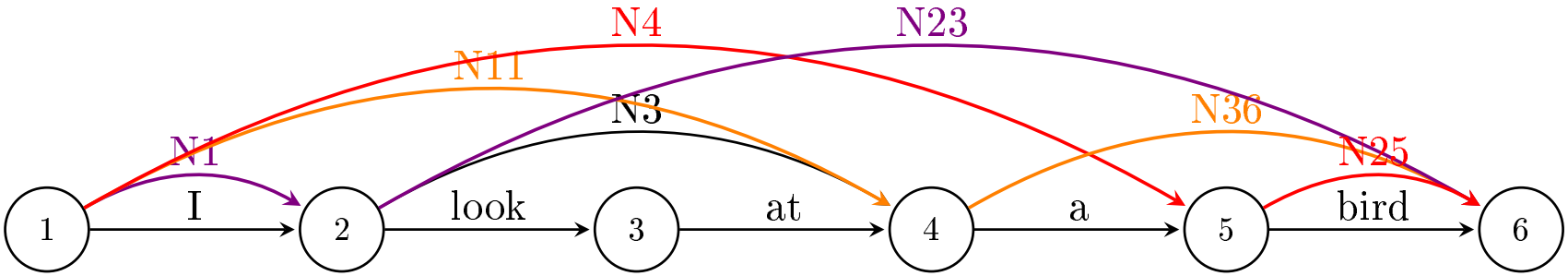
$(1,2,4,6)$ is reducible since it is a subset of $(1,4,6)$

- Dynamic programming algorithm ($\sim$ chart parsing)
  Parsing edges and irreducible paths[7] up to j from:
  - Parsing edges and irreducible paths up to $j-1$
  - Smaller reduced rules

[7] path encoding a fully reduced rule

Build parsing graph and reduce rule at once

`reduce_rhs`(α, R)



Irreducible paths from 1 to 6 are $(1,2,6)$, $(1,4,6)$ and $(1,5,6)$

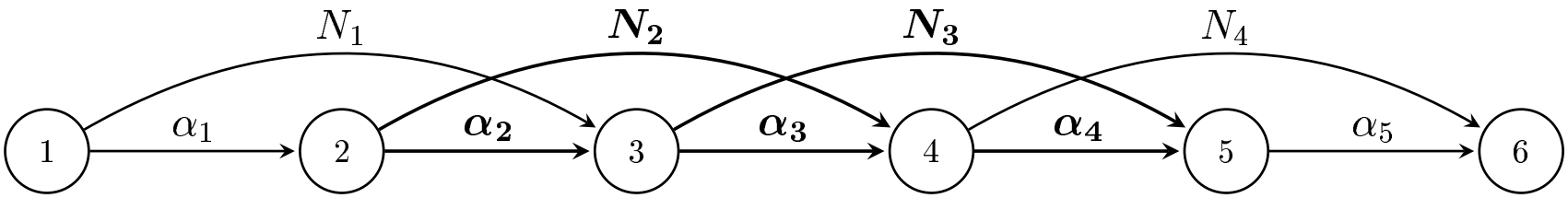
$(1,2,4,6)$ is reducible since it is a subset of $(1,4,6)$

- Dynamic programming algorithm ($\sim$ chart parsing)
  Parsing edges and irreducible paths[7] up to j from:
  - Parsing edges and irreducible paths up to $j-1$
  - Smaller reduced rules
- Complete parsing instead of syntactic matching
- Will enable us to ensure polynomial time...

[7] path encoding a fully reduced rule

Ensuring polynomial-time

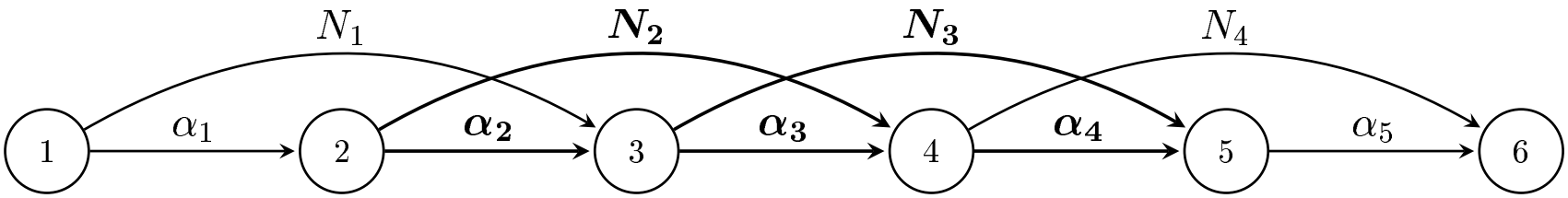
Parsing graph generating a non-polynomial number of reduced rules:



