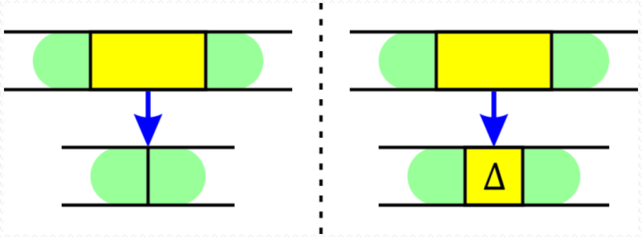
Δ-Clearing Restarting Automata and CFL

Peter Černo and František Mráz

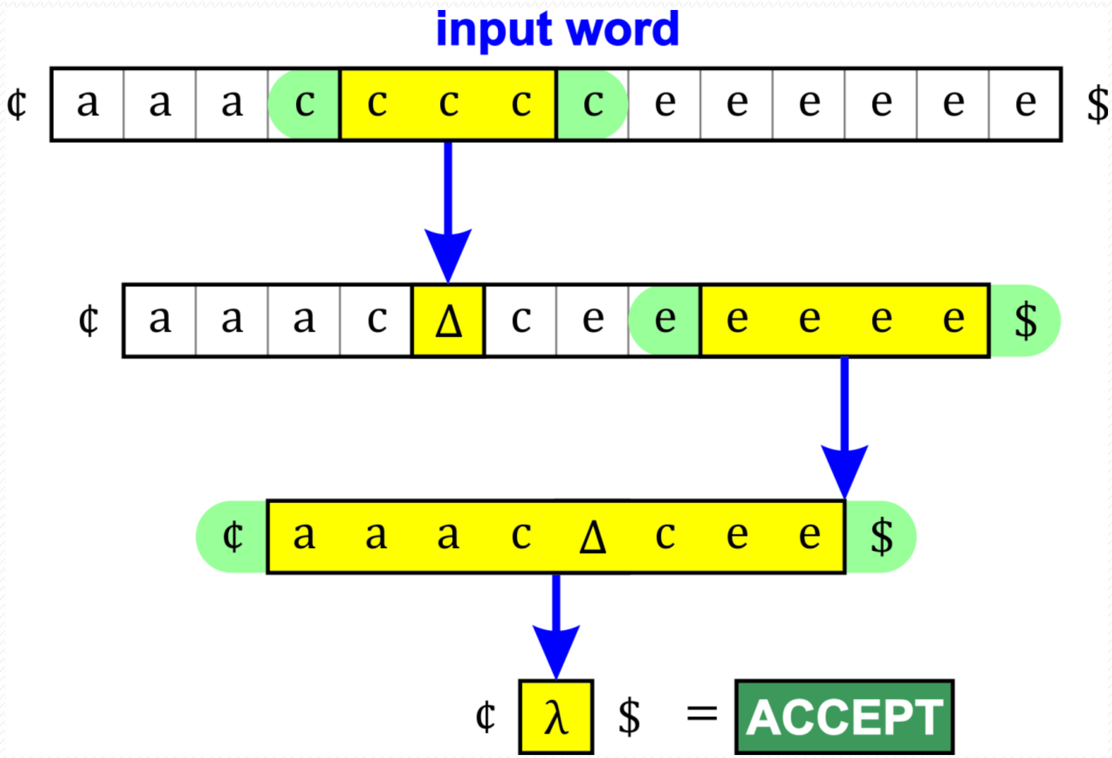
Introduction

- **Δ-Clearing Restarting Automata**:
- **Restricted model** of **Restarting Automata**.
- In **one step** (based on a **limited context**):
  - **Delete a substring**,
  - **Replace a substring** by Δ.



- **The main result**:
- **Δ-clearing restarting automata** recognize **all context-free languages**.

Example



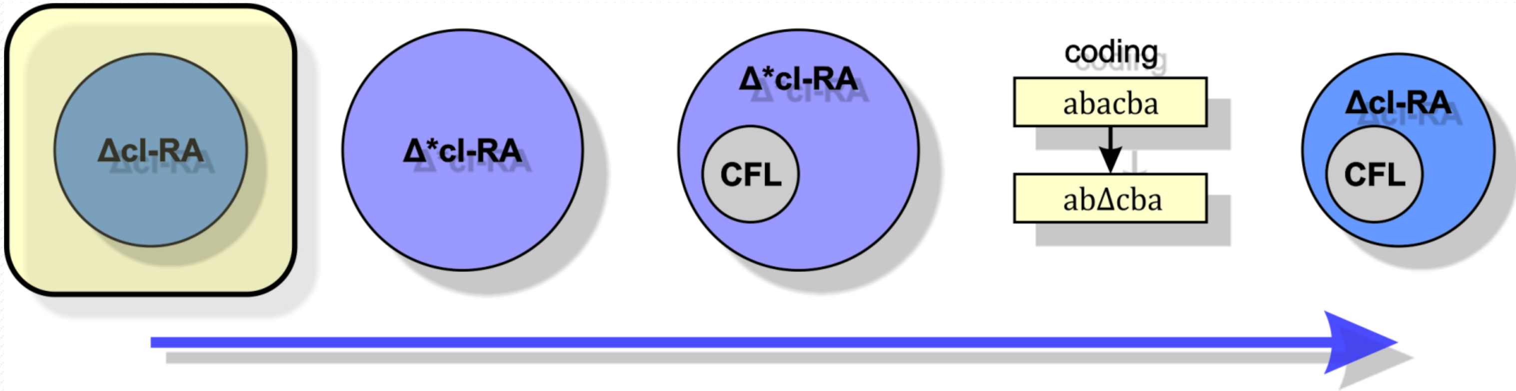
input word
¢ a a a c c c c c e e e e e e $
¢ a a a c Δ c e e e e e e $
¢ a a a c Δ c e e $
¢ λ $ = ACCEPT

Restarting Automata

- **Restarting Automata**:
  - Tool for modeling some techniques for natural language processing.
- **Analysis by Reduction**:
  - Method for checking [non-]correctness of a sentence.
  - Iterative application of simplifications.
  - Until the input cannot be simplified anymore.

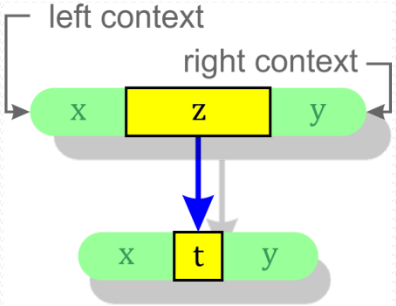
Organization

1. ***Δ*-clearing restarting automata**.
2. *Δ**-clearing restarting automata.
3. *Δ**-clearing restarting automata recognize *CFL*.
4. Special coding.
5. Reduction: *Δ**- to *Δ*-clearing restarting automata.



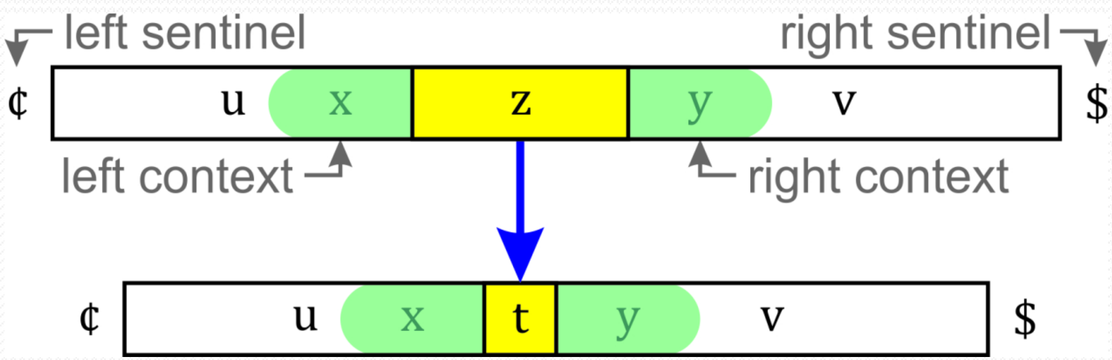
Δ-Clearing Restarting Automata

- Let k be a *positive integer*.
- ***k-Δ-clearing restarting automaton*** (*k-Δcl-RA*)
- **Is a couple** $M = (\Sigma, I)$:
  - Σ … ***input alphabet***, $¢, \$, \Delta \notin \Sigma$,
  - Γ … ***working alphabet***, $\Gamma = \Sigma \cup \{\Delta\}$
  - I … finite set of ***instructions*** $(x, z \to t, y)$:
    - $x \in \{¢, \lambda\}.\Gamma^*,\ |x| \leq k$ (left context)
    - $y \in \Gamma^*.\{\lambda, \$\},\ |y| \leq k$ (right context)
    - $z \in \Gamma^+,\ t \in \{\lambda, \Delta\}$.
  - $¢$ and $\$$ … ***sentinels***.



Rewriting

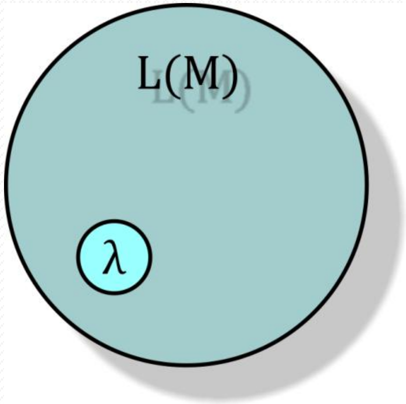
- $u\underline{z}v \vdash_M utv$ **iff** $\exists\, \varphi = (x, z \rightarrow t, y) \in I$:
- x is a ***suffix*** of $¢.u$ **and** y is a ***prefix*** of $v.\$$.



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

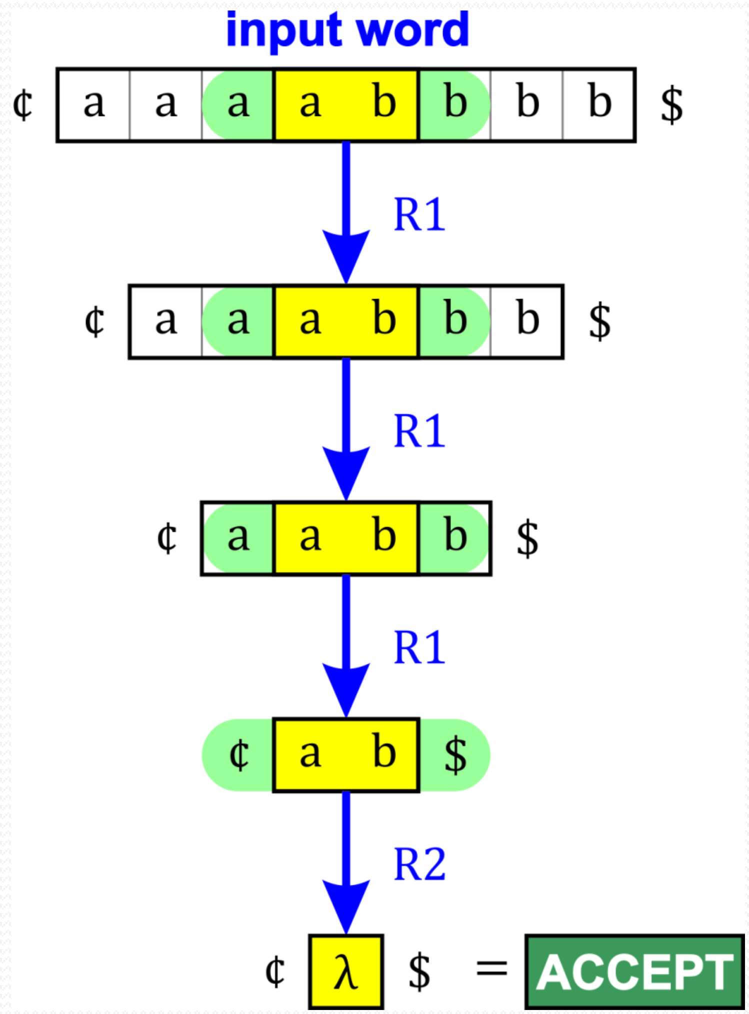
Empty Word

- **Note**: For every *Δcl-RA M:* $\lambda \vdash^{*}_{M} \lambda$ hence $\lambda \in L(M)$.
- Whenever we say that *Δcl-RA M* ***recognizes a language*** *L*, we always mean that $L(M) = L \cup \{\lambda\}$.



