GRAMMATICAL INFERENCE OF LAMBDA-CONFLUENT CONTEXT REWRITING SYSTEMS

Peter Černo
Department of Computer Science
Charles University in Prague, Faculty of Mathematics and Physics
Table of Contents

• Part I: Motivation,
• Part II: Definitions,
• Part III: Learning Algorithm,
• Part IV: Results,
• Part V: Concluding Remarks.
Part I
Motivation
Part I: Motivation

• Let’s say that we have the following sentence:

  Andrej, Monika and Peter like kitesurfing.

• We would like to verify the syntactical correctness of this sentence.

• One way to do this is to use Analysis by Reduction.
Part I: Motivation

- Analysis by Reduction – Step-wise simplifications.

Andrej, Monika and Peter like kitesurfing.

Andrej and Peter like kitesurfing.

They like kitesurfing. 😊
Part I: Motivation

- But how can we learn these reductions?
Part I: Motivation

- Let’s say that we are lucky and have the following two sentences in our database:

  Andrej, Monika and Peter like kitesurfing.

  Andrej and Peter like kitesurfing.
Part I: Motivation

- From these two samples we can, for instance, infer the following instruction:

Andrej, Monika and Peter like kitesurfing.

- **Instruction:**

, Monika → λ
Part I: Motivation

- **But** is the instruction (\(\text{Monika} \rightarrow \lambda\)) **correct**?
Part I: Motivation

- **But** is the instruction (',Monika → λ) correct?
- **Probably not:**

Peter goes with Andrej, **Monika** stays at home, and ...

Peter goes with Andrej, **Andrej** stays at home, and ...
Part I: Motivation

- What we need to do is to **capture a context** in which the instruction \((\text{Monika} \rightarrow \lambda)\) is **applicable**:

Andrej, Monika, and Peter like kitesurfing.

Andrej and Peter like kitesurfing.
Part II
Definitions
Part II: Definitions

- **Context Rewriting System (CRS)**
  - Is a triple \( M = (\Sigma, \Gamma, I) \):
    - \( \Sigma \) … input alphabet,
    - \( \Gamma \) … working alphabet, \( \Gamma \supseteq \Sigma \),
    - \( \$ \) and \( \$ \) … sentinels, \( \$ \) \( \not\in \Gamma \),
    - \( I \) … finite set of instructions \((x, z \rightarrow t, y)\):
      - \( x \in \{\lambda, \$\}.\Gamma^* \) (left context)
      - \( y \in \Gamma^*.\{\lambda, \$\} \) (right context)
      - \( z \in \Gamma^+, z \neq t \in \Gamma^* \).
    - The **width of instruction** \( \varphi = (x, z \rightarrow t, y) \) is \(|\varphi| = |xzty|\).
Part II: Definitions – Rewriting

- \[ u z v \vdash_M u t v \text{ iff } \exists (x, z \rightarrow t, y) \in I : \]
- \( x \) is a suffix of \( \$ . u \) and \( y \) is a prefix of \( v . \$ . \)

\[ L(M) = \{ w \in \Sigma^* / w \vdash^*_M \lambda \} . \]
Part II: Definitions – Empty Word

- **Note**: For every \( CRS M \models \lambda \vdash^*_M \lambda \), hence \( \lambda \in L(M) \).
- Whenever we say that a \( CRS M \) recognizes a language \( L \), we always mean that \( L(M) = L \cup \{\lambda\} \).
- We simply *ignore the empty word* in this setting.
Part II: Definitions – Example

- $L = \{a^n cb^n \mid n > 0\} \cup \{\lambda\}$:
- $CRS M = (\{a, b, c\}, I)$,
- **Instructions** $I$ are:
  - $R1 = (a, acb \rightarrow c, b)$,
  - $R2 = (\$, acb \rightarrow \lambda, \$)$.
Part II: Definitions – Restrictions

- Context Rewriting Systems are too powerful.

