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 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.

Andrej and Peter like kitesurfing.

- **Instruction**:

, Monika $\rightarrow$ **λ**

Part I: Motivation

- **But** is the instruction **(,Monika $\rightarrow$ λ)** **correct**?

Part I: Motivation

- **But** is the instruction **(,Monika $\rightarrow$ λ)** **correct**?
- **Probably not:**

Peter goes with Andrej, Monika stays at home, and …

Peter goes with Andrej stays at home, and …

Part I: Motivation

- What we need to do is to **capture a context** in which the instruction **(,Monika** → **λ)** is **applicable**:

Andrej, Monika and Peter like kitesurfing.

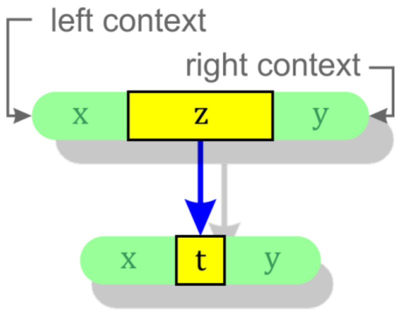
Andrej and Peter like kitesurfing.

¢ Andrej **, Monika** → **λ** **and Peter**

Part II
Definitions

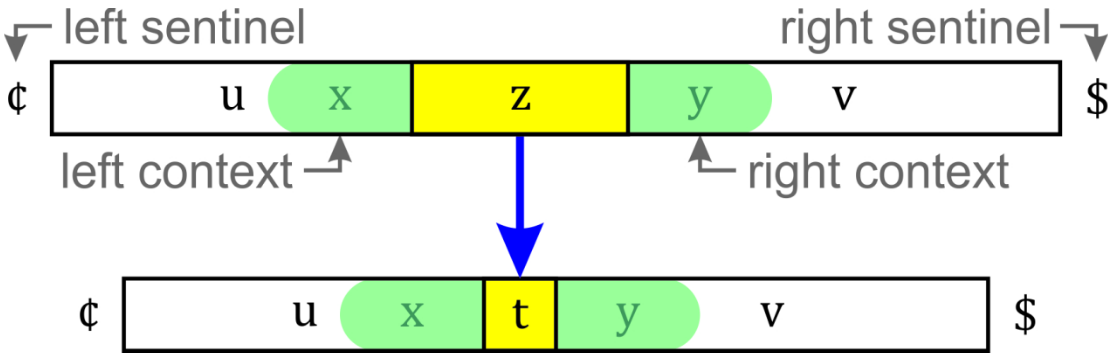
Part II: Definitions

- <u>**Context Rewriting System**</u> (***CRS***)
- **Is a triple** $M = (\Sigma, \Gamma, I)$:
  - Σ … ***input alphabet***,
  - Γ … ***working alphabet***, $\Gamma \supseteq \Sigma$,
  - ¢ and \$ … ***sentinels***, ¢, \$ $\notin \Gamma$,
  - 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

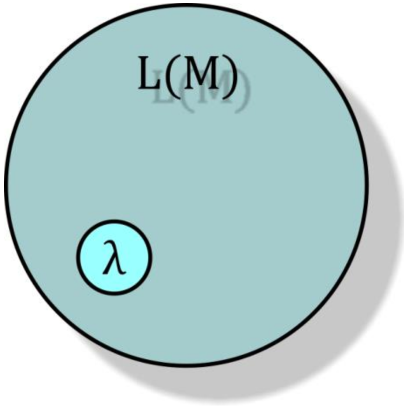
- $u\underline{z}v \vdash_M utv$ **iff** $\exists\,(x, z \to t, y) \in I$:
- x is a ***suffix*** of $¢.u$ **and** y is a ***prefix*** of $v.\$$.