Example 1

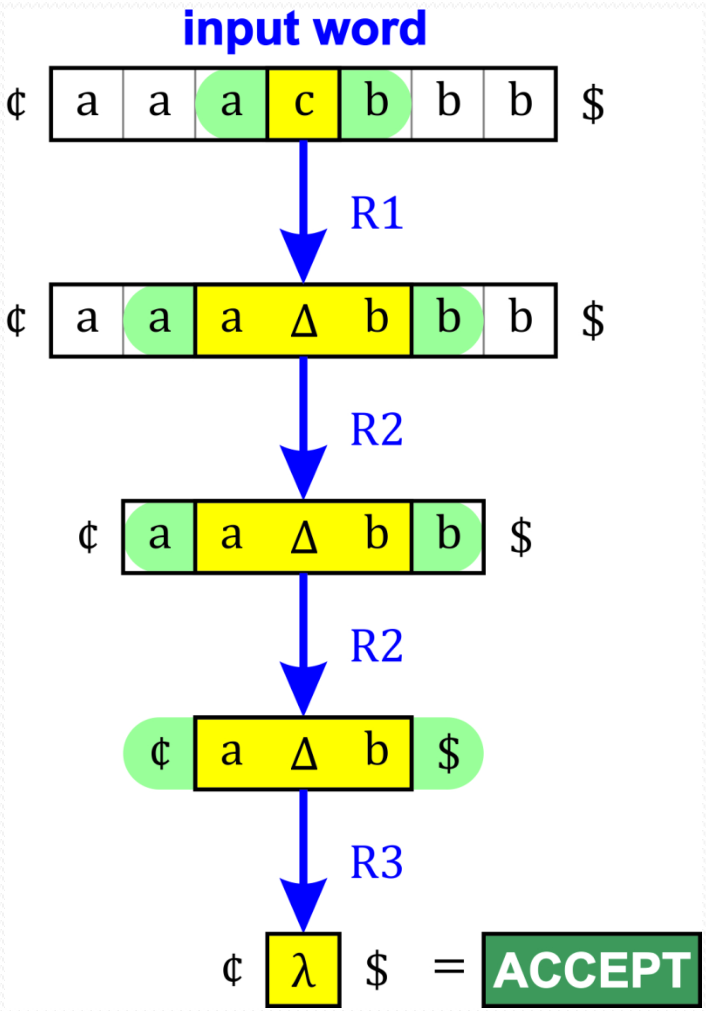
- $L_1 = \{a^n b^n \mid n > 0\} \cup \{\lambda\}$:
- *1-Δcl-RA* $M = (\{a, b\}, I)$,
- **Instructions** I are:
  - $R1 = (a, \underline{ab} \to \lambda, b)$,
  - $R2 = (¢, \underline{ab} \to \lambda, \$)$.



- **Note**:
  - We **did not use** *Δ*.

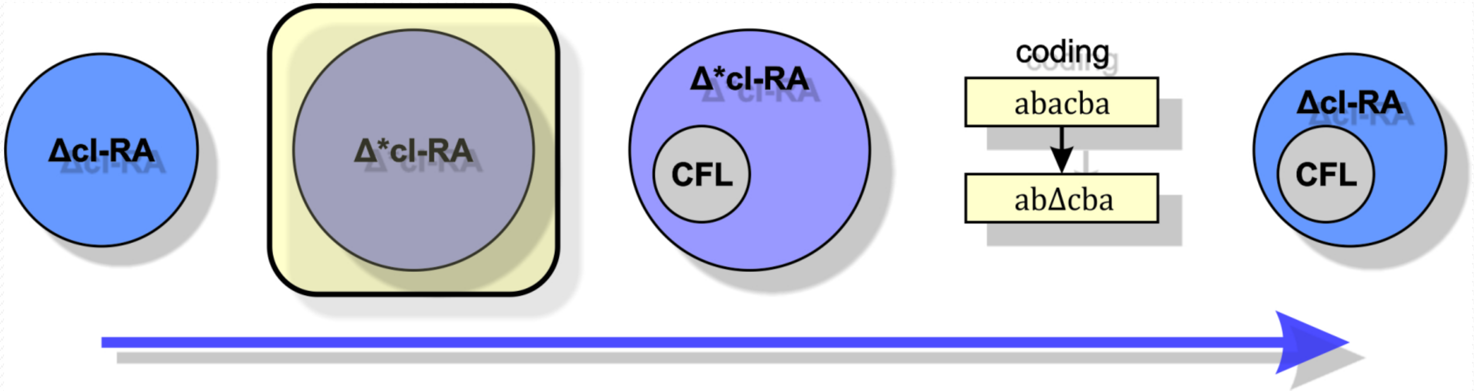
Example 2

- $L_2 = \{a^n c b^n \mid n > 0\} \cup \{\lambda\}$:
- 1-Δcl-RA $M = (\{a, b, c\}, I)$,
- **Instructions *I*** are:
  - $R1 = (a, \underline{c} \to \Delta, b)$,
  - $R2 = (a, \underline{a\Delta b} \to \Delta, b)$,
  - $R3 = (¢, \underline{a\Delta b} \to \lambda, \$)$.
- **<u>Note</u>**:
  - We **must use Δ**.



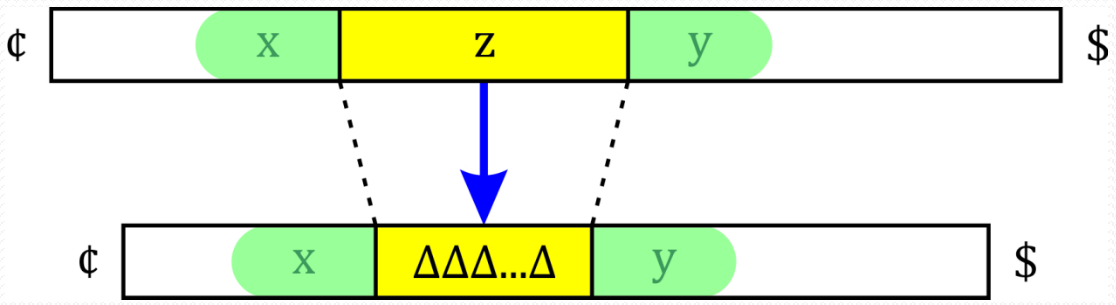
Organization

1. *Δ*-clearing restarting automata.
2. ***Δ*-clearing restarting automata***.
3. *Δ**-clearing restarting automata recognize *CFL*.
4. Special coding.
5. Reduction: *Δ**- to *Δ*-clearing restarting automata.



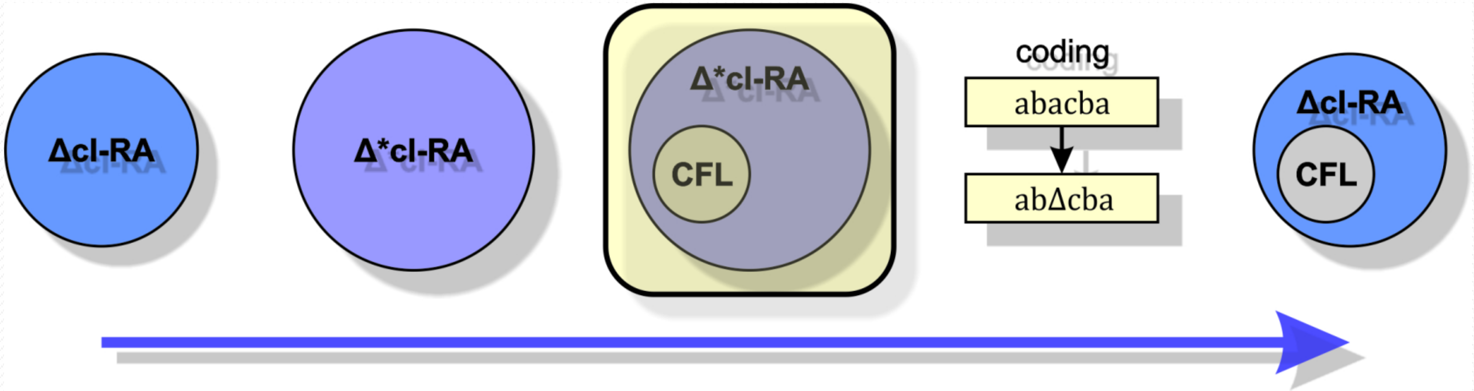
Δ^*-Clearing Restarting Automata

- <u>Δ^*-clearing restarting automata</u>
- **Similar** to Δ-clearing restarting automata.
- **We allow** instructions $(x, z \to \Delta^k, y)$, where $k \leq |z|$.



Organization

1. *Δ*-clearing restarting automata.
2. *Δ**-clearing restarting automata.
3. ***Δ*-clearing restarting automata recognize** ***CFL*.**
4. Special coding.
5. Reduction: *Δ**- to *Δ*-clearing restarting automata.



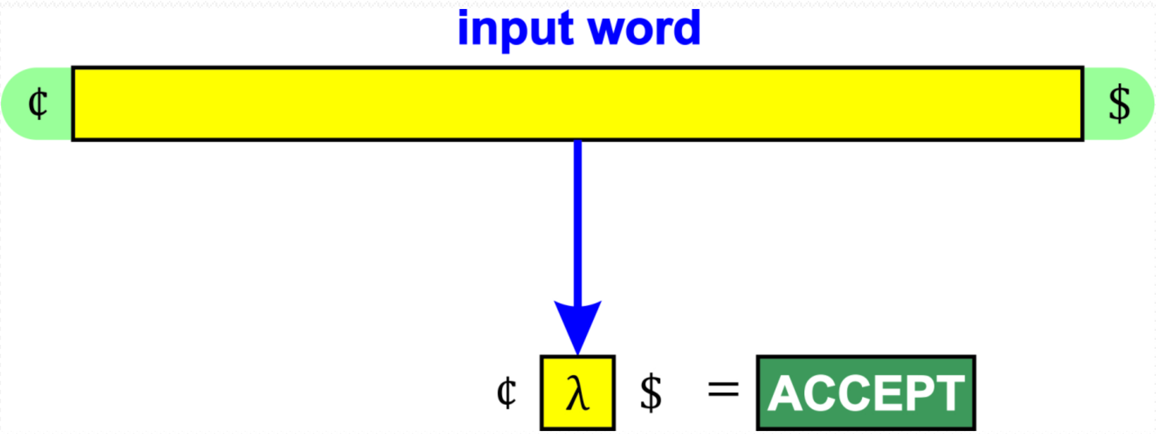
Δ*cl-RA* and *CFL*

- **Theorem**: For each **context-free language** L there exists a *1-Δ*cl-RA* M **recognizing** L.
- **Idea**.
  - M works in a **bottom-up manner**.

