CLEARING RESTARTING AUTOMATA AND GRAMMATICAL INFERENCE

Peter Černo

Department of Computer Science

Charles University in Prague, Faculty of Mathematics and Physics

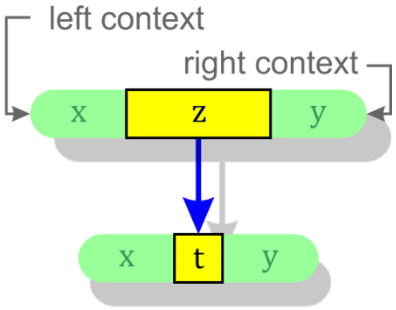
Table of Contents

Part I: Introduction

- **Restarting Automata**:
  - Model for the linguistic technique of ***analysis by reduction***.
  - Many different types have been defined and studied intensively.
- **Analysis by Reduction**:
  - Method for checking [non-]correctness of a sentence.
  - Iterative application of simplifications.
  - Until the input cannot be simplified anymore.
- **Restricted Models**:
  - *Clearing*, Δ-Clearing and Δ*-Clearing Restarting Automata,
  - *Subword-Clearing* Restarting Automata.
  - Our method is similar to the *delimited string-rewriting systems* *[Eyraud et al. (2007)]*.

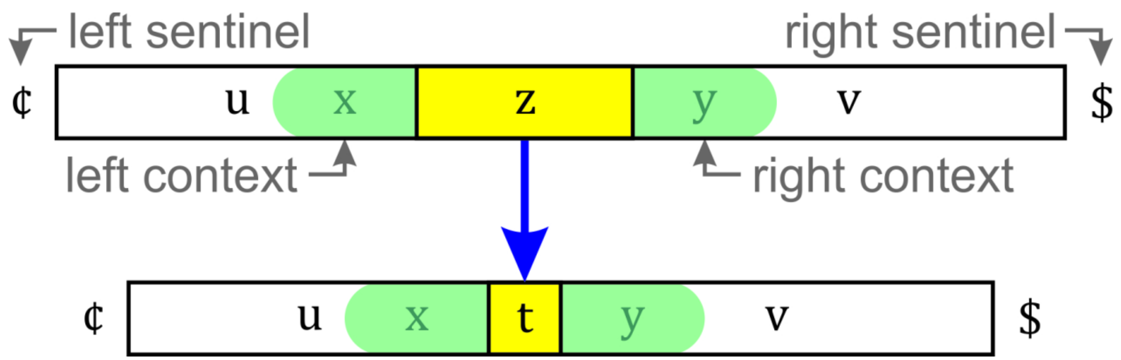
Context Rewriting Systems

- Let k be a *nonnegative integer*.
- ***k* – Context Rewriting System** (*k-CRS*)
- **Is a triple** $M=(\Sigma,\Gamma,I)$:
  - Σ … ***input alphabet***, $¢, \$ \notin \Sigma$,
  - Γ … ***working alphabet***, $\Gamma \supseteq \Sigma$,
  - I … finite set of ***instructions*** $(x, z \to t, y)$:
    - $x \in \Gamma^k \cup \{¢\}.\Gamma^{\leq k-1}$ **(left context)**
    - $y \in \Gamma^k \cup \Gamma^{\leq k-1}.\{\$\}$ **(right context)**
    - $z \in \Gamma^+, z \neq t \in \Gamma^*$.
  - $¢$ and $\$$ … ***sentinels***.
  - The ***width of instruction*** $i=(x, z \to t, y)$ is $|i| = |xzty|$.
  - In case $k=0$ we use $x=y=\lambda$.



Rewriting

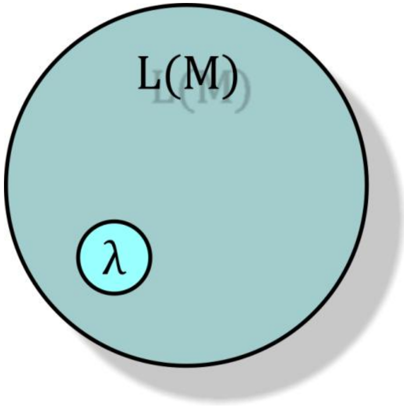
- $uzv \vdash_M utv$ **iff** $\exists\,(x, z \to t, y) \in I$:
- x is a ***suffix*** of $¢.u$ **and** y is a ***prefix*** of $v.\$$.



- $L(M) = \{w \in \Sigma^* \,/\, w \vdash^*_M \lambda\}$.
- $L_C(M) = \{w \in \Gamma^* \,/\, w \vdash^*_M \lambda\}$.

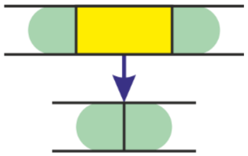
Empty Word

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

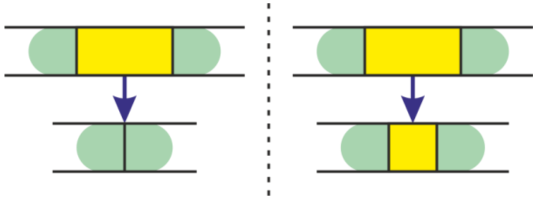


Clearing Restarting Automata

- ***k*– Clearing Restarting Automaton** (*k-cl-RA*)
  - Is a *k-CRS* $M = (\Sigma, \Sigma, I)$ such that:
  - For each $(x, z \to t, y) \in I$: $z \in \Sigma^+$, $t = \lambda$.