$$L(M) = \{w \in \Sigma^* \mid w \vdash^*_M \lambda\}.$$

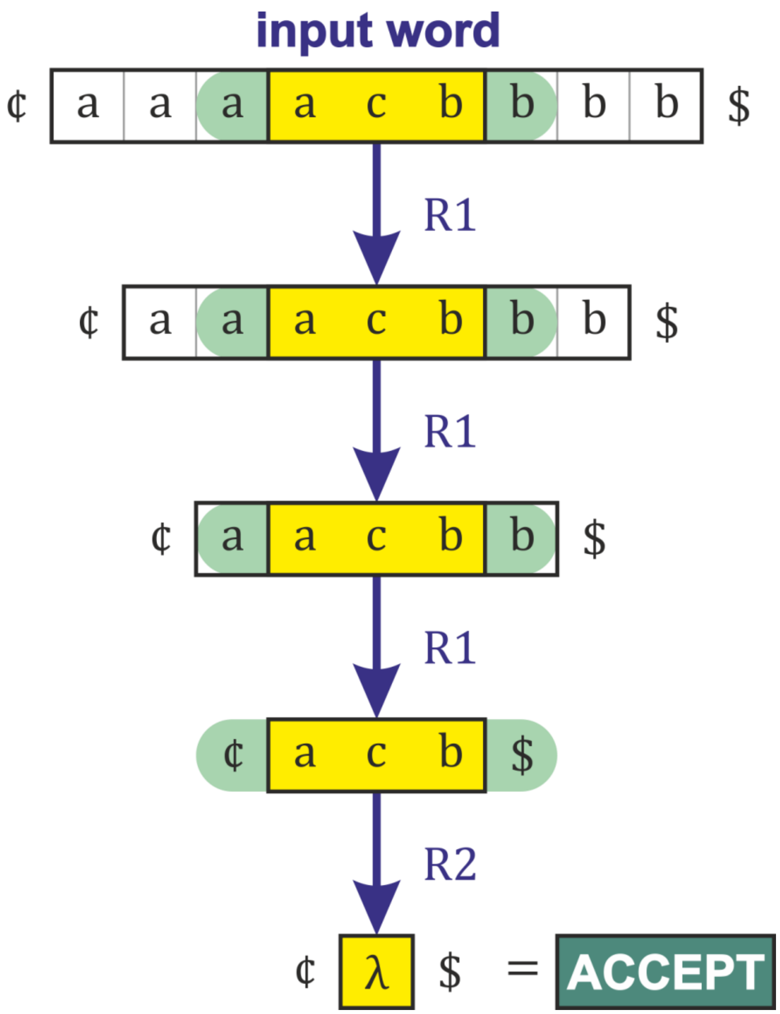
Part II: Definitions – Empty Word

- **<u>Note</u>**: For every *CRS M:* $\lambda \vdash^*_M \lambda$, 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

- $L = \{a^n c b^n \mid n > 0\} \cup \{\lambda\}$:
- $CRS\ M = (\{a, b, c\}, I)$,
- **Instructions** I are:
  - $R1 = (a, \underline{acb} \to c, b)$,
  - $R2 = (¢, \underline{acb} \to \lambda, \$)$.



Part II: Definitions – Restrictions

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

1. **Length of contexts = constant k.**
   - All ***instructions*** $\varphi = (x, z \to t, y)$ 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-CRS*.
2. **Width of instructions ≤ constant l.**
   - All ***instructions*** $\varphi = (x, z \to t, y)$ satisfy:
   - $|\varphi| = |xzty| \leq l$.
   - We use the **notation**: *(k, l)-CRS*.

Part II: Definitions – Restrictions

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

3. Restrict **instruction-rules** $z \to t$.
   - There are too many possibilities:
   - All ***instructions*** $\varphi = (x, z \to t, y)$ 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** $\mathcal{CRS}$.
- **Goal**: **Learning** $\mathcal{L}(\mathcal{M})$ from **informant**.
  - ***Identify*** any ***hidden target*** $\mathcal{CRS}$ 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**:
  - $\mathcal{CRS}\ M$ 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: $I = \{ (¢, w \to \lambda, \$) \mid w \in S^+ , w \neq \lambda \}$.
  - 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*** $k \geq 0$.

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

while $\exists w_- \in S^-, w_+ \in S^+, \phi \in \Phi : w_- \vdash^{(\phi)} w_+$ **do**

  $\Phi \leftarrow \Phi \setminus \{\phi\}$;

if $\mathcal{M}$ *is a class of* λ*-confluent models* **then**

  while $\exists w_+ \in S^+, w_- \in S^-, \phi \in \Phi : w_+ \vdash^{(\phi)} w_-$ **do**

    $\Phi \leftarrow \Phi \setminus \{\phi\}$;

$\Phi \leftarrow \mathsf{Simplify}(\Phi)$;

if $\mathsf{Consistent}(\Phi, S^+, S^-)$ **then**

  return *Model M with the set of instructions* Φ;

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*** $k \geq 0$.