- We consider the following restrictions:

1. **Length of contexts = constant \( k \).**
   - All instructions \( \varphi = (x, z \rightarrow t, y) \) satisfy:
     - \( x \in LC_k := \Gamma^k \cup \{\} . \Gamma^{\leq k-1} \) (left context)
     - \( y \in RC_k := \Gamma^k \cup \Gamma^{\leq k-1}.\{\}$\) (right context)
     - In case \( k = 0 \) we use \( LC_k = RC_k = \{\lambda\} \).
     - We use the notation: \( k\text{-CRS} \).

2. **Width of instructions \( \leq \) constant \( l \).
   - All instructions \( \varphi = (x, z \rightarrow t, y) \) satisfy:
     - \( |\varphi| = |xzty| \leq l \).
     - We use the notation: \( (k, l)\text{-CRS} \).
Part II: Definitions – Restrictions

- **Context Rewriting Systems** are too powerful.
- We consider the following **restrictions**:

3. ** Restrict instruction-rules** $z \rightarrow t$.  
   - There are too many possibilities:
   - All **instructions** $\varphi = (x, z \rightarrow t, y)$ satisfy:
     a) $t = \lambda$, (Clearing Restarting Automata)
     b) $t$ is a subword of $z$, (Subword-Clearing Restarting Automata)
     c) $|t| \leq 1$.

4. **Lambda-confluence**.
   - We restrict the whole model to be lambda-confluent.
   - Fast membership queries, undecidable verification.

- In addition, we assume no auxiliary symbols: $\Gamma = \Sigma$. 
Part III
Learning Algorithm
Part III: Learning Algorithm

• Consider a class $\mathcal{M}$ of restricted CRS.

• **Goal:** Learning $L(\mathcal{M})$ from informant.
  - Identify any hidden target CRS from $\mathcal{M}$ in the limit from positive and negative samples.

• **Input:**
  - Set of positive samples $S^+$,
  - Set of negative samples $S$,
  - We assume that $S^+ \cap S = \emptyset$, and $\lambda \in S^+$.

• **Output:**
  - CRS $M$ from $\mathcal{M}$ such that: $L(M) \subseteq S^+$ and $L(M) \cap S = \emptyset$. 
Part III: Learning Restrictions

- **Without restrictions:**
  - *Trivial* even for *Clearing Restarting Automata*.
  - Consider: \( I = \{ (\$, w \rightarrow \lambda, \$) \mid w \in S^+, w \neq \lambda \} \).
  - Apparently: \( L(M) = S^+ \), where \( M = (\Sigma, \Sigma, I) \).

- **Therefore, we impose:**
  - An **upper limit** \( l \geq 1 \) on the **width of instructions**.
Part III: Learning Algorithm

Look at **Positive Samples** and Infer **Instruction Candidates**

Look also at **Negative Samples** and Remove **Bad Instructions**

**Simplify** and **Check Consistency**
Part III: Learning Algorithm Infer $\mathcal{M}$

- **Input:**
  - Positive samples $S^+$, negative samples $S$, $S^+ \cap S = \emptyset$, $\lambda \in S^+$.
  - Maximal width of instructions $l \geq 1$.
  - Specific length of contexts $k \geq 0$.

```
1  $\Phi \leftarrow$ Assumptions($S^+, k, l$);
2  while $\exists w_- \in S^-, w_+ \in S^+, \phi \in \Phi : w_- \vdash (\phi) w_+$ do
3    $\Phi \leftarrow \Phi \setminus \{\phi\}$;
4  if $\mathcal{M}$ is a class of $\lambda$-confluent models then
5    while $\exists w_+ \in S^+, w_- \in S^-, \phi \in \Phi : w_+ \vdash (\phi) w_-$ do
6      $\Phi \leftarrow \Phi \setminus \{\phi\}$;
7  $\Phi \leftarrow$ Simplify($\Phi$);
8  if Consistent($\Phi, S^+, S^-$) then
9    return Model $\mathcal{M}$ with the set of instructions $\Phi$;
10  Fail;
```
Part III: Learning Algorithm – Step 1/5