$|\mathsf{IPaths}[1, i]| = |\mathsf{IPaths}[1, i-1]| + |\mathsf{IPaths}[1, i-2]| = F(i-1)$ (with $F(0) = 0, F(1) = 1$)

Ensuring polynomial-time

Parsing graph generating a non-polynomial number of reduced rules:

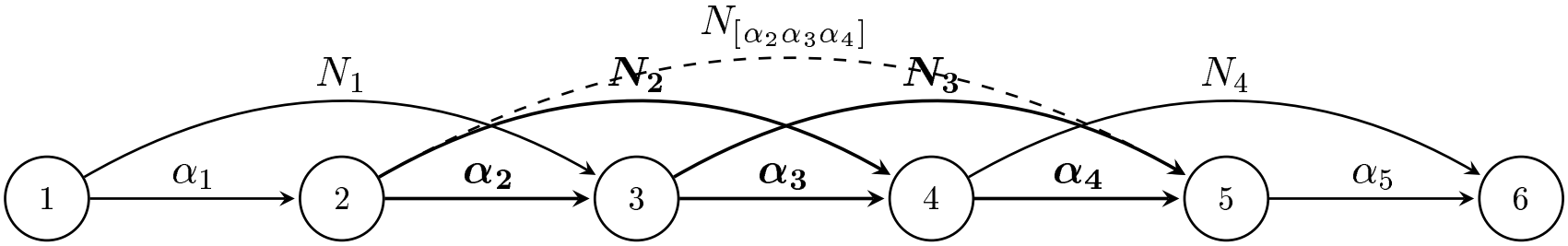


$|\mathsf{IPaths}[1,i]| = |\mathsf{IPaths}[1,i-1]| + |\mathsf{IPaths}[1,i-2]| = F(i-1)$ (with $F(0)=0, F(1)=1$)

- This is **not** the parsing graph for a substitutable language!
  Implies $N_{[\alpha_2\alpha_3\alpha_4]} \to N_2\alpha_4|\alpha_2 N_3$ and thus $[\alpha_2\alpha_3\alpha_4]$ prime: there should be an edge between 2 and 4 (or the edge for N_2 or N_3 is erroneous).

Ensuring polynomial-time

Parsing graph generating a non-polynomial number of reduced rules:

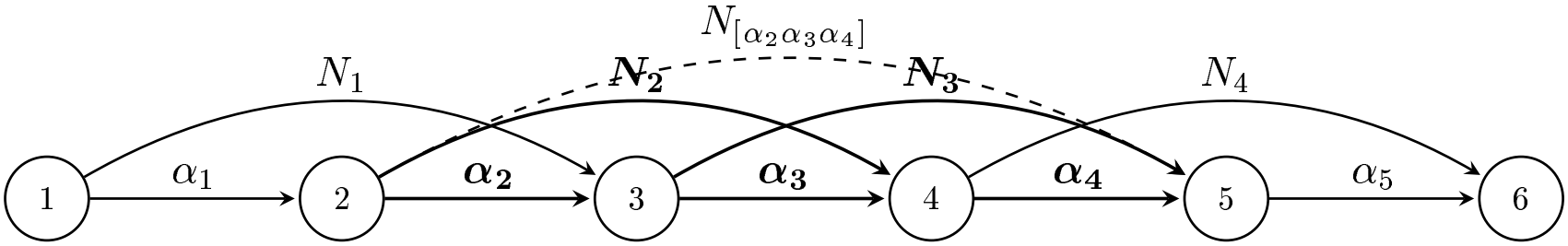


$|\mathsf{IPaths}[1,i]| = |\mathsf{IPaths}[1,i-1]| + |\mathsf{IPaths}[1,i-2]| = F(i-1)$ (with $F(0)=0, F(1)=1$)

- This is **not** the parsing graph for a substitutable language!
  Implies $N_{[\alpha_2\alpha_3\alpha_4]} \to N_2\alpha_4|\alpha_2 N_3$ and thus $[\alpha_2\alpha_3\alpha_4]$ prime: there should be an edge between 2 and 4 (or the edge for N_2 or N_3 is erroneous).