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

while $\exists w_- \in S^-, w_+ \in S^+, \phi \in \Phi : w_- \vdash^{(\phi)} w_+$ **do**

  $\Phi \leftarrow \Phi \setminus \{\phi\}$;

if $\mathcal{M}$ *is a class of* λ*-confluent models* **then**

  while $\exists w_+ \in S^+, w_- \in S^-, \phi \in \Phi : w_+ \vdash^{(\phi)} w_-$ **do**

    $\Phi \leftarrow \Phi \setminus \{\phi\}$;

$\Phi \leftarrow \mathsf{Simplify}(\Phi)$;

if $\mathsf{Consistent}(\Phi, S^+, S^-)$ **then**

  return *Model* M *with the set of instructions* Φ;

Fail;

Part III: Learning Algorithm – Step 1/5

- **Step 1**:

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

  - First, we obtain some set of ***instruction candidates***.
  - Let us ***assume***, for a moment, that this set Φ already ***contains all instructions*** of the ***hidden target*** $\mathcal{CRS}$.

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*** $k \geq 0$.

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

while $\exists w_- \in S^-, w_+ \in S^+, \phi \in \Phi : w_- \vdash^{(\phi)} w_+$ **do**

  $\Phi \leftarrow \Phi \setminus \{\phi\}$;

if $\mathcal{M}$ *is a class of* λ*-confluent models* **then**

  while $\exists w_+ \in S^+, w_- \in S^-, \phi \in \Phi : w_+ \vdash^{(\phi)} w_-$ **do**

    $\Phi \leftarrow \Phi \setminus \{\phi\}$;

$\Phi \leftarrow \mathsf{Simplify}(\Phi)$;

if $\mathsf{Consistent}(\Phi, S^+, S^-)$ **then**

  return *Model* M *with the set of instructions* Φ;

Fail;

Part III: Learning Algorithm – Step 2/5

- **<u>Step 2</u>**:

$$\begin{array}{ll} 2 & \textbf{while } \exists w_- \in S^-, w_+ \in S^+, \phi \in \Phi : w_- \vdash^{(\phi)} w_+ \textbf{ do} \\ 3 & \quad \lfloor\ \Phi \leftarrow \Phi \setminus \{\phi\}; \end{array}$$

  - 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*** $k \geq 0$.

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

while $\exists w_- \in S^-, w_+ \in S^+, \phi \in \Phi : w_- \vdash^{(\phi)} w_+$ **do**

  $\Phi \leftarrow \Phi \setminus \{\phi\}$;

if $\mathcal{M}$ *is a class of* λ*-confluent models* **then**

  while $\exists w_+ \in S^+, w_- \in S^-, \phi \in \Phi : w_+ \vdash^{(\phi)} w_-$ **do**

    $\Phi \leftarrow \Phi \setminus \{\phi\}$;

$\Phi \leftarrow \mathsf{Simplify}(\Phi)$;

if $\mathsf{Consistent}(\Phi, S^+, S^-)$ **then**

  return *Model M with the set of instructions* Φ;

Fail;

Part III: Learning Algorithm – Step 3/5

- **<u>Step 3</u>**:

$$
\begin{array}{l}
\textbf{if } \mathcal{M} \textit{ is a class of } \lambda\textit{-confluent models } \textbf{then} \\
\quad \textbf{while } \exists w_+ \in S^+, w_- \in S^-, \phi \in \Phi : w_+ \vdash^{(\phi)} w_- \textbf{ do} \\
\quad\quad \Phi \leftarrow \Phi \setminus \{\phi\};
\end{array}
$$

  - **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*** $k \geq 0$.