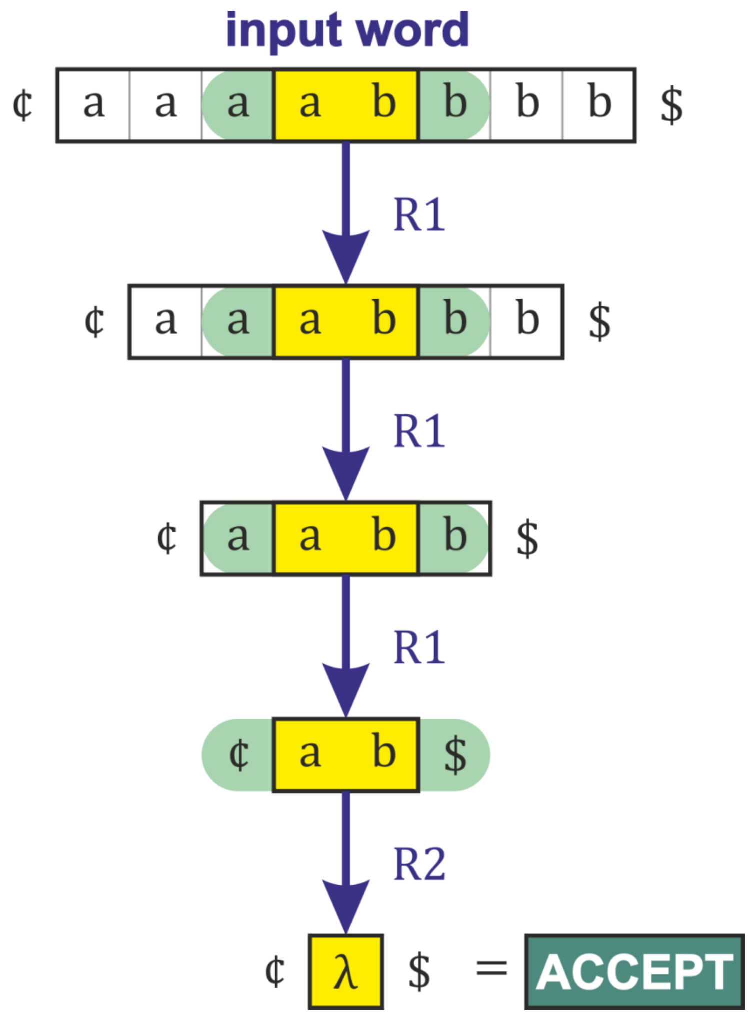


- ***k*– Subword-Clearing Rest. Automaton** (*k-scl-RA*)
  - Is a *k-CRS* $M = (\Sigma, \Sigma, I)$ such that:
  - For each $(x, z \to t, y) \in I$:
  - $z \in \Gamma^+$, t is a ***proper subword*** of z.



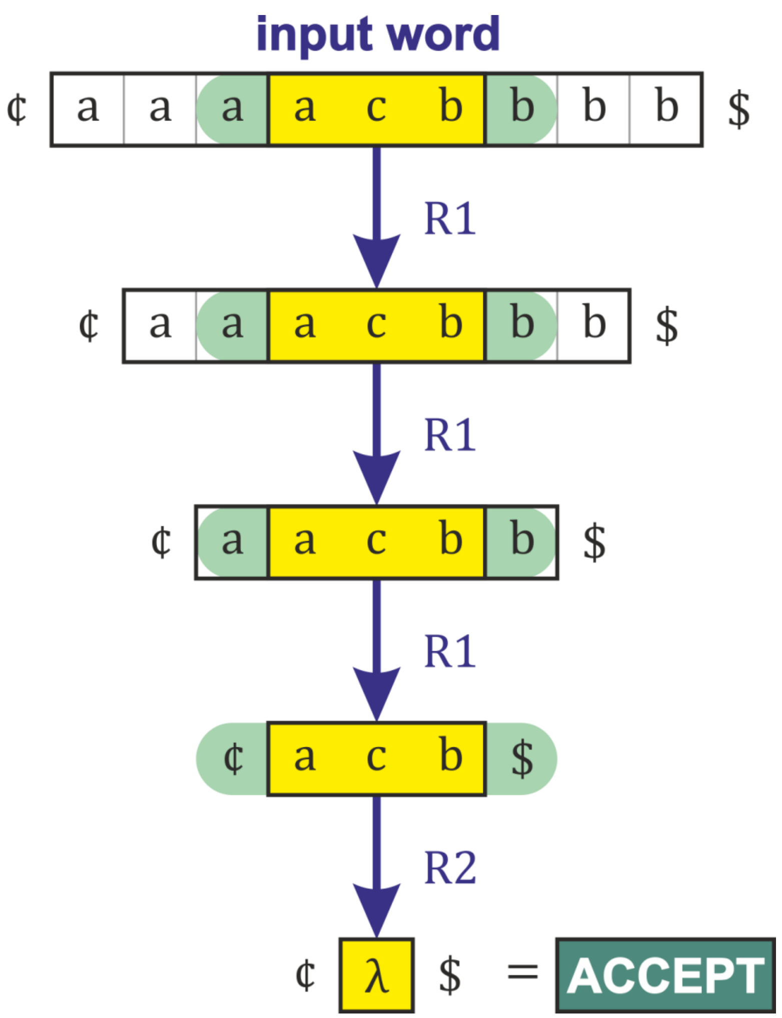
Example 1

- $L_1 = \{a^n b^n \mid n > 0\} \cup \{\lambda\}$:
- 1-cl-RA $M = (\{a, b\}, I)$,
- **Instructions** I are:
  - $R1 = (a, \underline{ab} \to \lambda, b)$,
  - $R2 = (¢, \underline{ab} \to \lambda, \$)$.