1. If the **input is small**, M may "**clear**" the whole input.
2. If the **input is long**, M may "**replace**" some subword by the "**code**" of nonterminal.

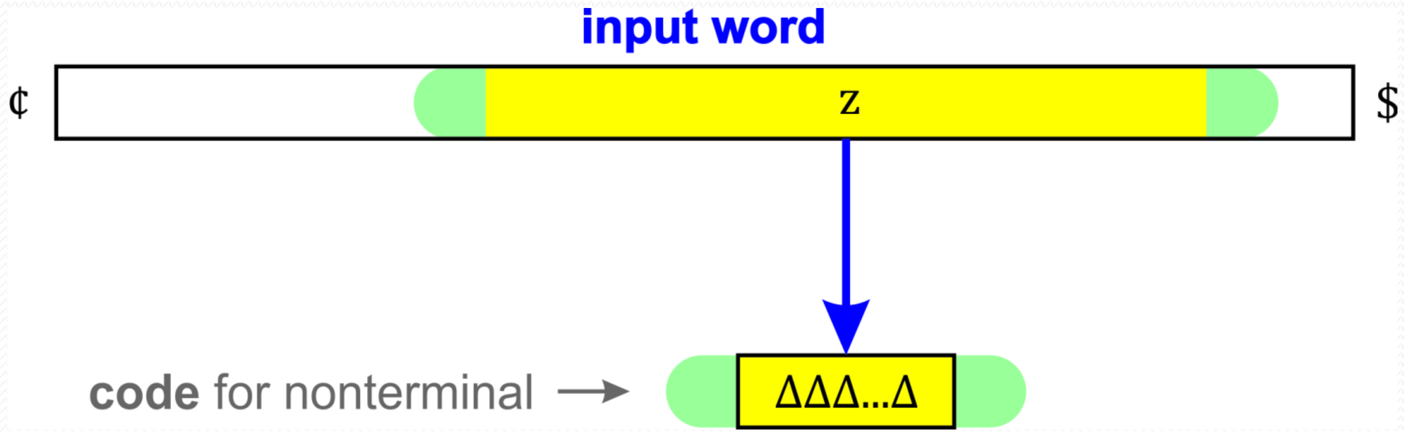
*Δ*cl-RA* and *CFL*

- If the **input is small**:



*Δ*cl-RA* and *CFL*

- If the **input is long**:



$\Delta^*cl\text{-}RA$ and CFL

- ***Proof***.
  - L ... context-free language over Σ.
  - $G = (V_N, V_T, S, P)$ ... context-free grammar :
    - G is in: **Chomsky normal form**,
    - G generates: $L(G) = L - \{\lambda\}$,
    - Nonterminals: $V_N = \{N_1, N_2, ..., N_m\}$,
    - Terminals: $V_T = \Sigma$, $\Gamma = \Sigma \cup \{\Delta\}$,
    - Start: $S = N_1$,

$¢, \$, \Delta \notin V_N \cup V_T$.

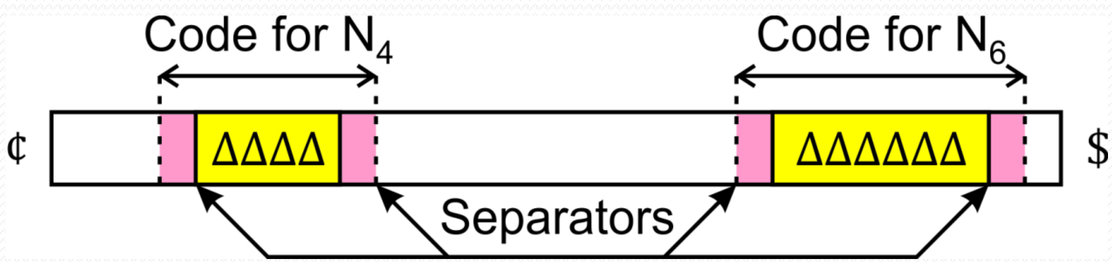
$\Delta^*cl\text{-}RA$ and CFL

- ***Proof*** *(Continued)*.
  - **Auxiliary** $G' = (V_N, V'_T, S, P')$ obtained from G:
  1. By **adding symbol** Δ **to** V_T,
     - $V'_T = V_T \cup \{\Delta\} = \Gamma$,
  2. By **adding productions** $N_i \to a\Delta^i b$ **to** P,
     - $P' = P \cup \{ N_i \to a\Delta^i b \mid i = 1, \ldots, m;\ a, b \in V_T\}$.

  - **Our goal**: $1\text{-}\Delta^*cl\text{-}RA\ M$: $L_C(M) = L(G') \cup \{\lambda\}$.
  - **Then**: $L(M) = L_C(M) \cap \Sigma^* = L(G) \cup \{\lambda\}$.

*Δ*cl-RA* and *CFL*

- ***Proof*** *(Continued)*.
  - $a\Delta^i b$ **code** for N_i ($\forall a, b \in V_T$).
  - $a, b \in V_T$ **separators** **(between codes)**.

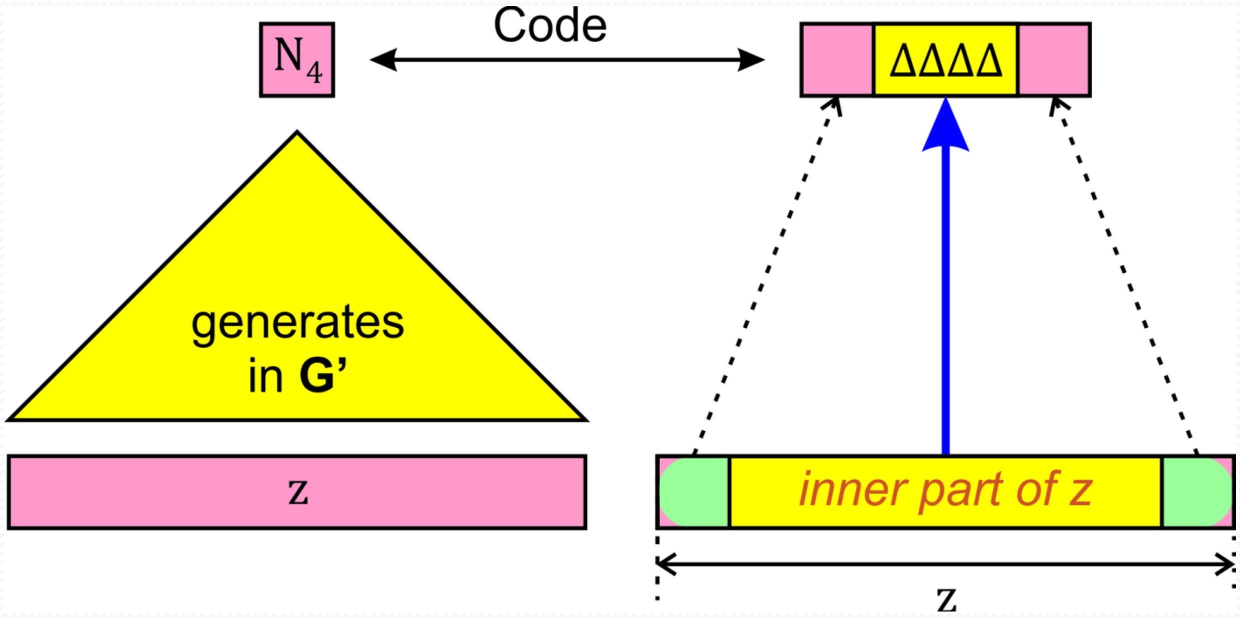


Δ^**cl-RA* and *CFL*

- ***Proof*** *(Continued)*.
- **Idea**:
  - **If** ***z*** **can be derived from** N_i (in *G'*),
  - **Then** ***M*** can replace ***z*** by a "**code**" **for** N_i.
  - ***M*** replaces only the **inner part of *z*** by Δ^i .
  - ***M*** leaves **first** and **last letter** of ***z*** as **separator**.

*Δ*cl-RA* and *CFL*

- **Idea**:



$\Delta^*cl\text{-}RA$ and CFL

- ***Proof*** *(Continued)*.
- **Two problems**:
  1. **|Inner part|** $\geq$ $|\Delta^i|$.
  2. **Finite many instructions**.
- **Proposition**:
  - **For any** $w \in L(G')$:
  - **If** $|w| > c = |V_N| + 2$, **then** $w = x\,z\,y$:
  1. $c < |z| \leq 2c$,
  2. $S \Rightarrow^* x\,N_i\,y \Rightarrow^* x\,z\,y$ for some N_i.

*Δ*cl-RA* and *CFL*

- ***Proof*** *(Continued)*.
- **Construction**:
  - I_1 … set of **all instructions**:

$$(¢, w \to \lambda, \$)$$

  - **Where** $w \in L(G')$ **and** $|w| \leq c$.
  - This resolves the "**small**" **inputs**.

$\Delta^*cl\text{-}RA$ and CFL

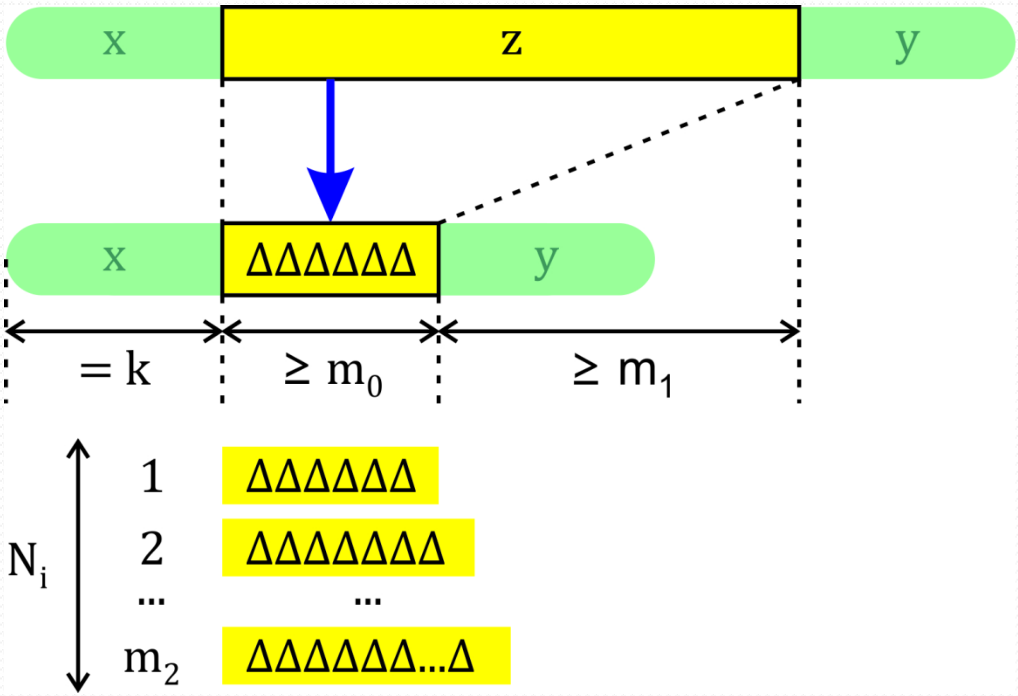
- ***Proof*** *(Continued)*.
- **Construction**:
  - For **every** $N_i \Rightarrow^* z_1 \dots z_s$, **where** $c < s \leq 2c$:

$$(z_1, z_2 \dots z_{s-1} \to \Delta^i, z_s)$$

  - I_2 ... set of **all such instructions**.
  - I_1, I_2 ... **finite** sets of instructions.
  - $M = (\Sigma, I_1 \cup I_2)$ ... **required automaton**. **Q.E.D. ■**