- **Input:**
  - **Positive samples** $S^+$, **negative samples** $S$, $S^+ \cap S = \emptyset$, $\lambda \in S^+$.
  - **Maximal width of instructions** $l \geq 1$,
  - **Specific length of contexts** $k \geq 0$.

```
1  $\Phi \leftarrow \text{Assumptions}(S^+, k, l);
2  \text{while } \exists w_- \in S^-, w_+ \in S^+, \phi \in \Phi : w_- \vdash (\phi) w_+ \text{ do}
3     \Phi \leftarrow \Phi \setminus \{\phi\};
4  \text{if } M \text{ is a class of } \lambda\text{-confluent models then}
5     \text{while } \exists w_+ \in S^+, w_- \in S^-, \phi \in \Phi : w_+ \vdash (\phi) w_- \text{ do}
6         \Phi \leftarrow \Phi \setminus \{\phi\};
7  \Phi \leftarrow \text{Simplify}(\Phi);
8  \text{if } \text{Consistent}(\Phi, S^+, S^-) \text{ then}
9      \text{return Model } M \text{ with the set of instructions } \Phi;
10  \text{Fail;}
```
Part III: Learning Algorithm – Step 1/5

Step 1:

1. $\Phi \leftarrow \text{Assumptions}(S^+, k, l)$;

- First, we obtain some set of instruction candidates.
- Let us assume, for a moment, that this set $\Phi$ already contains all instructions of the hidden target CRS.
Part III: Learning Algorithm – Step 2/5

Input:
- Positive samples $S^+$, negative samples $S$, $S^+ \cap S = \emptyset$, $\lambda \in S^+$.
- Maximal width of instructions $l \geq 1$.
- Specific length of contexts $k \geq 0$.

1. $\Phi \leftarrow \text{Assumptions}(S^+, k, l)$;
2. while $\exists w_-, w_+ \in S^-, S^+, \phi \in \Phi : w_- \vdash (\phi) w_+$ do
   3. $\Phi \leftarrow \Phi \setminus \{\phi\}$;
4. if $\mathcal{M}$ is a class of $\lambda$-confluent models then
   5. while $\exists w_+ \in S^+, w_- \in S^-, \phi \in \Phi : w_+ \vdash (\phi) w_-$ do
      6. $\Phi \leftarrow \Phi \setminus \{\phi\}$;
7. $\Phi \leftarrow \text{Simplify}(\Phi)$;
8. if $\text{Consistent}(\Phi, S^+, S^-)$ then
   9. return Model $\mathcal{M}$ with the set of instructions $\Phi$;
10. Fail;
Part III: Learning Algorithm – Step 2/5

- **Step 2:**

```latex
define two while loop
2: while \exists w_\neg \in S^- \land w_\pos \in S^+, \phi \in \Phi : w_\neg \vdash^{(\phi)} w_\pos do
3: \Phi \leftarrow \Phi \setminus \{\phi\};
```

- We gradually **remove all instructions** that allow a single-step reduction **from a negative sample to a positive sample**.
- Such instructions **violate** the so-called **error-preserving property**.
Part III: Learning Algorithm – Step 3/5

- **Input:**
  - *Positive samples* $S^+$, *negative samples* $S$, $S^+ \cap S = \emptyset$, $\lambda \in S^+$.
  - *Maximal width of instructions* $l \geq 1$.
  - *Specific length of contexts* $k \geq 0$.

```plaintext
1 $\Phi \leftarrow \text{Assumptions}(S^+, k, l);
2 \textbf{while } \exists w_- \in S^-, w_+ \in S^+, \phi \in \Phi : w_- \vdash (\phi) w_+ \textbf{ do}
3 \quad \Phi \leftarrow \Phi \setminus \{\phi\};
4 \textbf{if } \mathcal{M} \text{ is a class of } \lambda\text{-confluent models } \textbf{then}
5 \quad \textbf{while } \exists w_+ \in S^+, w_- \in S^-, \phi \in \Phi : w_+ \vdash (\phi) w_- \textbf{ do}
6 \quad \quad \Phi \leftarrow \Phi \setminus \{\phi\};
7 \Phi \leftarrow \text{Simplify}(\Phi);
8 \textbf{if } \text{Consistent}(\Phi, S^+, S^-) \textbf{ then}
9 \quad \textbf{return } \text{Model } \mathcal{M} \text{ with the set of instructions } \Phi;
10 \text{Fail;}
```
Part III: Learning Algorithm – Step 3/5

