MFCS 2021

The Simplest Non-Regular Deterministic Context-Free Language

Jiří Šíma

sima@cs.cas.cz

Institute of Computer Science
Czech Academy of Sciences, Prague, Czechia

joint work with **Petr Jančar**

pj.jancar@gmail.com

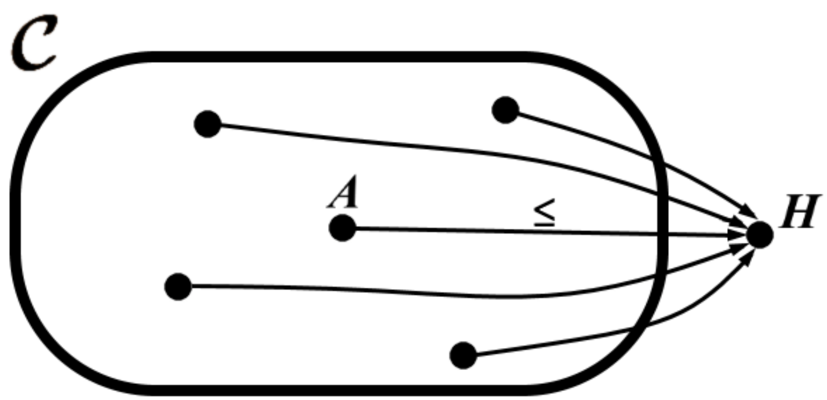
Department of Computer Science
Palacký University Olomouc, Czechia

$\mathcal{C}$-Hard Problems

$\mathcal{C}$ is a complexity class of decision problems (i.e. formal languages)

$A \leq B$ is a reduction transforming a problem A to a problem B (a preorder), which is assumed not to have a higher computational complexity than $\mathcal{C}$

H is a $\mathcal{C}$-hard problem (under the reduction $\leq$) if for every $A \in \mathcal{C}$, $A \leq H$



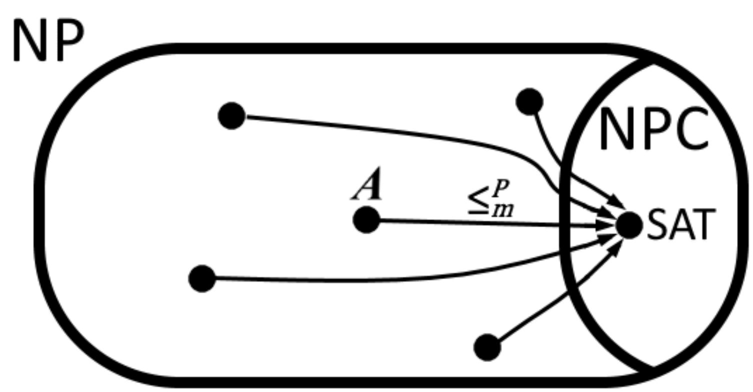
- If a $\mathcal{C}$-hard problem has a (computationally) "easy" solution, then each problem in $\mathcal{C}$ has an "easy" solution (via the reduction).
- If a $\mathcal{C}$-hard problem H is in $\mathcal{C}$ (a so-called $\mathcal{C}$-complete problem), then H belongs to the hardest problems in the class $\mathcal{C}$.

The Most Prominent Example: NP-Hard Problems

$\mathcal{C} =$ NP is the class of decision problems solvable in polynomial time by a nondeterministic Turing machine

$A \leq_m^P B$ is a polynomial-time many-one reduction (Karp reduction) from A to B

the satisfiability problem SAT is NP-hard: for every $A \in$ NP, $A \leq_m^P$ SAT