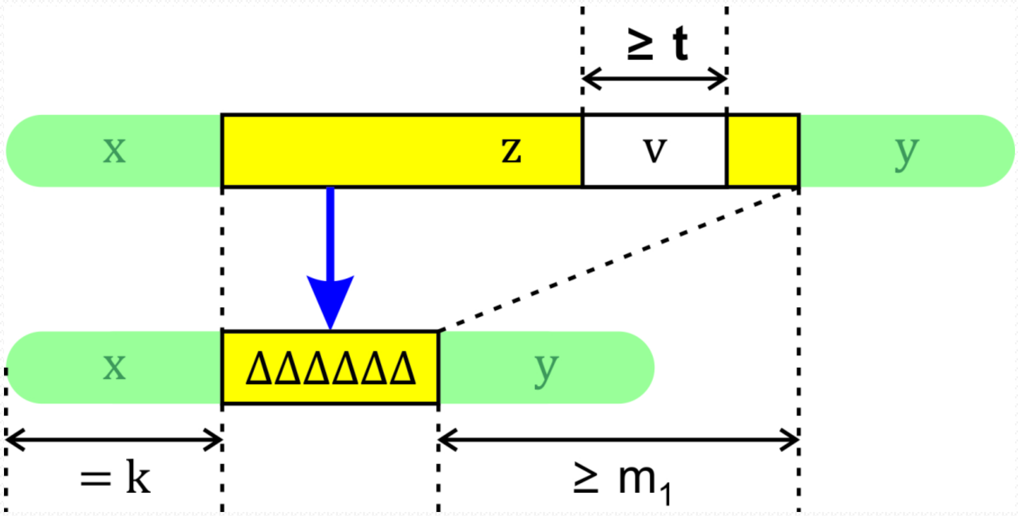
Generalization

- **We can choose**:



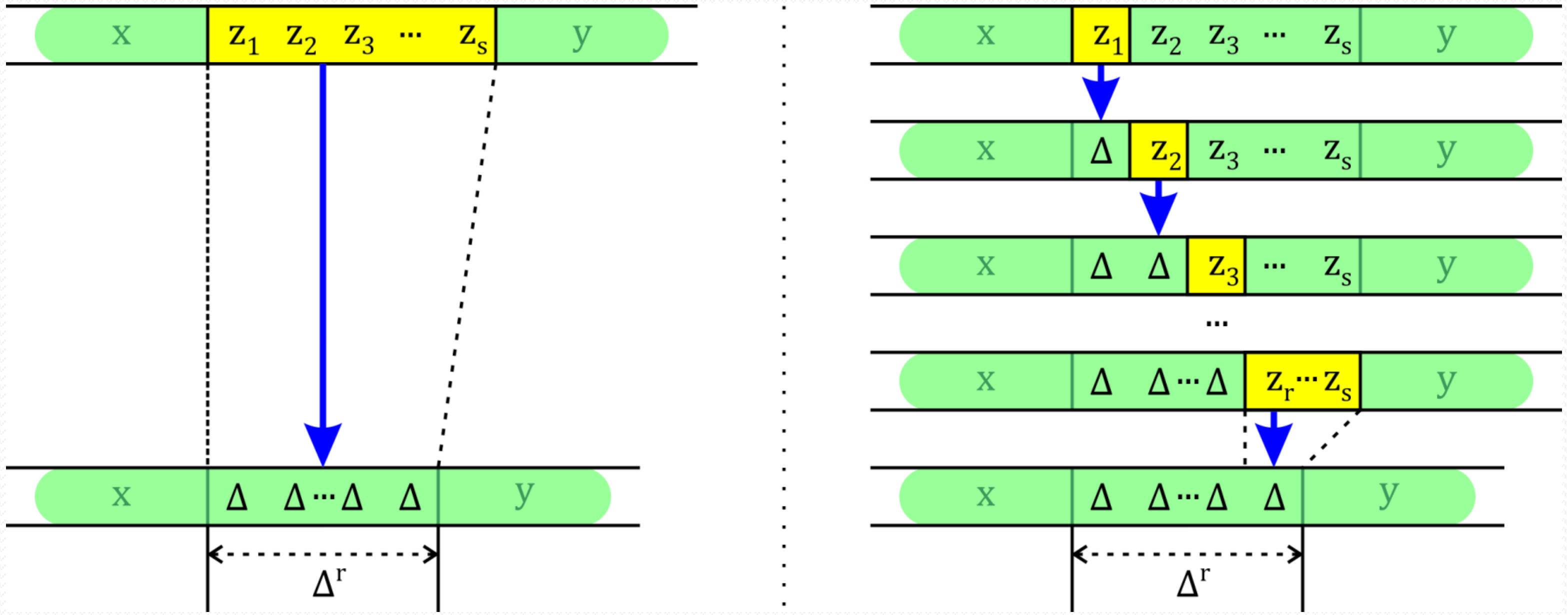
Generalization

- **Observation**:
  - For every $t \geq 1$ … $\exists\ m_1 , k$:
  - z **contains** $v \in \Sigma^{\geq t}$ … **empty space**.



Trivial Reduction

- **Why empty space**?
- **Trivial simulation**:



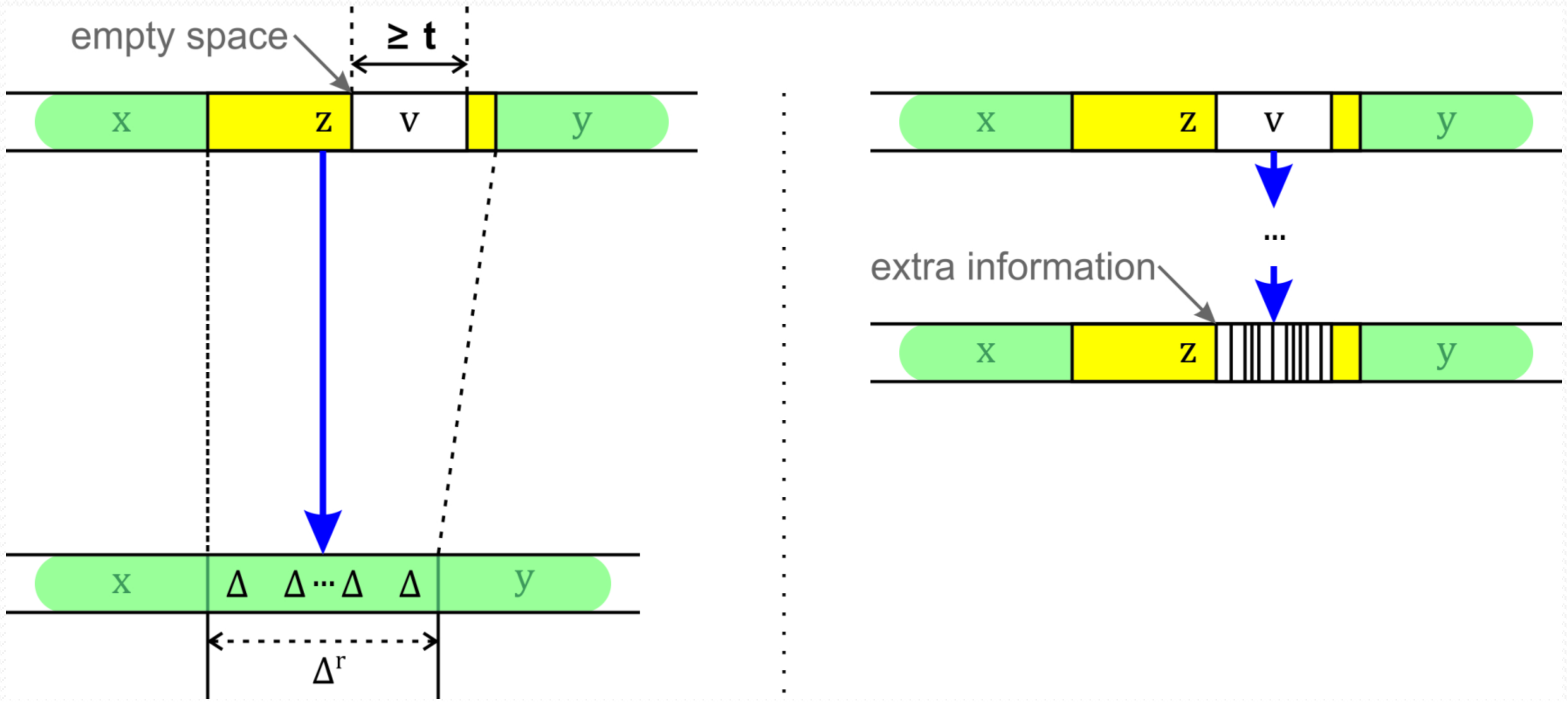
Trivial Reduction

- **Why empty space**?
- **Partial Δ-instructions**:
  - $\varphi_1 = (x, z_1 \rightarrow \Delta, z_2 z_3 \dots z_s y)$,
  - $\varphi_2 = (x\Delta, z_2 \rightarrow \Delta, z_3 \dots z_s y)$,
  - …
  - $\varphi_r = (x\Delta^{r-1}, z_r \dots z_s \rightarrow \Delta, y)$.

- **Problem**:
  - The **equivalence** is **not guaranteed**.

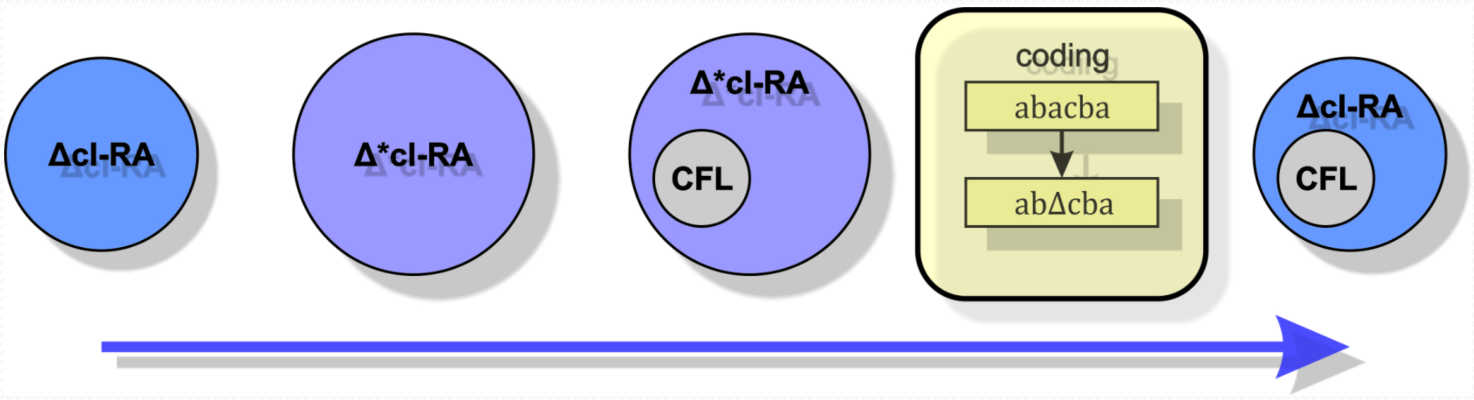
Avoiding Conflicts

- How to **avoid conflicts**?
- We can **encode** some **extra information** into ***z***.