Step 3:

4 if $\mathcal{M}$ is a class of $\lambda$-confluent models then
5 while $\exists w_+ \in S^+, w_- \in S^-, \phi \in \Phi : w_+ \vdash (\phi) w_-$ do
6 $\Phi \leftarrow \Phi \setminus \{\phi\};$

- If the target class $\mathcal{M}$ consists of lambda-confluent CRS:
- We gradually remove all instructions that allow a single-step reduction from a positive sample to a negative sample.
- Such instructions violate the so-called correctness-preserving property.
Part III: Learning Algorithm – Step 4/5

**Input:**
- Positive samples $S^+$, negative samples $S$, $S^+ \cap S = \emptyset$, $\lambda \in S^+$.
- Maximal width of instructions $l \geq 1$.
- Specific length of contexts $k \geq 0$.

```plaintext
1 \( \Phi \leftarrow \text{Assumptions}(S^+, k, l) \);
2 \textbf{while} \( \exists w_- \in S^-, w_+ \in S^+, \phi \in \Phi : w_- \vdash \phi w_+ \) \textbf{do}
3 \hspace{1em} \Phi \leftarrow \Phi \setminus \{\phi\};
4 \textbf{if} \( M \) is a class of \( \lambda \)-confluent models \textbf{then}
5 \hspace{1em} \textbf{while} \( \exists w_+ \in S^+, w_- \in S^-, \phi \in \Phi : w_+ \vdash \phi w_- \) \textbf{do}
6 \hspace{2em} \Phi \leftarrow \Phi \setminus \{\phi\};
7 \Phi \leftarrow \text{Simplify}(\Phi);
8 \textbf{if} \ Consistent(\Phi, S^+, S^-) \textbf{then}
9 \hspace{1em} \textbf{return} Model \( M \) with the set of instructions \( \Phi \);
10 \text{Fail;}
```
Part III: Learning Algorithm – Step 4/5

• **Step 4:**

```plaintext
7 \( \Phi \leftarrow \text{Simplify}(\Phi) \);
```

• We *remove the redundant instructions*.
• This step is *optional* and *can be omitted* – it does not affect the properties or the correctness of the *Learning Algorithm*. 
Part III: Learning Algorithm – Step 5/5

- **Input:**
  - Positive samples $S^+$, negative samples $S$, $S^+ \cap S = \emptyset$, $\lambda \in S^+$.
  - Maximal width of instructions $l \geq 1$.
  - Specific *length of contexts* $k \geq 0$.

\[
\begin{align*}
1 & \quad \Phi \leftarrow \text{Assumptions}(S^+, k, l); \\
2 & \quad \textbf{while} \; \exists w_- \in S^-, w_+ \in S^+, \phi \in \Phi : w_- \vdash (\phi) \; w_+ \; \textbf{do} \\
3 & \quad \quad \Phi \leftarrow \Phi \setminus \{\phi\}; \\
4 & \quad \textbf{if} \; \mathcal{M} \; \text{is a class of } \lambda\text{-confluent models} \; \textbf{then} \\
5 & \quad \quad \textbf{while} \; \exists w_+ \in S^+, w_- \in S^-, \phi \in \Phi : w_+ \vdash (\phi) \; w_- \; \textbf{do} \\
6 & \quad \quad \quad \Phi \leftarrow \Phi \setminus \{\phi\}; \\
7 & \quad \Phi \leftarrow \text{Simplify}(\Phi); \\
8 & \quad \textbf{if} \; \text{Consistent}(\Phi, S^+, S^-) \; \textbf{then} \\
9 & \quad \quad \textbf{return} \; \text{Model } \mathcal{M} \; \text{with the set of instructions } \Phi; \\
10 & \quad \text{Fail};
\end{align*}
\]
Part III: Learning Algorithm – Step 5/5