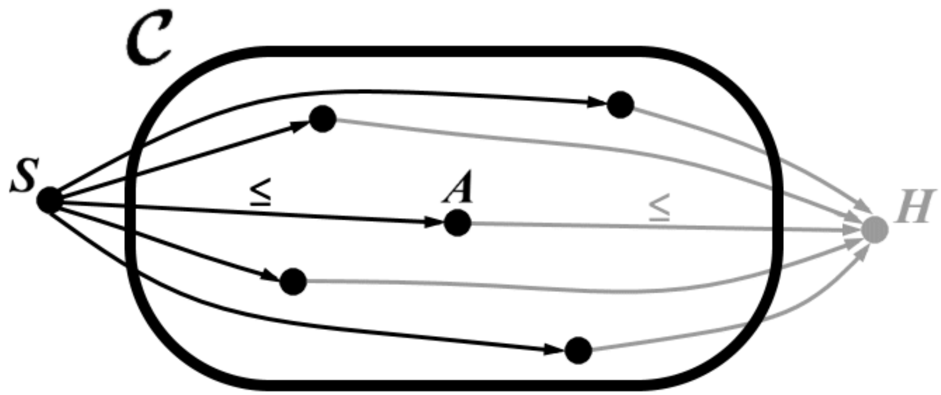


- If an NP-hard problem is polynomial-time solvable,
  then each NP problem would be solved in polynomial time.
- The NP-hard problem SAT is in NP (i.e. SAT is NP-complete),
  that is, SAT belongs to the hardest problems (NPC) in the class NP.

$\mathcal{C}$-Simple Problems

a conceptual counterpart to $\mathcal{C}$-hard problems:

S is a $\mathcal{C}$-simple problem (under the reduction $\leq$) if for every $A \in \mathcal{C}$, $S \leq A$



- If a $\mathcal{C}$-simple problem S proves to be not "easy",
  e.g. S is not solvable by a machine M (M can compute the reduction $\leq$),
  then all problems in $\mathcal{C}$ are not "easy", i.e. $\mathcal{C}$ cannot be solved by M.

  $\longrightarrow$ **a new proof technique:** a lower bound known for one $\mathcal{C}$-simple problem S extends to the whole class of problems $\mathcal{C}$

- If a $\mathcal{C}$-simple problem S is in $\mathcal{C}$, then S is the simplest problem in the class $\mathcal{C}$.

A Trivial Example: SAT is simple for the class of NP-hard problems under $\leq_m^P$

A Nontrivial Example of a $\mathcal{C}$-Simple Problem

$\mathcal{C}$ = DCFL' = DCFL \ REG
is the class of non-regular deterministic context-free languages

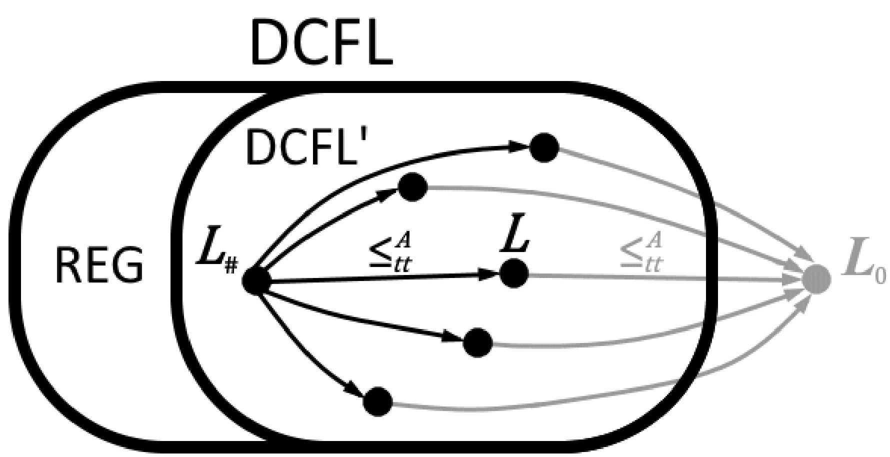
$L_1 \leq_{tt}^{A} L_2$ is a truth-table reduction (a stronger Turing reduction) from L_1 to L_2
implemented by a Mealy machine with the oracle L_2

The Main Result:

the language $L_{\#} = \{0^n 1^n \mid n \geq 1\}$ over the binary alphabet $\{0, 1\}$ is
DCFL'-simple under the reduction $\leq_{tt}^{A}$: for every $L \in$ DCFL', $L_{\#} \leq_{tt}^{A} L$

$\longrightarrow$ $L_{\#} \in$ DCFL' is the *simplest* non-regular deterministic context-free languages