Organization

1. *Δ*-clearing restarting automata.
2. *Δ**-clearing restarting automata.
3. *Δ**-clearing restarting automata recognize *CFL*.
4. **Special coding**.
5. Reduction: *Δ**- to *Δ*-clearing restarting automata.

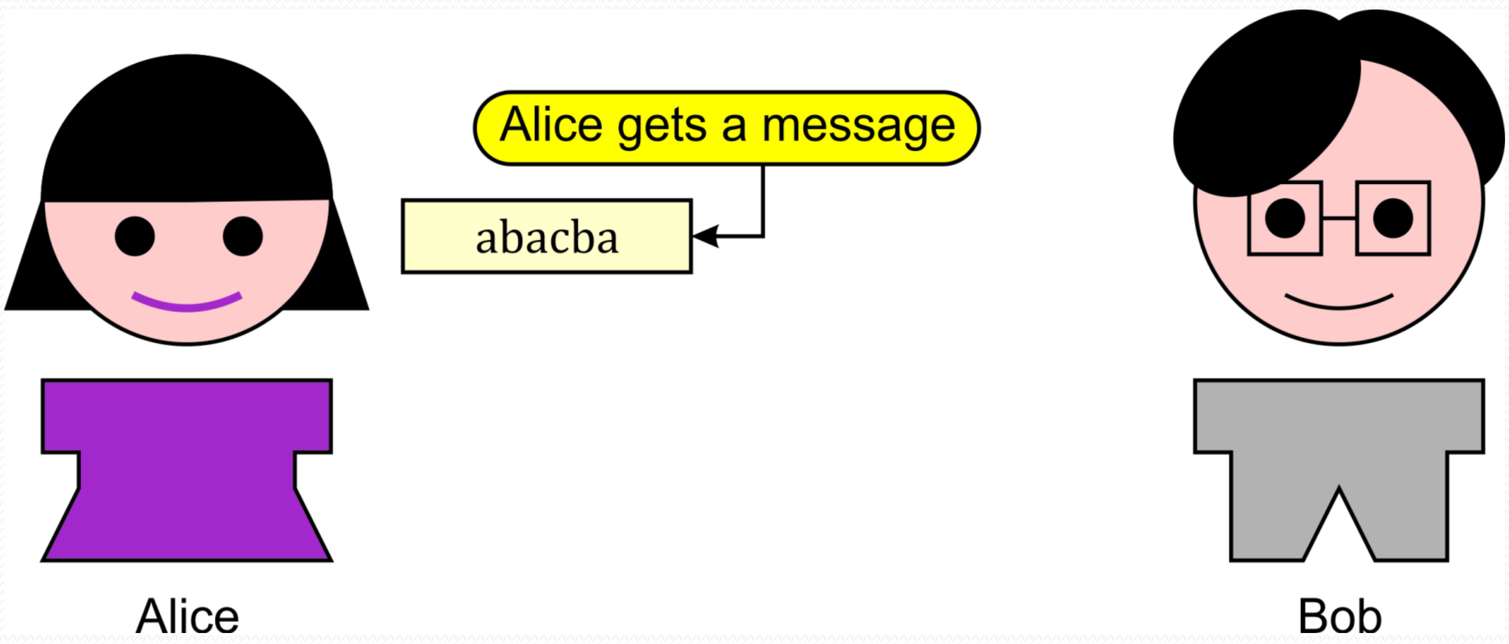


Coding

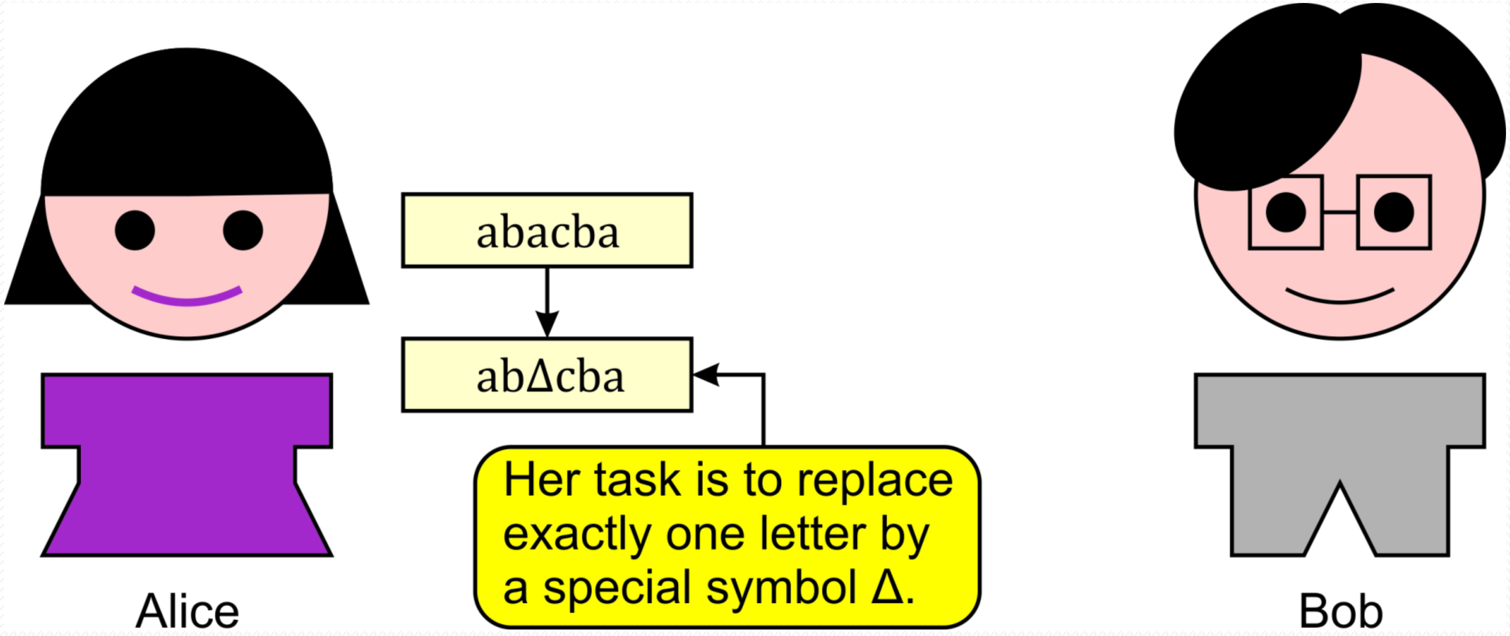
- **<u>We require</u>**:
  - Coding by **means** of Δ-clearing restarting automata.
  - Ability to **recover** the original word at any time.

Alice and Bob

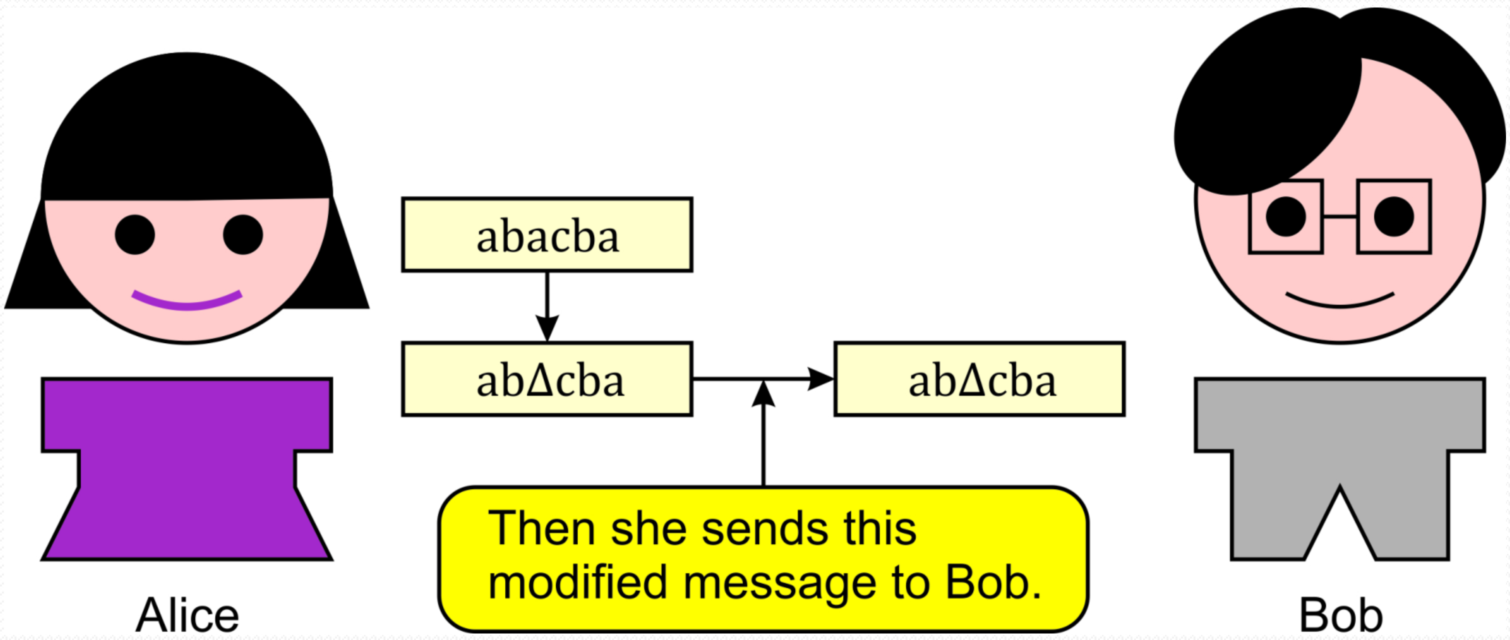
- Consider the following **game**:



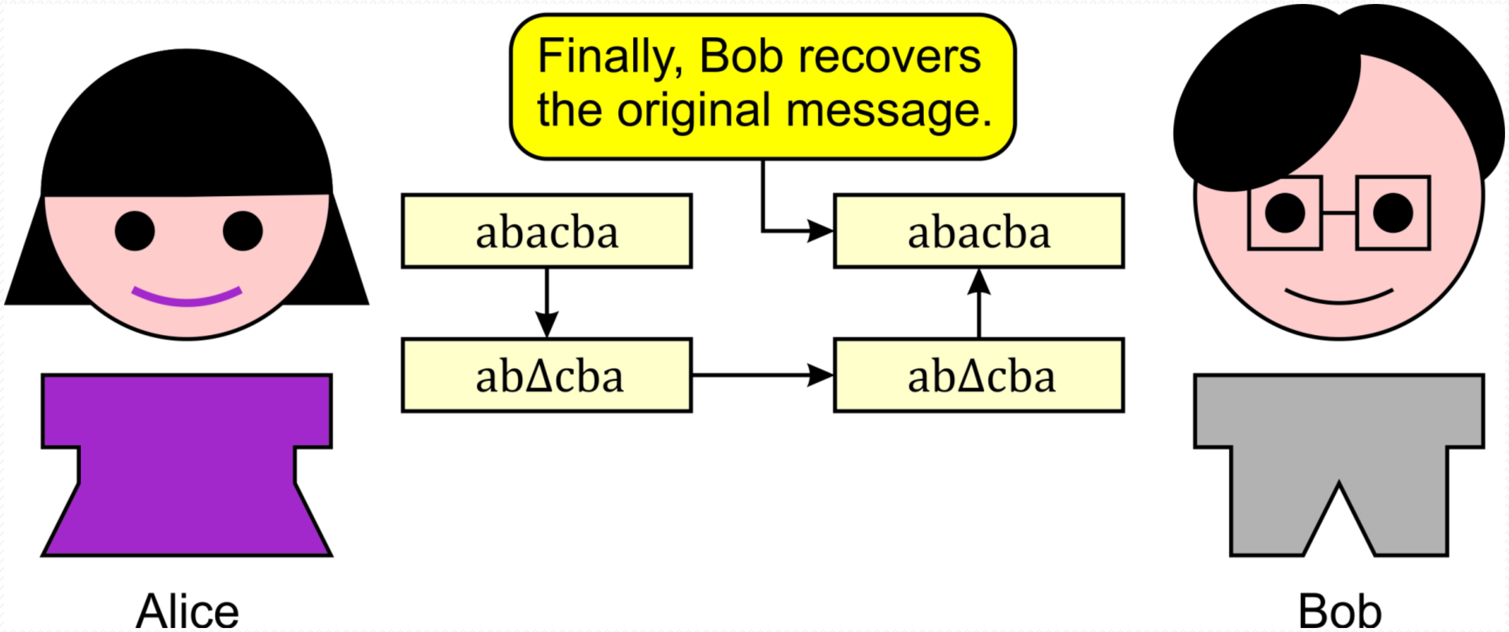
Alice and Bob



Alice and Bob



Alice and Bob



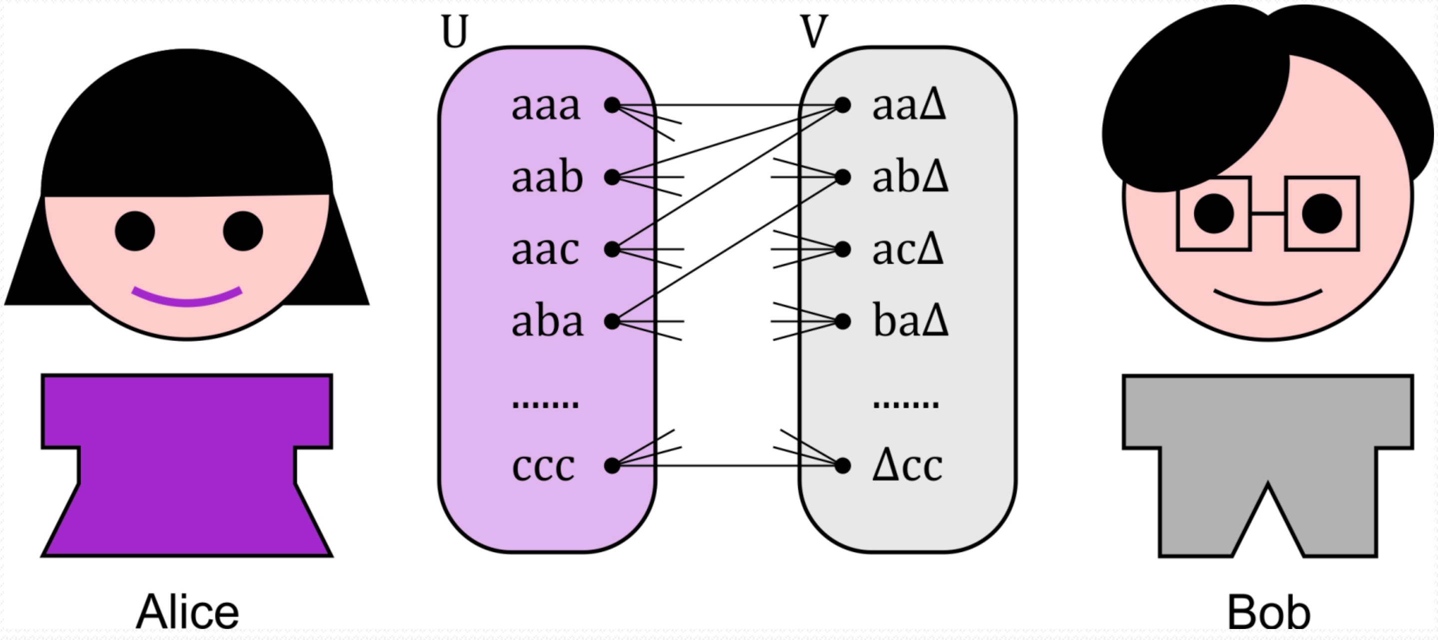
Protocol

- **We assume**:
  - **fixed alphabet Σ**,
  - **fixed length** of all **initial messages** given to Alice.

- **Is there any such protocol**?
- **Yes**. **Basic intuition**:
  - Alice **adds** information by choosing a position of Δ.
  - Alice **loses** information by deleting one letter.

Coding

- **Idea**: **Length of $|messages| = |\Sigma|$** .
- For $\Sigma = \{a, b, c\}$:



Example – 3-Letter Alphabet

- **Perfect matching** for $\Sigma = \{a, b, c\}$:

$aaa \leftrightarrow \Delta aa$	$baa \leftrightarrow b\Delta a$	$caa \leftrightarrow c\Delta a$
$aab \leftrightarrow \Delta ab$	$bab \leftrightarrow b\Delta b$	$cab \leftrightarrow ca\Delta$
$aac \leftrightarrow aa\Delta$	$bac \leftrightarrow ba\Delta$	$cac \leftrightarrow \Delta ac$
$aba \leftrightarrow \Delta ba$	$bba \leftrightarrow bb\Delta$	$cba \leftrightarrow cb\Delta$
$abb \leftrightarrow a\Delta b$	$bbb \leftrightarrow \Delta bb$	$cbb \leftrightarrow c\Delta b$
$abc \leftrightarrow ab\Delta$	$bbc \leftrightarrow \Delta bc$	$cbc \leftrightarrow c\Delta c$
$aca \leftrightarrow a\Delta a$	$bca \leftrightarrow \Delta ca$	$cca \leftrightarrow cc\Delta$
$acb \leftrightarrow ac\Delta$	$bcb \leftrightarrow bc\Delta$	$ccb \leftrightarrow \Delta cb$
$acc \leftrightarrow a\Delta c$	$bcc \leftrightarrow b\Delta c$	$ccc \leftrightarrow \Delta cc$

Coding – Encoding Example

- Consider word $\boldsymbol{w}$ over $\Sigma = \{a, b, c\}$:
  - $w = accbabccacaabbcabcbcacaa$.
- Let us **factorize** $\boldsymbol{w}$ into **groups of** $\boldsymbol{|\Sigma| = 3}$ letters:
  - $w = acc \,/\, bab \,/\, cca \,/\, caa \,/\, bbc \,/\, abc \,/\, bca \,/\, caa$.
- We want to **encode information** $\boldsymbol{i}$ into $\boldsymbol{w}$:
  - $i = 11001000$.

Coding – Encoding Example

- $i \quad = \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0$:
- $w \quad = \mathbf{acc} \,/\, \mathbf{bab} \,/\, cca \,/\, caa \,/\, \mathbf{bbc} \,/\, abc \,/\, bca \,/\, caa$,
- $w' \quad = \mathbf{a\Delta c} \,/\, \mathbf{b\Delta b} \,/\, cca \,/\, caa \,/\, \mathbf{\Delta bc} \,/\, abc \,/\, bca \,/\, caa$.

$aaa \leftrightarrow \Delta aa$	$baa \leftrightarrow b\Delta a$	$caa \leftrightarrow c\Delta a$
$aab \leftrightarrow \Delta ab$	$bab \leftrightarrow b\Delta b$	$cab \leftrightarrow ca\Delta$
$aac \leftrightarrow aa\Delta$	$bac \leftrightarrow ba\Delta$	$cac \leftrightarrow \Delta ac$
$aba \leftrightarrow \Delta ba$	$bba \leftrightarrow bb\Delta$	$cba \leftrightarrow cb\Delta$
$abb \leftrightarrow a\Delta b$	$bbb \leftrightarrow \Delta bb$	$cbb \leftrightarrow c\Delta b$
$abc \leftrightarrow ab\Delta$	$bbc \leftrightarrow \Delta bc$	$cbc \leftrightarrow c\Delta c$
$aca \leftrightarrow a\Delta a$	$bca \leftrightarrow \Delta ca$	$cca \leftrightarrow cc\Delta$
$acb \leftrightarrow ac\Delta$	$bcb \leftrightarrow bc\Delta$	$ccb \leftrightarrow \Delta cb$
$acc \leftrightarrow a\Delta c$	$bcc \leftrightarrow b\Delta c$	$ccc \leftrightarrow \Delta cc$

Coding – Major Drawback

- <u>**The major drawback**</u>:
- If ***w*** **does not start** with **left sentinel** ***¢*** then we **cannot factorize** ***w*** into **groups of** ***/Σ/*** **letters**.

- Word *w* can be factorized as:

1. *acc* / *bab* / *cca* / *caa* / *bbc* / *abc* / *bca* / *caa* ,
2. *ac* / *cba* / *bcc* / *aca* / *abb* / *cab* / *cbc* / *aca* / *a* ,
3. *a* / *ccb* / *abc* / *cac* / *aab* / *bca* / *bcb* / *cac* / *aa* .

Coding – Fixed Points

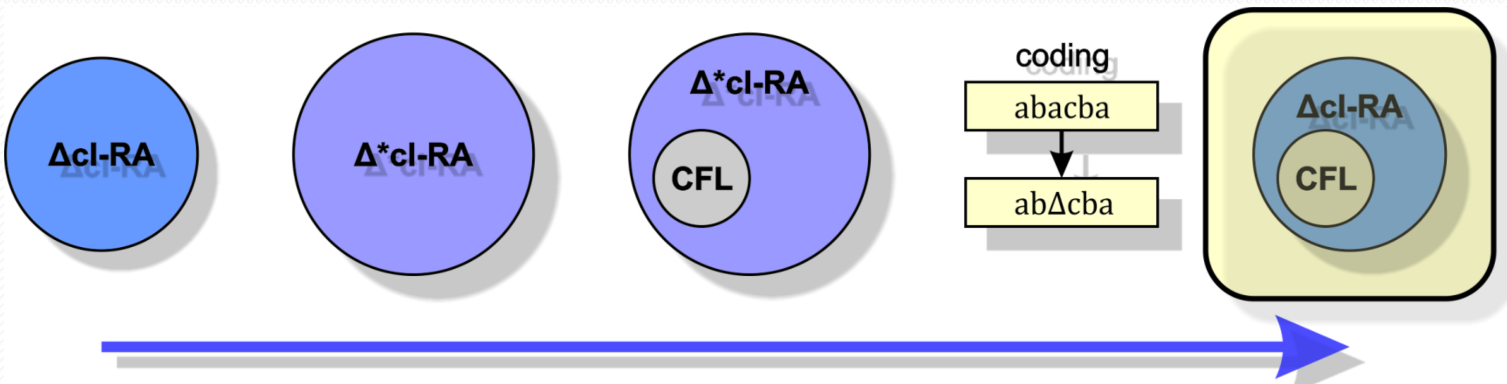
- **<u>Simple trick</u>**:
- To factorize w we need some "**fixed point**".
- The **left sentinel** $¢$ is **one example**.
- In the first phase we **distribute fixed points** throughout the whole input tape.

Coding – Fixed Points

- Suppose that we have:
  - $w = abacc\Delta accbabccacaabbcabcbcacaa$.
- The symbol Δ in w is our **fixed point**:
  - $w = abacc\Delta\,/\,acc\,/\,bab\,/\,cca\,/\,caa\,/\,bbc\,/\,abc\,/\,bca\,/\,caa$.
- Now we can place the **next fixed point**:
  - $w = abacc\Delta\,/\,acc\,/\,bab\,/\,cca\,/\,caa\,/\,bbc\,/\,abc\,/\,bca\,/\,c\Delta a$.