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

while $\exists w_- \in S^-, w_+ \in S^+, \phi \in \Phi : w_- \vdash^{(\phi)} w_+$ **do**

  $\Phi \leftarrow \Phi \setminus \{\phi\}$;

if $\mathcal{M}$ *is a class of* λ*-confluent models* **then**

  while $\exists w_+ \in S^+, w_- \in S^-, \phi \in \Phi : w_+ \vdash^{(\phi)} w_-$ **do**

    $\Phi \leftarrow \Phi \setminus \{\phi\}$;

$\boxed{\Phi \leftarrow \text{Simplify}(\Phi);}$

if $\text{Consistent}(\Phi, S^+, S^-)$ **then**

  return *Model* M *with the set of instructions* Φ;

Fail;

Part III: Learning Algorithm – Step 4/5

- **Step 4**:

$$7 \quad \Phi \leftarrow \mathsf{Simplify}(\Phi);$$

  - 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*** $k \geq 0$.

```
Φ ← Assumptions(S⁺, k, l);
while ∃w₋ ∈ S⁻, w₊ ∈ S⁺, φ ∈ Φ : w₋ ⊢^(φ) w₊ do
    Φ ← Φ \ {φ};
if M is a class of λ-confluent models then
    while ∃w₊ ∈ S⁺, w₋ ∈ S⁻, φ ∈ Φ : w₊ ⊢^(φ) w₋ do
        Φ ← Φ \ {φ};
Φ ← Simplify(Φ);
if Consistent(Φ, S⁺, S⁻) then
    return Model M with the set of instructions Φ;
Fail;
```

Part III: Learning Algorithm – Step 5/5

- **Step 5**:

$\Phi \leftarrow \mathsf{Simplify}(\Phi);$
if $\mathsf{Consistent}(\Phi, S^{+}, S^{-})$ **then**
  return *Model M with the set of instructions* Φ;
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*** $\boldsymbol{Assumptions}$ **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$_{weak}$

- $\mathit{Assumptions}_{weak}(S^+, l, k)$:= all instructions $(x, z \rightarrow t, y)$:
  - **The length of contexts is k :**
    - $x \in \Sigma^k \cup \{¢\}.\Sigma^{\leq k-1}$ **(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 $¢ w_1 \$$,
    - xty is a ***subword*** of $¢ w_2 \$$.

- This function is ***correct*** and runs in a ***polynomial time***.