Example 2

- $L_2 = \{a^n c b^n \mid n > 0\} \cup \{\lambda\}$:
- *1-scl-RA* $M = (\{a, b, c\}, I)$,
- **Instructions** I are:
  - $R1 = (a, \underline{acb} \rightarrow c, b)$,
  - $R2 = (¢, \underline{acb} \rightarrow \lambda, \$)$.

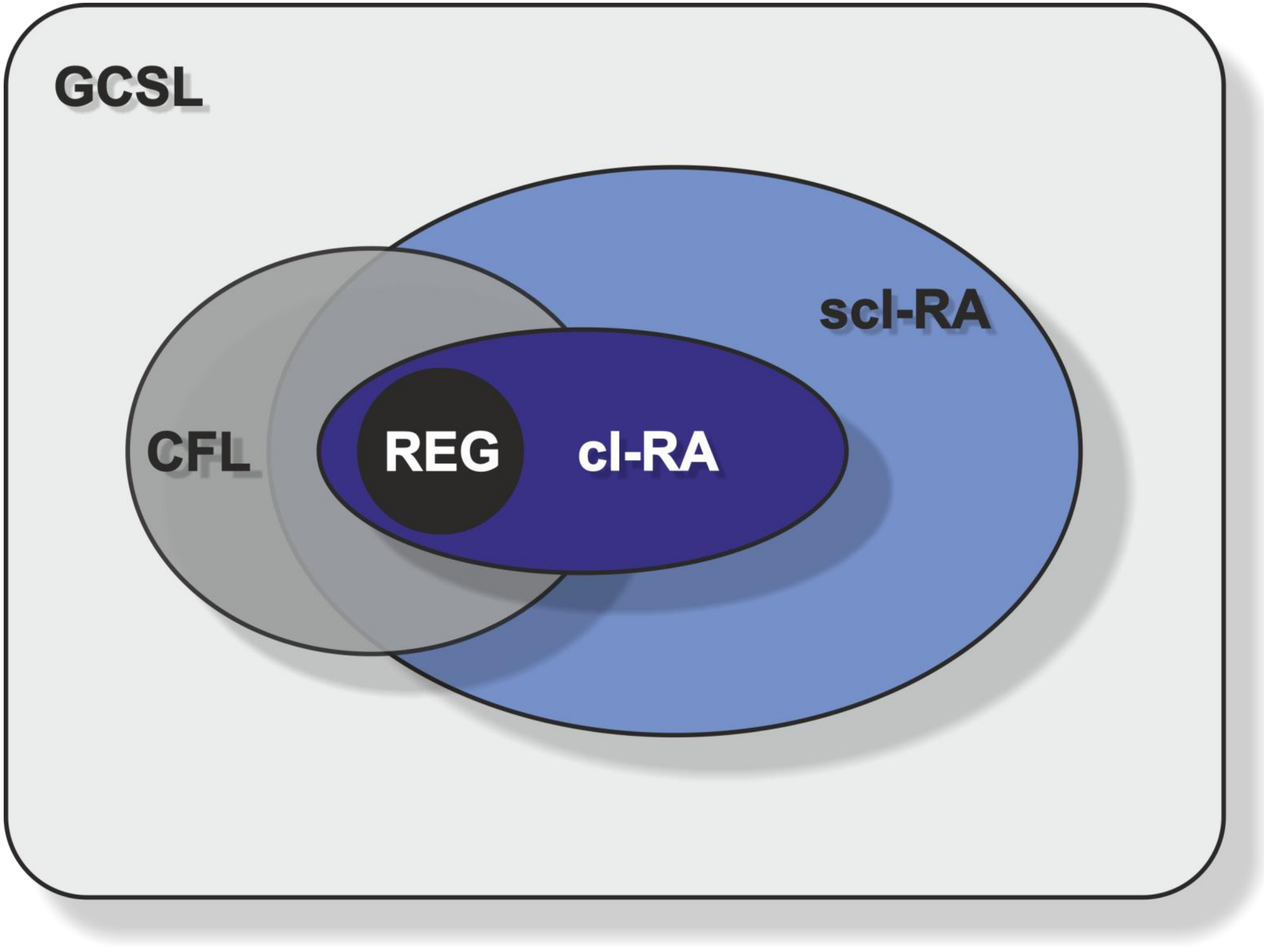


- **Note**:
  - The language L_2 **cannot**
  - be recognized by **any** *cl-RA*.

Clearing Restarting Automata

- **Clearing Restarting Automata**:
  - Accept **all regular** and even **some non-context-free** languages.
  - They do **not** accept **all context-free** languages ($\{a^n c b^n \mid n > 0\}$).
- **Subword-Clearing Restarting Automata**:
  - Are **strictly more powerful** than **Clearing Restarting Automata**.
  - They do **not** accept **all context-free** languages ($\{w w^R \mid w \in \Sigma^*\}$).
- **Upper bound**:
  - **Subword-Clearing Restarting Automata** only accept languages that are **growing context-sensitive** *[Dahlhaus, Warmuth]*.