Organization

1. *Δ*-clearing restarting automata.
2. *Δ**-clearing restarting automata.
3. *Δ**-clearing restarting automata recognize *CFL*.
4. Special coding.
5. **<u>Reduction</u>**: *Δ**- to *Δ*-clearing restarting automata.



Algorithmic Viewpoint

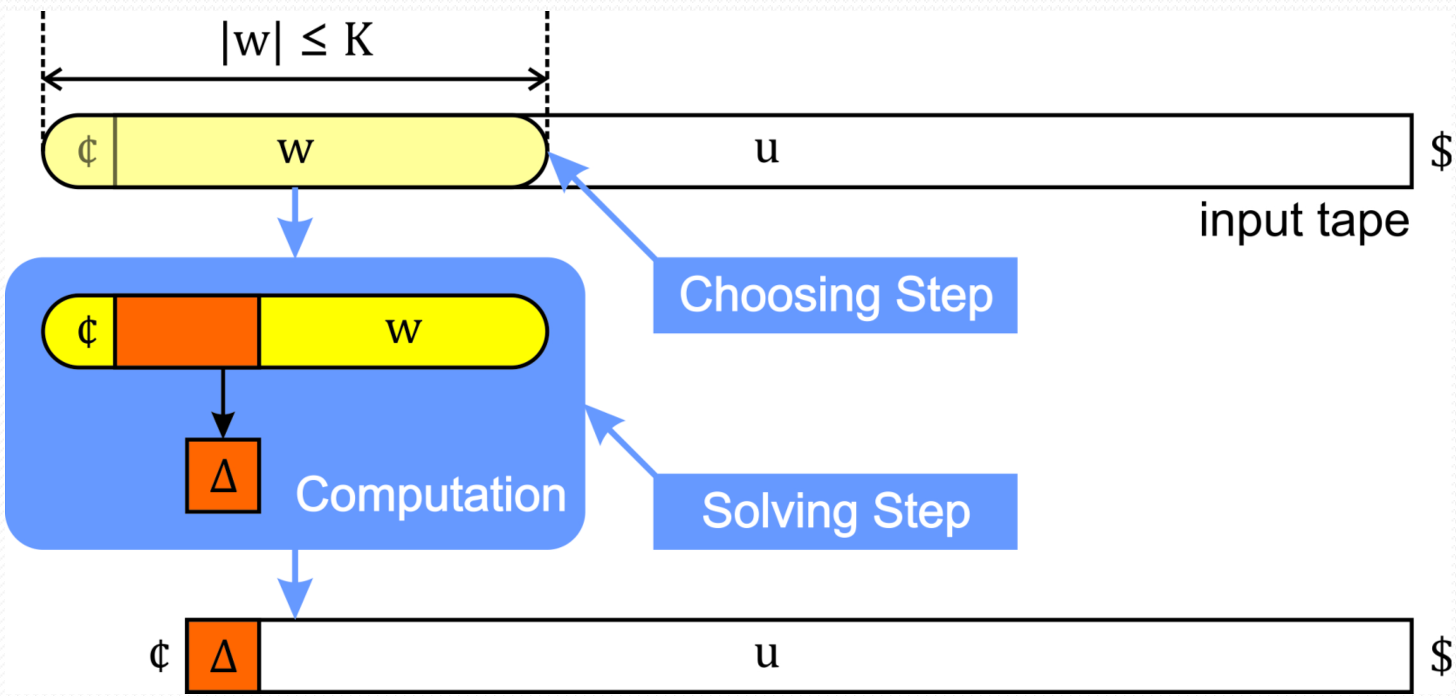
- Imagine a Δ-clearing restarting automaton as a nondeterministic machine N, which repeatedly executes the following **two steps**:

1. "**Choosing Step**": ***N* chooses** a subword w of the input $¢u\$$, $|w| \leq K$. (K is a fixed constant)
2. "**Solving Step**": ***N* runs a computation** on w, which either **rejects**, or **replaces** a subword of w by λ or Δ.

- N **accepts** u **iff** it can "**clear**" the whole word u.

Algorithmic Viewpoint

- **Illustration**:



Algorithmic Viewpoint

- To define **any** *Δ*-clearing restarting automaton:

1. Define the **solving algorithm** ***S***, called the **solver**.
2. Show the existence of a **suitable limit** ***K***.

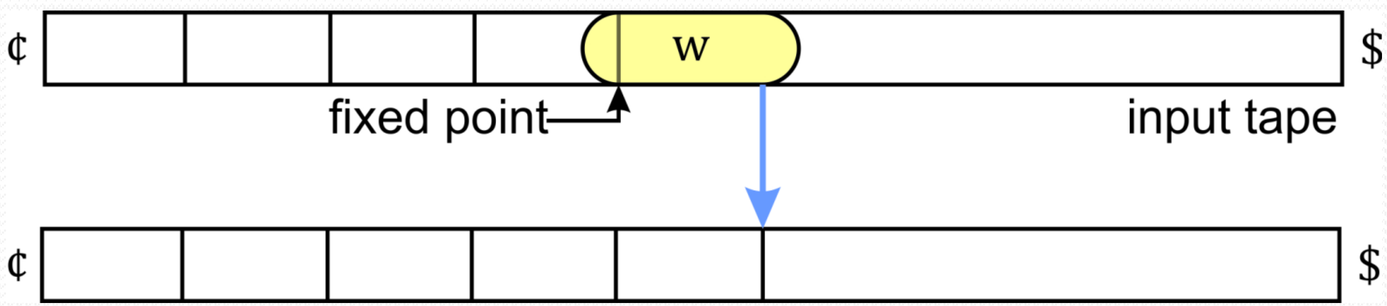
- We put **no resource limits** on the solving algorithm.

Idea of the Algorithm

- Consider *Δ*cl-RA M* whose [generalized] construction was based on a context-free grammar *G* in ChNF.
- We want the **solving algorithm *S* imitating *M***.
- We **do not preserve** the original representation of ***M***.

Idea of the Algorithm

- First, we **distribute fixed points** throughout the whole input tape in approximately equal distances:



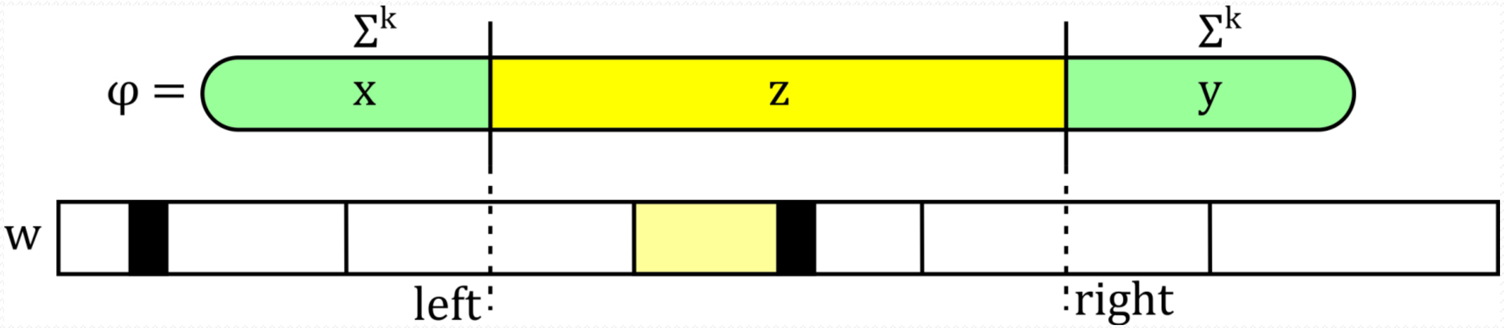
Idea of the Algorithm

- **Suppose** that *w* already **contains fixed points**.

1. We [internally] translate *Δ* symbols occurring in *w*.
2. We find an instruction $\varphi = (x, z \rightarrow \Delta^r, y)$ of the original ***Δ*cl-RA M*** **applicable inside *w***.
3. If there is no such instruction, **reject**.

Idea of the Algorithm

- **Suppose** $\varphi = (x, z \rightarrow \Delta^r, y)$ is **applicable inside** w.

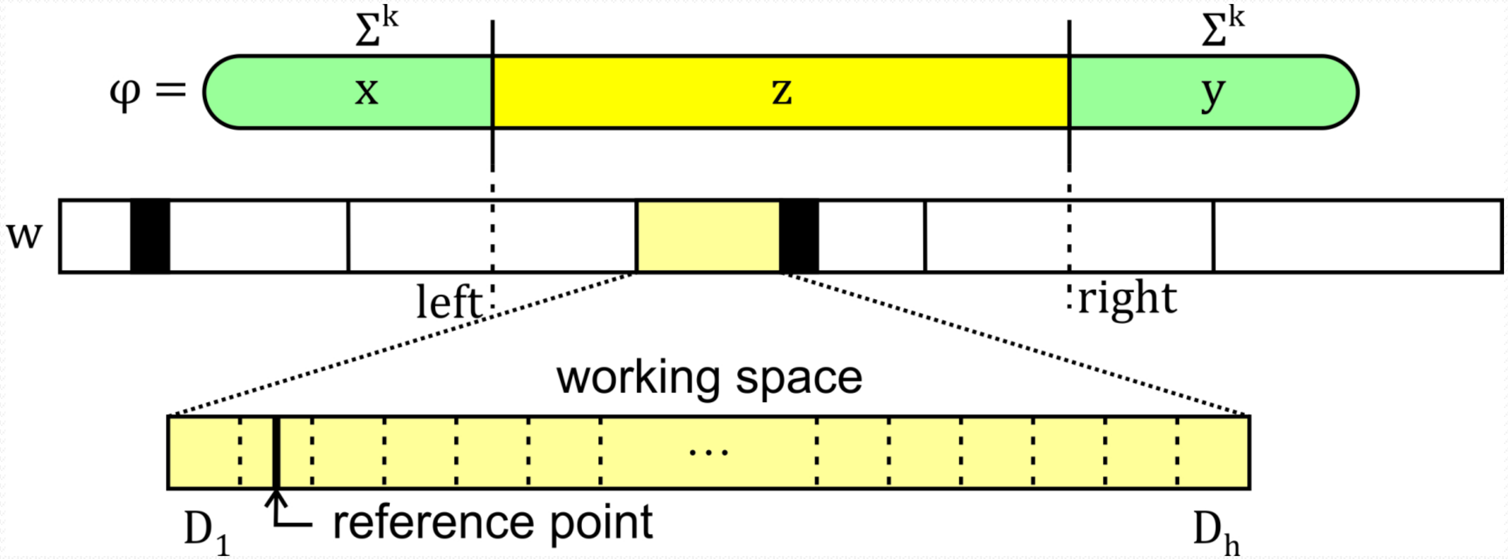


Idea of the Algorithm

- $\varphi = (x, z \to \Delta^r, y)$ is applicable inside w
- **Our goal**: **replace z by Δ^r**.
- To **avoid conflicts** we **encode information into z**.
- z contains a long enough **empty space $v \in \Sigma^*$**.
- v may be **interrupted** by **fixed points**.
- **Space** between fixed points is **long enough**.
- We **choose one such space** … **working space**.
- We **reserve this space** … **reference point Δ**.