Part III: Example – Assumptions$_{\text{weak}}$

Positive Samples

¢ a+a $

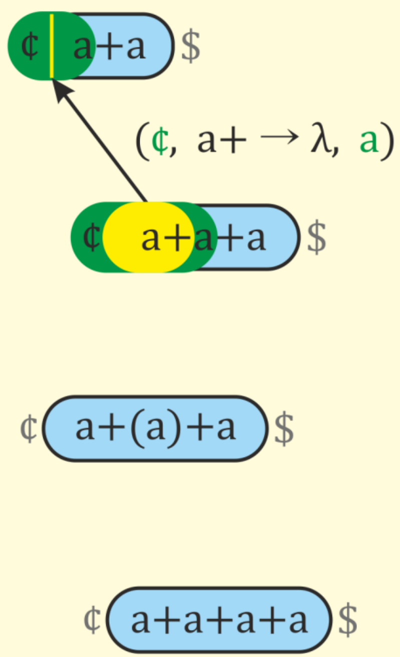
¢ a+a+a $

¢ a+(a)+a $

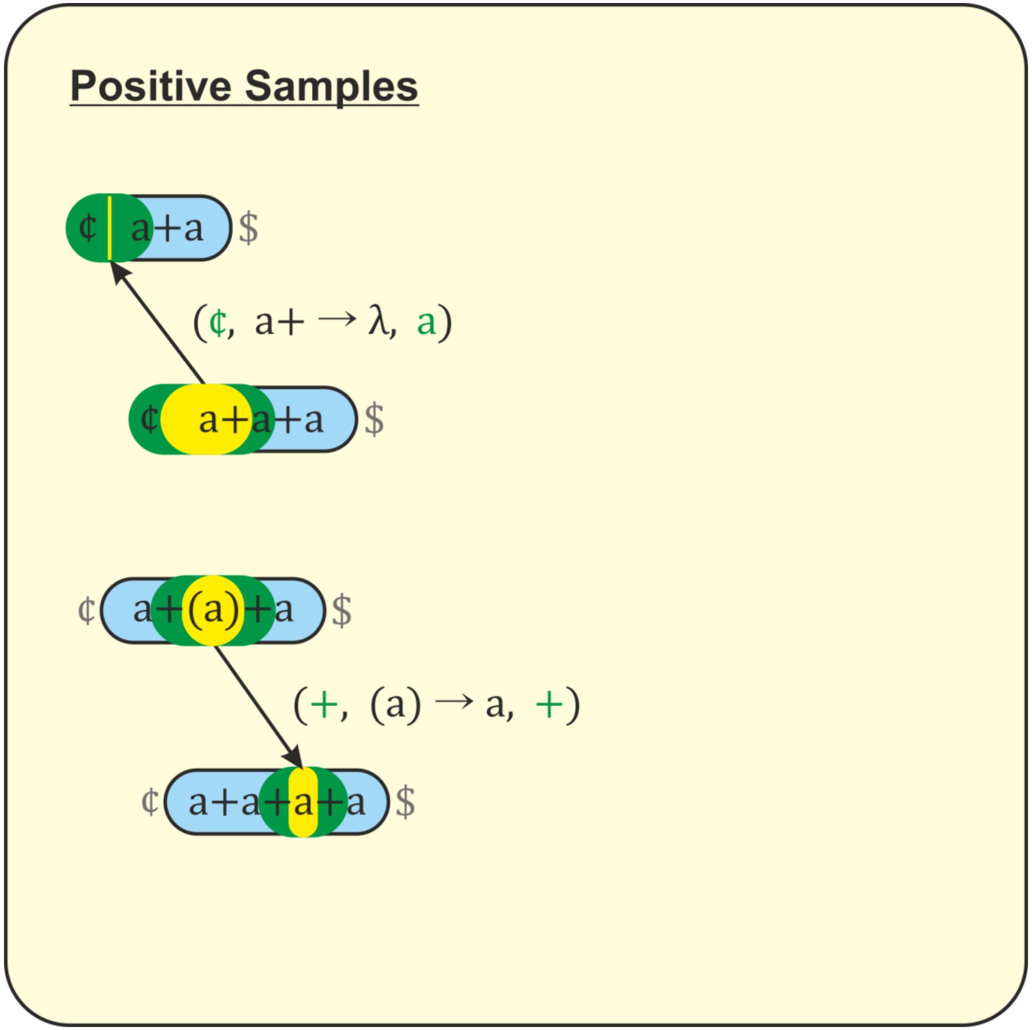
¢ a+a+a+a $