Hierarchy of Language Classes



Part II: Learning Schema

- **Goal**: ***Identify*** any ***hidden target*** automaton ***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**:
  - ***Automaton*** M such that: $L(M) \subseteq S^+$ and $L(M) \cap S^- = \emptyset$.
  - The term ***automaton*** = ***Clearing*** or ***Subword-Clearing*** Restarting Automaton, or any other ***similar model***.

Learning Schema – Restrictions

- **Without further restrictions**:
  - The task becomes ***trivial*** even for ***Clearing Rest. Aut.***.
  - Just consider: $I = \{ (¢, w, \$) \mid w \in S^+ , w \neq \lambda \}$.
  - Apparently: $L(M) = S^+$, where $M = (\Sigma, \Sigma, I)$.
- **Therefore, we impose**:
  - An ***upper limit*** $l \geq 1$ on the ***width of instructions***,
  - A specific ***length of contexts*** $k \geq 0$.
- **Note**:
  - We can ***effectively enumerate all automata*** satisfying these ***restrictions***, thus the identification in the limit can be easily deduced from the classical result of ***Gold*** …
  - ***Nevertheless***, we propose an ***algorithm***, which, under certain conditions, works in a polynomial time.

Learning Schema – Algorithm

- **Input**:
  - ***Positive samples*** S^+, ***negative samples*** S^-, $S^+ \cap S^- = \emptyset$, $\lambda \in S^+$.
  - ***Upper limit*** $l \geq 1$ on the ***width of instructions***,
  - A specific ***length of contexts*** $k \geq 0$.
- **Output**:
  - ***Automaton*** M such that: $L(M) \subseteq S^+$ and $L(M) \cap S^- = \emptyset$, or ***Fail***.

$\Phi \leftarrow \mathsf{Assumptions}(S^+, l, k)$;
while $\exists w_- \in S^-, w_+ \in S^+, \phi \in \Phi : w_- \vdash^{(\phi)} w_+$ **do**
    $\Phi \leftarrow \Phi \setminus \{\phi\}$;
end
$\Phi \leftarrow \mathsf{Simplify}(\Phi)$;
if $\mathsf{Consistent}(\Phi, S^+, S^-)$ **then**
    return *Automaton with the set of instructions* Φ;
end
Fail;

Learning Schema – Step 1/4

- **Step 1**:

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

  - We obtain some set of ***instruction candidates***.
  - *Note*: We use ***only the positive samples*** to obtain the instructions.
  - Let us ***assume***, for a moment, that this set Φ already ***contains all instructions*** of the ***hidden target automaton***.
  - Later we will show how to define the function *Assumptions* in such a way that the above assumption can be always satisfied.

Learning Schema – Step 2/4

- **Step 2**:

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

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

  end

  - 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***.
  - It is easy to see, that such instructions ***cannot be in our hidden target automaton***.
  - *Note*: Here we use ***also the negative samples***.

Learning Schema – Step 3/4

- **Step 3**:

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

  - 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 Schema***.

- **Possible implementation**:

  Input: The set of instructions Φ.
  Output: The simplified set of instructions Ψ.

  $\Psi \leftarrow \emptyset;$
  foreach $\phi = (x, z \rightarrow t, y) \in \Phi$ *in some fixed order* **do**
      if $z \nvdash^*_\Psi t$ *in the context* (x, y) **then**
          $\Psi \leftarrow \Psi \cup \{(x, z \rightarrow t, y)\};$
      end
  end
  return Ψ;

Learning Schema – Step 4/4

- **Step 4**:

  if Consistent(Φ, S^+, S^-) **then**
  | **return** *Automaton with the set of instructions* Φ;
  end
  Fail;

  - We ***check the consistency*** of the remaining set of instructions with the given input set of positive and negative samples.
  - Concerning the identification in the limit, we can ***omit the consistency check*** – it does not affect the correctness of the ***Learning Schema***. In the limit, we always get a correct solution.

Learning Schema – 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 a polynomial time.
- If the function Assumptions runs in a polynomial time (Step 1) then also the size of the set Φ is polynomial and then also the cycle (Step 2) runs in a polynomial time.
- It is an open problem, whether the ***simplification*** and the ***consistency check*** can be done in a polynomial time. Fortunately, we can omit these steps.

Learning Schema – Assumptions

- We call the ***function Assumptions*** **correct**, if it is possible to obtain instructions of **any hidden target automaton in the limit** by using this function.
- To be more **precise**:
  - For every *k-cl-RA M* (or *k-scl-RA M*) with the maximal width of instructions bounded from above by $l \geq 1$ 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 some automaton N equivalent to M.

Example – $\text{Assumptions}_{weak}$

- $Assumptions_{weak}(S^+, l, k)$:= all instructions $(x, z \to t, y)$:
  - **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)**
  - **Our model is a Subword-Clearing Rest. Aut.:**
    - $z \in \Sigma^+$, t is a ***proper subword*** of z.
  - **The width is bounded by l :**
    - $|xzty| \leq l$.
  - **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***.