Ensuring polynomial-time

Parsing graph generating a non-polynomial number of reduced rules:



$|\mathsf{IPaths}[1,i]| = |\mathsf{IPaths}[1,i-1]| + |\mathsf{IPaths}[1,i-2]| = F(i-1)$ (with $F(0)=0, F(1)=1$)

- This is **not** the parsing graph for a substitutable language!
  Implies $N_{[\alpha_2\alpha_3\alpha_4]} \to N_2\alpha_4|\alpha_2 N_3$ and thus $[\alpha_2\alpha_3\alpha_4]$ prime: there should be an edge between 2 and 4 (or the edge for N_2 or N_3 is erroneous).

More generally

If an *intermediate* state is involved in more than one irreducible path, then the language is not substitutable. Otherwise, the parsing graph is said nice (for learning substitutable languages).

We can consider only nice graphs to ensure polynomial time...

Handling "bad" training samples

Bad parsing graphs

- When more than one irreducible path to (or from) an intermediate state have to be stored, raise an exception
- Exception handling is drastic here: stop everything and return trivial RNF grammar of training sample S: $\langle \Sigma, \{N_0\}, \{N_0 \to w : w \in S\}, N_0 \rangle$
  (*"Wait/ask for a more complete sample"*)
- Then the number of irreducible paths between *start* and *end* positions is at most the number of intermediate positions.
- For a training sample S with n sequences of maximal length m:

  Time complexity of $\text{ReGLiS}_{\text{core}}$ is $\mathcal{O}(nm^5)$

Non-RNF grammars

- The same strategy of raising an exception is used to deal with non-RNF grammars

Thank God, there are also "good" training samples

Let $G_R(L) = \langle \Sigma, N, P, N_0 \rangle$ denote the canonical RNF grammar of L in LS

Characteristic sample for L

$$CS(L) = \{uwv \in \Sigma^* : (A \to \alpha) \in P,\ (u, v) = c(A),\ w = w(\alpha)\}$$

where
$w(\alpha)$: smallest string generated from rsh α
$c(N)$: smallest context enabling to reach a non-terminal N from N_0

$CS(L)$ is polynomial wrt the size $|G_R(L)|$ and the thickness $\tau_{G_R(L)}$.
+ ReGLiS$_{\text{core}}$ returns a RNF grammar consistent with S in $\mathcal{O}(nm^5)$:

IPTtD Result

k, l local substitutable languages are IPTtD using RNF context-free grammars by reduction

Conclusion, perspectives and questions

Contributions

- Reduced normal form (RNF)
- ReGLiS$_{\text{core}}$ (Python code soon at `http://people.rennes.inria.fr/Francois.Coste/reglis2/`)
  - Single pass algorithm, building parsing graph and reducing rhs together
  - Polynomial time by limitation of the number of reductions
  - Raise errors on bad training samples to ensure returning a RNF
- IPTtD result wrt RNF by reduction

Perspectives

- IPTD result? using only RNF properties, or using also k, l-LS properties?
  - New insights for improving IPTxxD?

    (No compression, pol. data wrt smallest word of congruence classes. . .)
  - Do we need to return a grammar in $\mathcal{R}$ before convergence?
- Back to more practical algorithms
  - Couldn't we fix the parsing graph instead of stopping generalization?
  - We could also use this detection to interact with the user or oracle. . .
- Better/complete characterization of LS?

  and their links with substitutable languages with a finite set of prime?
- Learn by reduction other languages than the substitutable ones. . .

Other questions?

Bibliography I

[CE07] A. Clark and R. Eyraud. "Polynomial Identification in the Limit of Substitutable Context-free Languages". In: *Journal of Machine Learning Research* 8 (2007), pp. 1725–1745.

[Yos08] R. Yoshinaka. "Identification in the Limit of k,l-Substitutable Context-Free Languages". In: *ICGI*. Vol. 5278. Lecture Notes in Computer Science. Springer, 2008, pp. 266–279.

[Hig97] C. de la Higuera. "Characteristic Sets for Polynomial Grammatical Inference". In: *Machine Learning* 27.2 (1997), pp. 125–138.

[EHY16] R. Eyraud, J. Heinz, and R. Yoshinaka. "Efficiency in the Identification in the Limit Learning Paradigm". In: *Topics in Grammatical Inference*. Ed. by J. Heinz and J. M. Sempere. Springer-Verlag Berlin Heidelberg, 2016.

Bibliography II

[Cla13] A. Clark. "Learning trees from strings: a strong learning algorithm for some context-free grammars". In: *Journal of Machine Learning Research* 14.1 (2013), pp. 3537–3559.

[Sci14] J. Scicluna. "Grammatical inference of probalistic context-free grammars". PhD thesis. Nantes University, 2014.