Part III: Example – Assumptions$_{\text{weak}}$

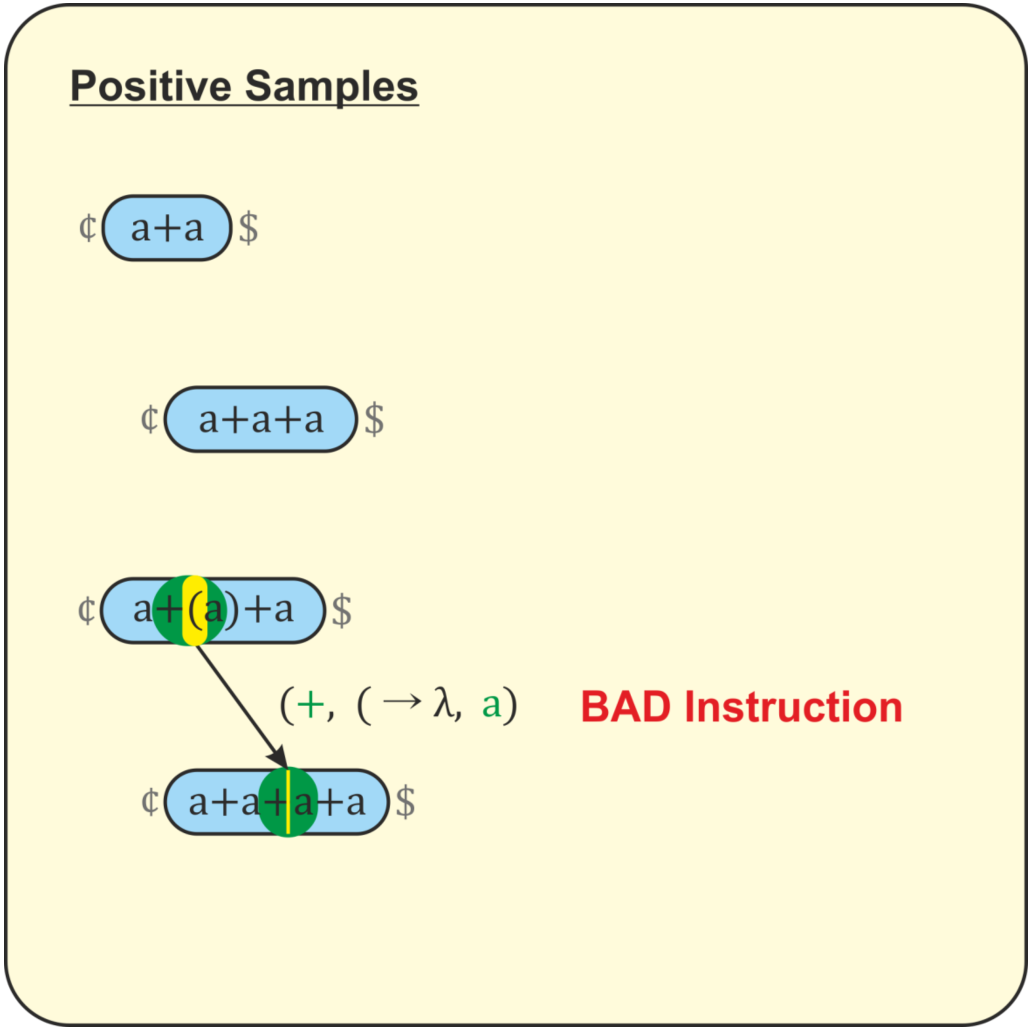
Positive Samples



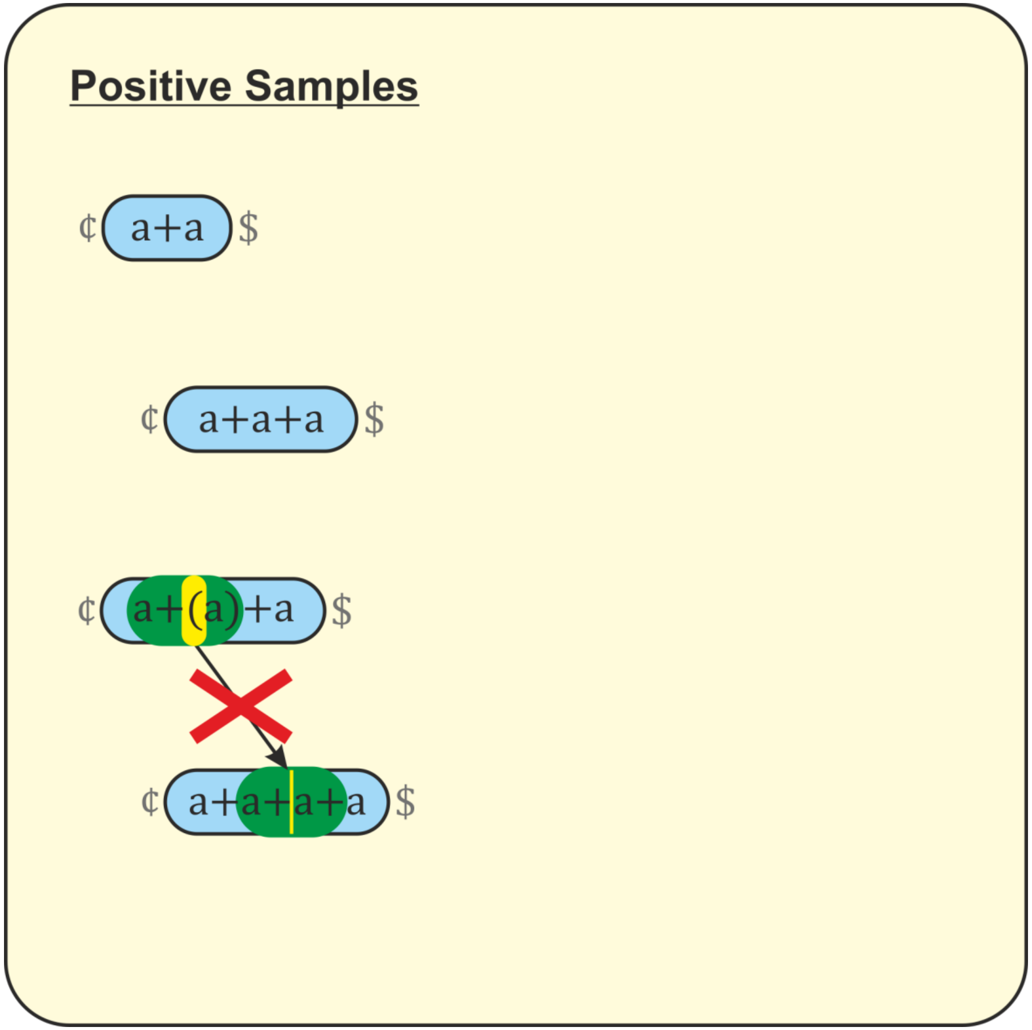
Part III: Example – Assumptions$_{\text{weak}}$