Example – Assumptions$_{weak}$

Positive Samples

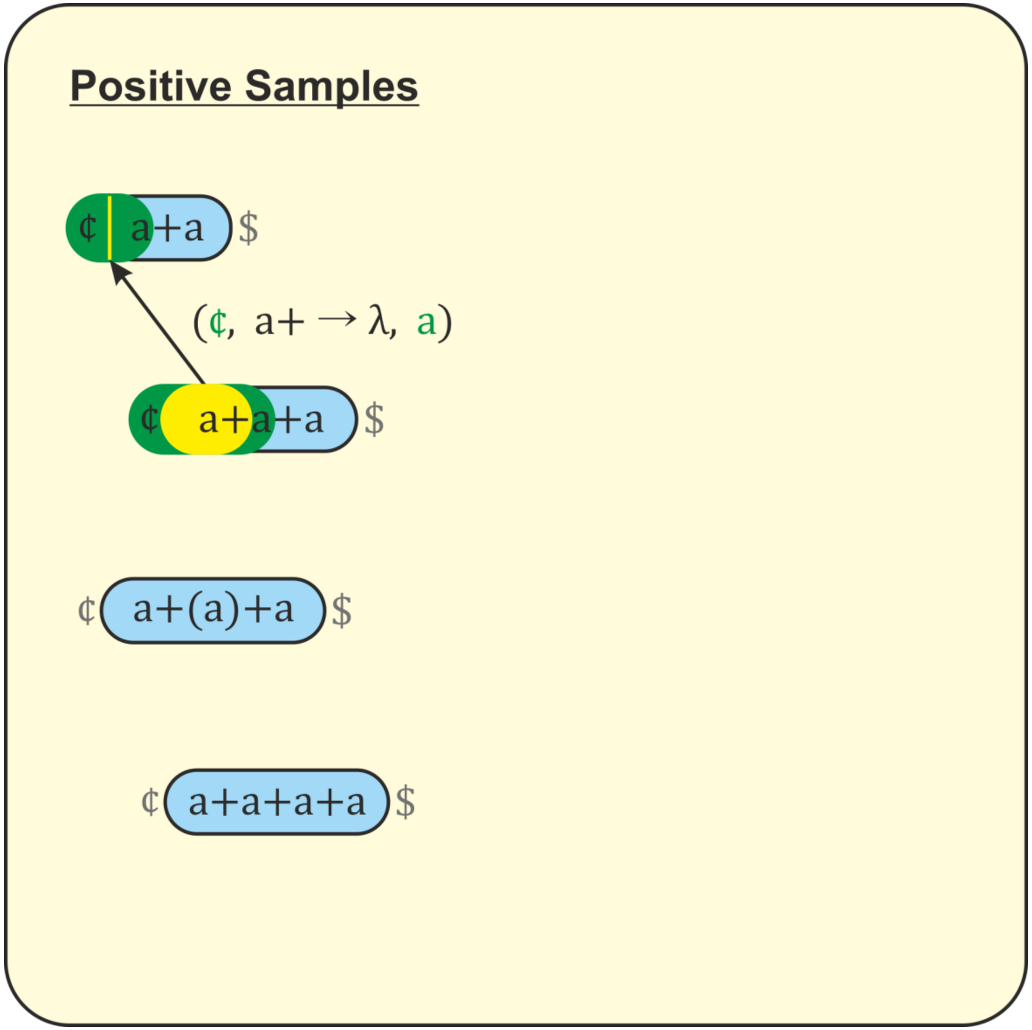
¢ a+a $

¢ a+a+a $

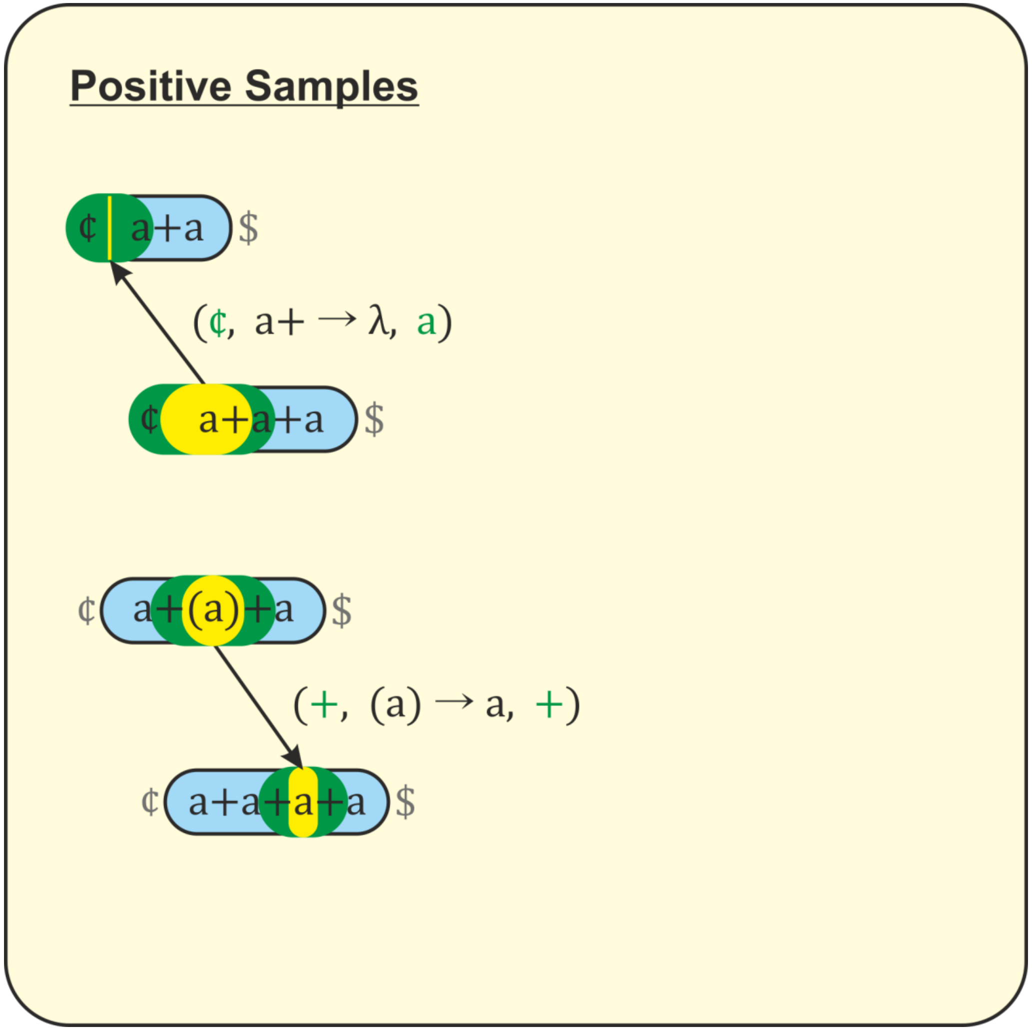
¢ a+(a)+a $

¢ a+a+a+a $

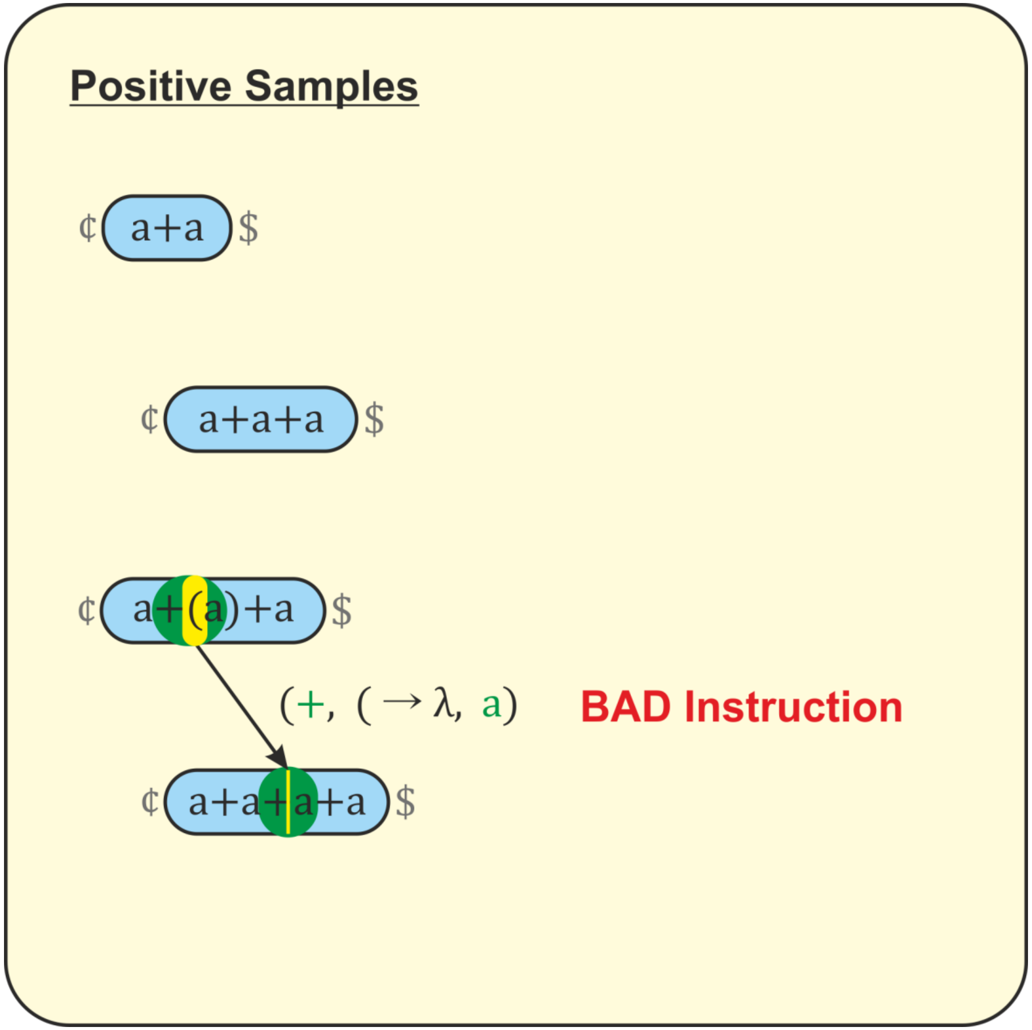
Example – Assumptions$_{weak}$



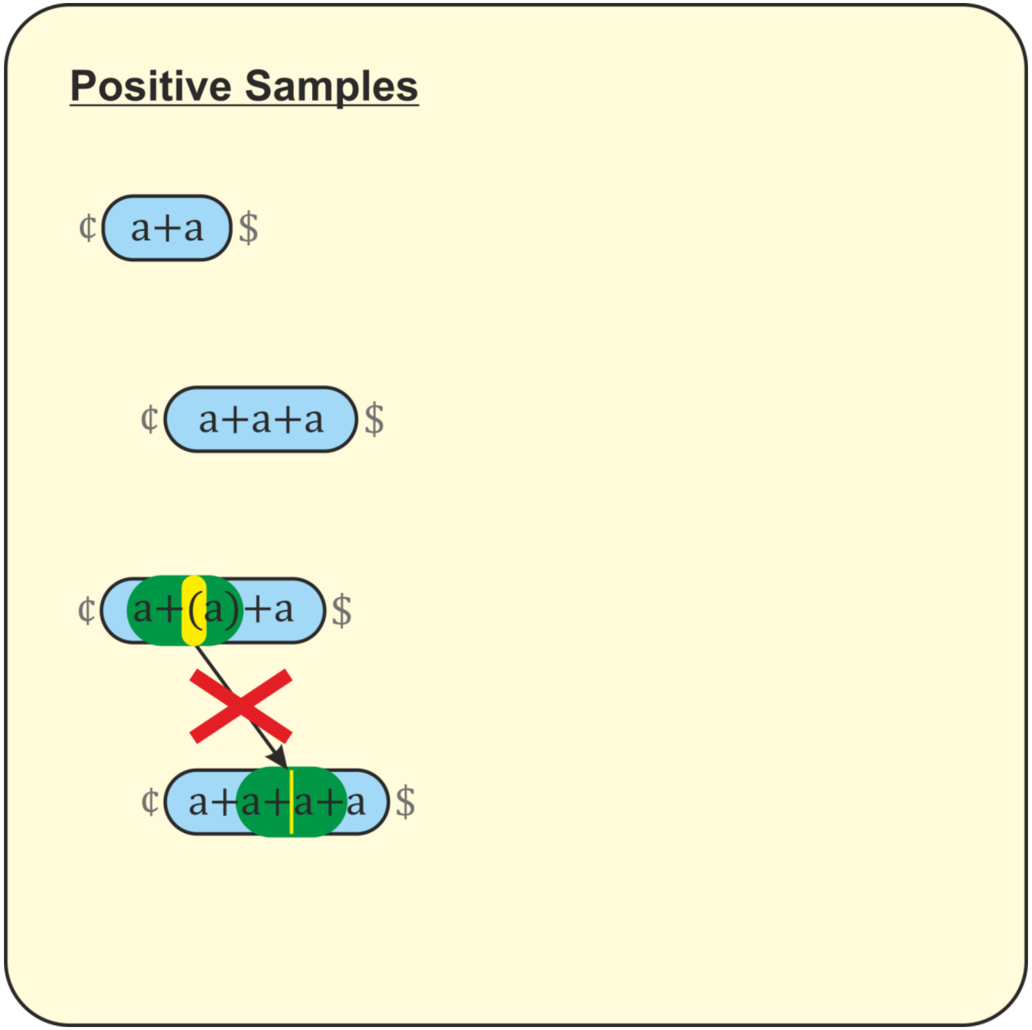
Example – Assumptions$_{weak}$



Example – Assumptions$_{weak}$



Example – Assumptions$_{weak}$



Part III: Active Learning Example

- **Our goal**:
  - Infer a model of *scl-RA* recognizing the language of ***simplified arithmetical expressions*** over the alphabet $\Sigma = \{a, +, (,)\}$.
- **Correct** arithmetical expressions:
  - $a + (a + a)$,