- **Step 5:**

```plaintext
7 \( \Phi \leftarrow \text{Simplify}(\Phi) \);
8 \text{if } \text{Consistent}(\Phi, S^+, S^-) \text{ then}
9     \text{return } \text{Model } M \text{ with the set of instructions } \Phi;
10    \text{Fail;}
```

- We **check the consistency** of the remaining set of instructions with the given *input set of positive* and *negative samples*. 
Part III: Complexity

- Time complexity of the *Algorithm* depends on:
  - Time complexity of the *function Assumptions*,
  - Time complexity of the *simplification*,
  - Time complexity of the *consistency check*.

- There are *correct* implementations of the function *Assumptions* that run in polynomial time.

- The *simplification* and the *consistency check* can be done in polynomial time when using lambda-confluent *CRS*. Otherwise, it is an open problem.
Part III: Assumptions

• We call the function Assumptions correct, if it is possible to obtain all instructions of any hidden target CRS in the limit by using this function.

• To be more precise:
  • For every minimal $(k, l)$-CRS $M$ there exists a finite set $S_0^+ \subseteq L(M)$ such that for every $S^+ \supseteq S_0^+$ the $Assumptions(S^+, l, k)$ contains all instructions of $M$. 
Part III: Example – Assumptions_{weak}

- \textit{Assumptions}_{weak}(S^+, l, k) := all instructions \((x, z \rightarrow t, y)\) :
  - The length of contexts is \(k\):
    - \(x \in \Sigma^k \cup \{\epsilon\}, \Sigma^{\leq k-1}\) (left context)
    - \(y \in \Sigma^k \cup \Sigma^{\leq k-1}.\{\}$ (right context)
  - The width is bounded by \(l\):
    - \(|xzty| \leq l\).
  - The rule \(z \rightarrow t\) satisfies all rule restrictions.
  - There are two words \(w_1, w_2 \in S^+\) such that:
    - \(xzy\) is a subword of \(\epsilon w_1\$,
    - \(xty\) is a subword of \(\epsilon w_2\$.

- This function is \textit{correct} and runs in a \textit{polynomial time}. 
Part III: Example – Assumptions

Positive Samples

- \( a+a \)
- \( a+a+a \)
- \( a+(a)+a \)
- \( a+a+a+a \)
Part III: Example – Assumptions

Positive Samples

\[
\begin{align*}
\& a+a \\
\& (\& , a+ \rightarrow \lambda, a) \\
\& a+a+a \\
\& a+(a)+a \\
\& a+a+a+a
\end{align*}
\]
Part III: Example – Assumptions

Positive Samples

\((\$, \text{a+} \rightarrow \lambda, \text{a})\)

\((\$, \text{a+} \rightarrow \lambda, \text{a})\)

\((\$, \text{a+} \rightarrow \lambda, \text{a})\)

\((\$, \text{a+} \rightarrow \lambda, \text{a})\)
Part III: Example – Assumptions

Positive Samples

\[ a + a \]

\[ a + a + a \]

\[ a + (a) + a \]

\[ a + a + (a + a) \]

BAD Instruction

\[ (+, ( \rightarrow \lambda, a) \]
Part III: Example – Assumptions

Positive Samples

\[ a + a \]
\[ a + a + a \]
\[ a + (a) + a \]
\[ a + a + a + a \]
Part IV
Results
Part IV: Results

- $\mathcal{M}$ – class of restricted $(k, l)$-CRS,
- $\mathcal{M}$ – a model from $\mathcal{M}$,
- Then there exist:
  - Finite sets $S_0^+, S_0^-$ of positive, negative samples:
  - For every $S^+ \supseteq S_0^+, S^- \supseteq S_0^-$ consistent with $\mathcal{M}$:
    - $\text{Infer}_{\mathcal{M}}(S^+, S, k, l) = N \quad : \quad L(N) = L(M)$.
- Positive side:
  - The class $\mathcal{L}(\mathcal{M})$ is learnable in the limit from informant.
- Negative side:
  - $\text{size}(S_0^+, S_0^-)$ can be exponentially large w.r.t. $\text{size}(\mathcal{M})$.
  - We do not know $k, l$.
  - If $l$ is specified, $\mathcal{L}(\mathcal{M})$ is finite!
Part IV: Unconstrained Learning