Part III: Example – Assumptions$_{weak}$



Part III: Example – Assumptions$_{\text{weak}}$



Part IV
Results

Part IV: Results

- $\mathcal{M}$ – class of restricted *(k, l)-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:
  - $Infer_{\mathcal{M}}(S^+, S^-, k, l) = N \ : \ L(N) = L(M)$.
- **Positive side**:
  - The class $\mathcal{L}(\mathcal{M})$ is **learnable in the limit** from **informant**.
- **Negative side**:
  - $size(S_0^+, S_0^-)$ can be exponentially large w.r.t. $size(M)$.
  - 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*** $k \geq 0$.

---

for $l = 1 \ldots \infty$ **do**
    $M \leftarrow \mathsf{Infer}_{\mathcal{M}}(S^+, S^-, k, l)$;
    if $M \neq$ **Fail then**
        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:
  - $UnconstrainedInfer_{\mathcal{M}}(S^+, S^-, k) = N \;:\; L(N) = L(M)$.
  - N has **minimal width**!
- **Positive side**:
  - The *infinite* class $\mathcal{L}(\mathcal{M})$ is **learnable in the limit** from **informant**.
- **Negative side**:
  - $size(S_0^+, S_0^-)$ can be exponentially large w.r.t. $size(M)$.
  - We do not know k.

Part V
Concluding Remarks

Part V: Concluding Remarks

- **Remarks**:
  - We have shown that $\mathcal{L}(\mathcal{M})$ is **learnable in the limit** from **informant** for any class $\mathcal{M}$ of restricted *k-CRS*.
  - $UnconstrainedInfer_{\mathcal{M}}(S^+, S^-, k)$ ***always returns*** a model consistent with the given input S^+, S^-. In the worst case it returns:

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

  - This is **not true** for $Infer_{\mathcal{M}}(S^+, S^-, k, l)$, (it can ***Fail***). In some cases, finding a consistent model with maximal width l is ***NP**-hard*.
  - If $\mathcal{M}$ 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 $\mathcal{M}$ 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**.
  - Theoretical Computer Science, 290(3):1581-1628, 2003.
- Ronald V Book and Friedrich Otto. **String-rewriting systems**.
  - Springer-Verlag, New York, NY, USA, 1993.
- Peter Černo. **Clearing restarting automata and grammatical inference**.
  - In: J. HEINZ, C. DE LA HIGUERA, T. OATES (eds.), Proceedings of the Eleventh International Conference on Grammatical Inference. JMLR Workshop and Conference Proceedings 21, 2012, 54-68.
- Peter Černo and František Mráz. **Clearing restarting automata**.
  - Fundamenta Informaticae, 104(1):17-54, 2010.
- C. de la Higuera. **Grammatical Inference: Learning Automata and Grammars**.
  - Cambridge University Press, New York, NY, USA, 2010.
- R. Eyraud, C. de la Higuera, and J.-C. Janodet. **Lars: A learning algorithm for rewriting systems**.
  - Machine Learning, 66:7-31, 2007.
- 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**.
  - Theor. Comput. Sci., 397(1-3):194-232, May 2008.
- R. McNaughton. **Algebraic decision procedures for local testability**.
  - Theory of Computing Systems, 8:60-76, 1974.
- F. Otto. **Restarting automata**.
  - In Zoltán Ésik, Carlos Martín-Vide, and Victor Mitrana, editors, Recent Advances in Formal Languages and Applications, volume 25 of Studies in Computational Intelligence, pages 269-303. Springer, Berlin, 2006.
- F. OTTO, F. MRAZ, **Lambda-Conuence is Undecidable for Clearing Restarting Automata**.
  - In: CIAA 2013, Proceedings. LNCS 7982, Berlin, 2013, 256-267.

Thank You!

- This ***presentation*** is available on:

  http://popelka.ms.mff.cuni.cz/cerno/files/cerno_gi_of_crs.pdf

- An ***implementation*** of the algorithms can be found on:

  http://code.google.com/p/clearing-restarting-automata/