  - $(a + a)$,
  - $((a))$, etc.
- **Incorrect** arithmetical expressions:
  - $a +$,
  - $)\, a$,
  - $(a + a$, etc.
- We fix ***maximal width*** l to 6, ***length of context*** k to 1.

Active Learning Example

- **Initial set** of ***positive*** ***(S_1^+)*** and ***negative*** ***(S_1^-)*** samples.

Table 1: The Initial Set of Positive and Negative Samples.

Positive Samples S_1^+			Negative Samples S_1^-					
a	(a)	$((a+a))$	$+$	$a+$	$++$	$(+$	$)+$	$+a$
$a+a$	$((a))$	$a+(a+a)$	$($	$a($	$+($	$(($	$)($	$(a$
$a+a+a$	$(a+a)$	$(a+a)+a$	$)$	$a)$	$+)$	$()$	$))$	$)a$

Active Learning Example

- *Assumptions*$_{weak}$*(S_1^+, l, k)* gives us ***64*** instructions.
- After ***filtering*** bad instructions and after ***simplification*** we get a consistent automaton ***M_1*** with ***21 instructions***:

Table 2: The Instructions of the Resulting Automaton M_1 After Simplification.

$[(, (, a]$	$[¢, (, (]$	$[+, (, a]$	$[),), \$]$	$[a,),)]$	$[a,), +]$	$[¢, a, \$]$
$[¢, ((, a]$	$[¢, (a \to a, +]$	$[a,)), \$]$	$[), +a, \$]$	$[a, +a, \$]$	$[a, +a,)]$	$[a, +a, +]$
$[+, a) \to a, \$]$	$[(, a+, a]$	$[¢, a+, (]$	$[¢, a+, a]$	$[+, a+, a]$	$[¢, (a), \$]$	$[¢, (a) \to a, \$]$

Active Learning Example

- All expressions recognized by M_1 up to length 5:

Table 3: The Set of Expressions Recognized by M_1.