- **Input:**
  - **Positive samples** $S^+$, **negative samples** $S$, $S^+ \cap S = \emptyset$, $\lambda \in S^+$.
  - Specific **length of contexts** $k \geq 0$.

```plaintext
1  for \ l = 1 \ldots \infty \ do \\
2     M \leftarrow \text{Infer}_M(S^+, S^-, k, l); \\
3     \text{if } M \neq \text{Fail then} \\
4     \quad \text{return } M;
```
Part IV: Results

- $\mathcal{M}$ – class of restricted $k$-CRS,
- $M$ – a model from $\mathcal{M}$,
- Then there exist:
  - Finite sets $S_0^+, S_0^-$ of positive, negative samples:
  - For every $S^+ \supseteq S_0^+$, $S^- \supseteq S_0^-$ consistent with $M$:
    - $\text{UnconstrainedInfer}_\mathcal{M}(S^+, S^- k) = N : L(N) = L(M)$.
  - $N$ has minimal width!

- Positive side:
  - The infinite class $\mathcal{L}(M)$ is learnable in the limit from informant.

- Negative side:
  - $\text{size}(S_0^+, S_0^-)$ can be exponentially large w.r.t. $\text{size}(M)$.
  - We do not know $k$. 
Part V
Concluding Remarks
Part V: Concluding Remarks

• Remarks:

• We have shown that $L(ℳ)$ is learnable in the limit from informant for any class $ℳ$ of restricted $k$-CRS.

• $UnconstrainedInfer_ℳ(S^+, S^-, k)$ always returns a model consistent with the given input $S^+, S^-$. In the worst case it returns:

$$I = \{(\varepsilon, w \rightarrow \lambda, \$) \mid w \in S^+, w \neq \lambda\}.$$

• This is not true for $Infer_ℳ(S^+, S^-, k, l)$, (it can Fail). In some cases, finding a consistent model with maximal width $l$ is NP-hard.

• If $ℳ$ is a class of lambda-confluent $k$-CRS, then $UnconstrainedInfer$ runs in polynomial time w.r.t. size$(S^+, S^-)$.

• But in most cases, it is not possible to verify lambda-confluence. It is often not even recursively enumerable.

• If $ℳ$ is a class of ordinary $k$-CRS, the time complexity of $UnconstrainedInfer$ is an open problem.
Selected References

- M. Beaudry, M. Holzer, G. Niemann, and F. Otto. **Mcnaughton families of languages.**
- Ronald V Book and Friedrich Otto. **String-rewriting systems.**
  - Springer-Verlag, New York, NY, USA, 1993.
- Peter Černo. **Clearing restarting automata and grammatical inference.**
- Peter Černo and František Mráz. **Clearing restarting automata.**
- C. de la Higuera. **Grammatical Inference: Learning Automata and Grammars.**
- R. Eyraud, C. de la Higuera, and J.-C. Janodet. **Lars: A learning algorithm for rewriting systems.**
- E. Mark Gold. **Complexity of automaton identification from given data.**
  - Information and Control, 37, 1978.
- John E. Hopcroft and J. D. Ullman. **Formal Languages and their Relation to Automata.**
  - Addison-Wesley, Reading, 1969.
- S. Lange, T. Zeugmann, and S. Zilles. **Learning indexed families of recursive languages from positive data: A survey.**
- R. McNaughton. **Algebraic decision procedures for local testability.**
- F. Otto. **Restarting automata.**
- F. OTTO, F. MRAZ, **Lambda-Conuence is Undecidable for Clearing Restarting Automata.**
Thank You!

• This *presentation* is available on:

• An *implementation* of the algorithms can be found on:
  http://code.google.com/p/clearing-restarting-automata/