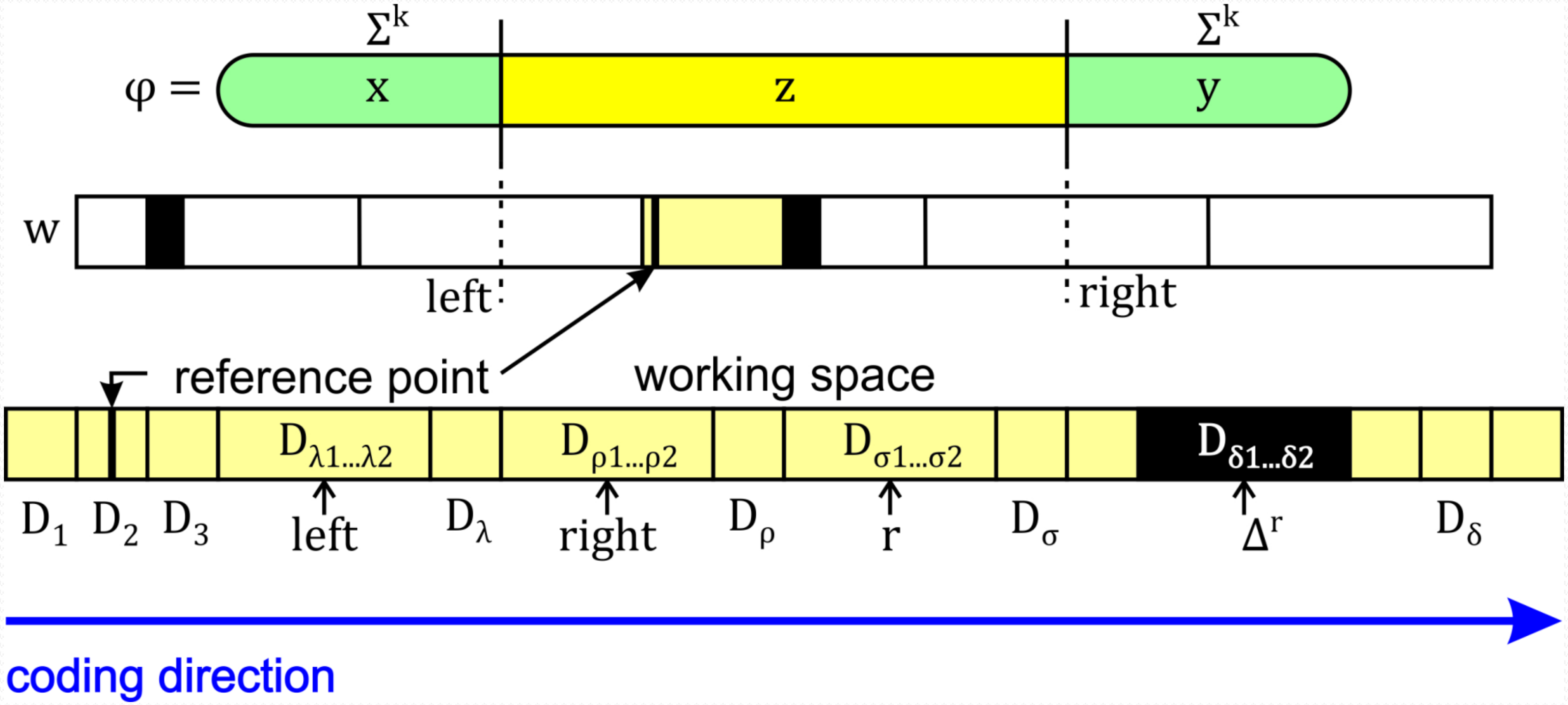
Idea of the Algorithm

- **Illustration**: $\varphi = (x, z \rightarrow \Delta^r, y)$



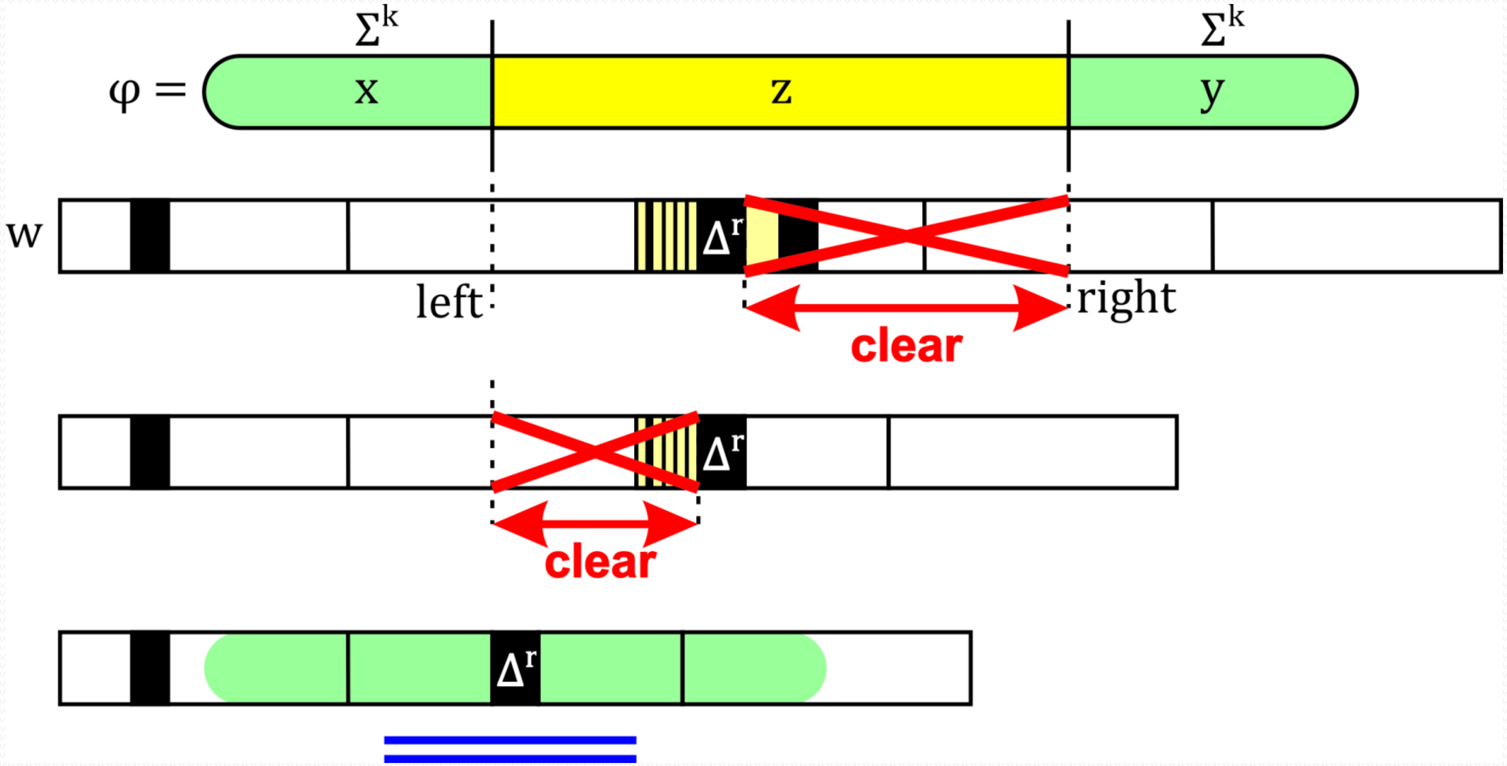
Idea of the Algorithm

- **<u>Working space</u>**: $\varphi = (x, z \rightarrow \Delta^r, y)$

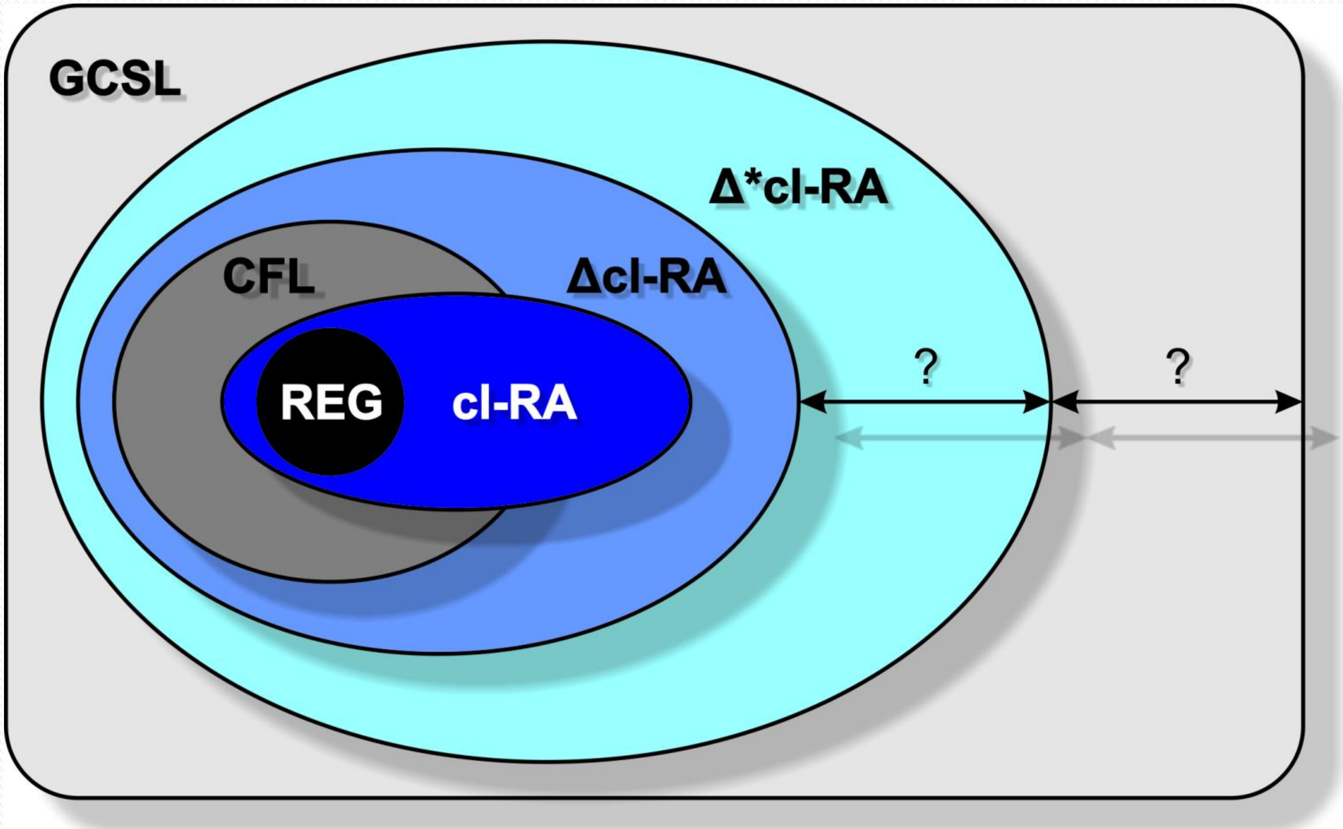


Idea of the Algorithm

- **Cleaning**: $\varphi = (x, z \to \Delta^r, y)$



Conclusion



References

- *Černo, P., Mráz, F.*: **Clearing restarting automata**. Fundamenta Informaticae 104(1), 17 – 54 (2010)
- *Černo, P., Mráz, F.*: **Delta-clearing restarting automata and CFL**. Tech. rep., Charles University, Faculty of Mathematics and Physics, Prague (2011)

  http://popelka.ms.mff.cuni.cz/cerno/structure_and_recognition/