λ	$\mathbf{a+a}$	$\mathbf{a+(a)}$	$a)))$	$(a+a$	$((((a$
$\mathbf{a}$	$((a)$	$(((a$	$a))+a$	$\mathbf{a+a+a}$	$a))))$
$\mathbf{(a)}$	$(a))$	$\mathbf{((a))}$	$a+a))$	$a+a)$	$a)+(a$
$((a$	$\mathbf{(a)+a}$	$((a+a$	$a+(a$	$(((a)$	$(a+(a$
$a))$	$\mathbf{(a+a)}$	$a+((a$	$a)+a$	$(a)))$	$a)+a)$

- There are both ***correct*** and ***incorrect*** arithmetical expressions. Note that $(a)+a$ was never seen before.
- **Next step**: Add all ***incorrect*** arithmetical expressions to the set of ***negative samples***. (We get: $S_2^+ = S_1^+$ and S_2^-).

Active Learning Example

- We get a consistent automaton M_2 with ***16 instructions***.
- ***Up to length 5***, the automaton M_2 recognizes ***only correct*** arithmetical expressions.
- **However**, it recognizes also ***some incorrect*** arithmetical expressions ***beyond this length***, e.g.:
  - $((a+a)$,
  - $(a+a))$,
  - $a+(a+a$,
  - $a+a)+a$.
- Add also these ***incorrect*** arithmetical expressions to the set of ***negative samples***. (We get: $S_3^+ = S_2^+$ and S_3^-).

Active Learning Example

- Now we get a consistent automaton ***M_3*** with ***12 instructions*** recognizing only ***correct*** expressions.

Table 4: The Instructions of the Resulting Automaton M_3 After Simplification.

$[¢, a, \$]$	$[), +a, \$]$	$[a, +a, \$]$	$[a, +a,)]$
$[a, +a, +]$	$[(, a+, a]$	$[¢, a+, (]$	$[¢, a+, a]$
$[+, a+, a]$	$[(, (a) \to a,)]$	$[¢, (a), \$]$	$[¢, (a) \to a, \$]$

- The automaton is ***not complete*** yet.
- It does not recognize e.g. ***a + (a + (a))***.
- This time we would need to extend the ***positive*** samples.

Part III: Hardness Results

- In general, the ***task of finding*** a ***consistent Clearing*** Rest. Aut. with the given set of positive and negative samples is ***NP-hard***, provided that we impose an ***upper bound on the width of instructions***.
- This resembles a ***famous result of Gold*** who showed that the question of whether there is a ***finite automaton with at most n states*** consistent with a given list of input/output pairs is ***NP-complete***.
- **Indeed**, for every ***n-state finite automaton***, there is an equivalent ***Clearing Restarting Automaton*** that has the ***width of instructions bounded from above by O(n)***.

Hardness Results

- Let $l \geq 2$ be a ***fixed integer***. Consider the following **task**:
- **Input**:
  - Set of ***positive samples*** S^+,
  - Set of ***negative samples*** S^-,
  - We assume that $S^+ \cap S^- = \emptyset$, and $\lambda \in S^+$.
- **Output**:
  - *0-cl-RA* M such that:

1. The ***width of instructions*** of M is at most l.
2. $L(M) \subseteq S^+$ and $L(M) \cap S^- = \emptyset$.

- **Theorem**:
  - This task is **NP**-complete.

Hardness Results – Generalization

- Let $k \geq 1$ and $l \geq 4k+4$ be ***fixed integers***. Consider:
- **Input**:
  - Set of ***positive samples*** S^+,
  - Set of ***negative samples*** S^-,
  - We assume that $S^+ \cap S^- = \emptyset$, and $\lambda \in S^+$.
- **Output**:
  - *k-cl-RA* M such that:

1. The ***width of instructions*** of M is at most l.
2. $L(M) \subseteq S^+$ and $L(M) \cap S^- = \emptyset$.

- **Theorem**:
  - This task is **NP**-complete for $k = 1$ and **NP**-hard for $k > 1$.

Part V: Concluding Remarks

- We have shown that it is possible to ***infer*** any ***hidden target Clearing*** (***Subword-Clearing***) Rest. Aut. ***in the limit*** from ***positive*** and ***negative*** samples.
- However, the ***task of finding*** a ***consistent Clearing*** Rest. Aut. with the given set of ***positive*** and ***negative*** samples is ***NP-hard***, provided that we impose an ***upper bound on the width of instructions***.
- If we ***do not impose any upper bound*** on the maximal ***width of instructions***, then the task is trivially decidable in a polynomial time for any $k \geq 0$.

Open Problems

- Do similar ***hardness results*** hold also for ***other*** (more powerful) ***models*** like ***Subword-Clearing*** Rest. Aut.?
- What is the ***time complexity*** of the ***membership*** and ***equivalence*** queries for these models?

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**.
  - Technical Report 1/2012, Charles University, Faculty of Mathematics and Physics, Prague, 2012. URL http://popelka.ms.mff.cuni.cz/cerno/files/cerno_clra_and_gi.pdf.
- 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.
- Y. Zalcstein. **Locally testable languages**.
  - J. Comput. Syst. Sci, 6(2):151-167, 1972.

Thank You!

- The ***technical report*** is available on:
  http://popelka.ms.mff.cuni.cz/cerno/files/cerno_clra_and_gi.pdf
- This ***presentation*** is available on:
  http://popelka.ms.mff.cuni.cz/cerno/files/cerno_clra_and_gi_presentation.pdf
- An ***implementation*** of the algorithms can be found on:
  http://code.google.com/p/clearing-restarting-automata/