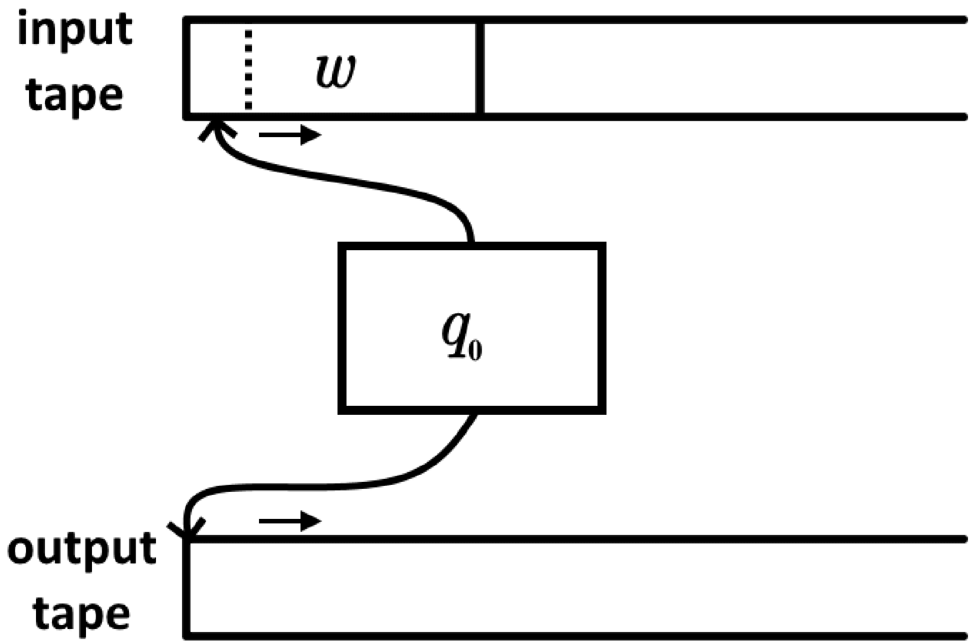
cf. the *hardest* context-free language L_0 due to `S. Greibach (1973)` is CFL-hard



Mealy Machines

$\mathcal{A}$ is a Mealy Machine with an input/output alphabet Σ/Δ

i.e. a deterministic finite automaton with an output tape:

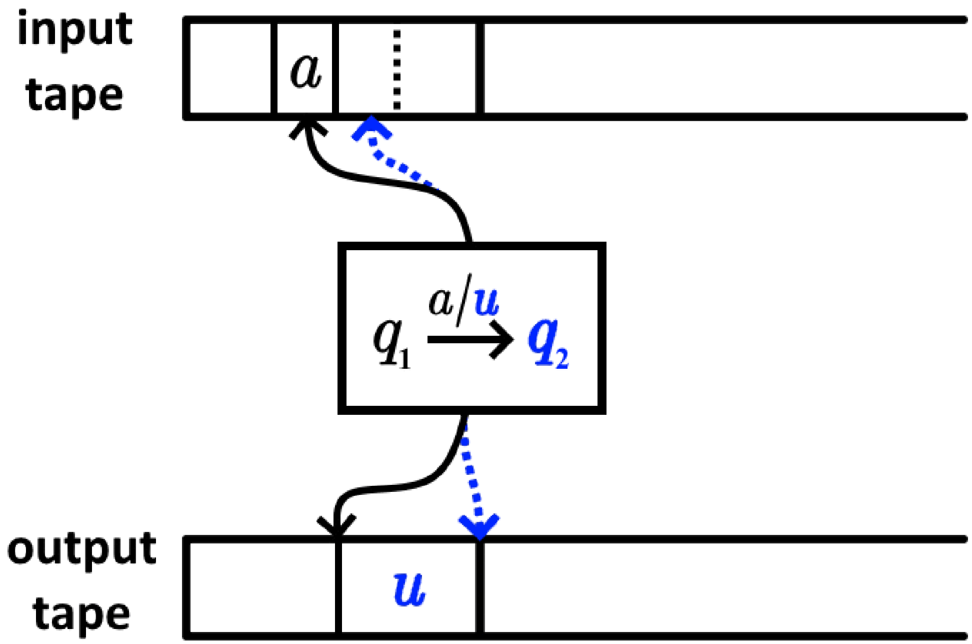


input word $w \in \Sigma^*$

initial state q_0

Mealy Machines

$\mathcal{A}$ is a Mealy Machine with an input/output alphabet Σ/Δ
i.e. a deterministic finite automaton with an output tape:



current input symbol $a \in \Sigma$

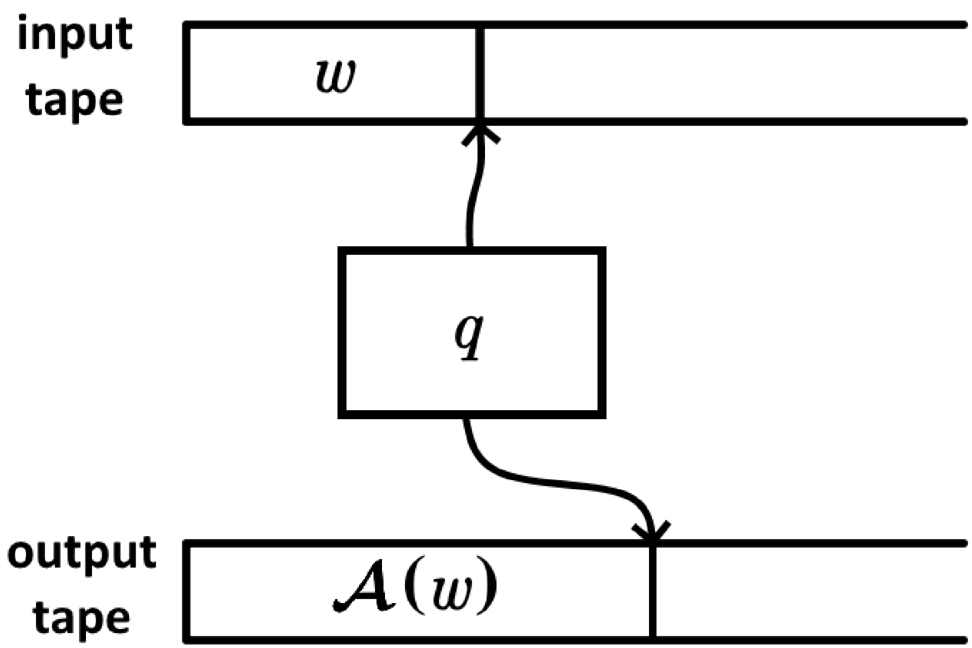
state transition
from q_1 to q_2

output string $u \in \Delta^*$

Mealy Machines

$\mathcal{A}$ is a Mealy Machine with an input/output alphabet Σ/Δ

i.e. a deterministic finite automaton with an output tape:



input $w \in \Sigma^*$

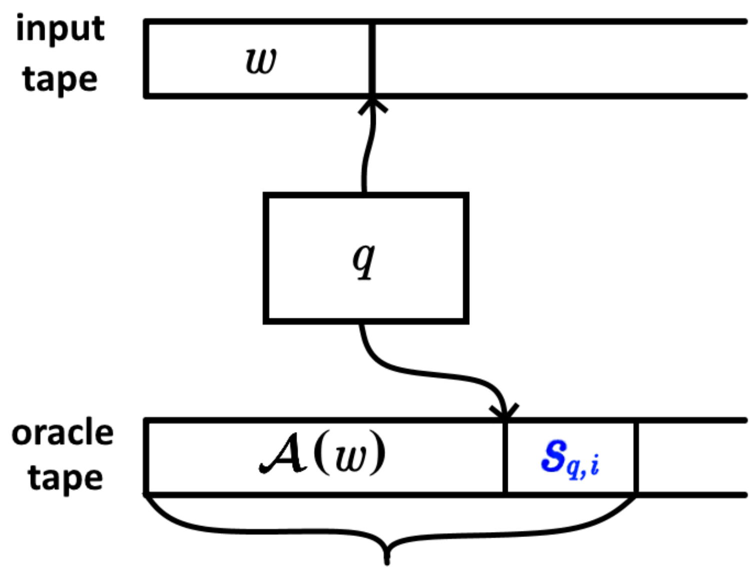
"final" state q

output $\mathcal{A}(w) \in \Delta^*$

$\longrightarrow$ a deterministic finite-state transducer: $w \in \Sigma^* \longmapsto \mathcal{A}(w) \in \Delta^*$

The Truth-Table Reduction by Oracle Mealy Machines

$\mathcal{A}^{L_2}$ is a Mealy Machine $\mathcal{A}$ with an oracle $L_2 \subseteq \Delta^*$:



for each state q of $\mathcal{A}$:

- r_q suffixes $s_{q,1}, \ldots, s_{q,r_q} \in \Delta^*$
- truth table $T_q : \{0,1\}^{r_q} \to \{0,1\}$ with r_q variables

r_q queries: $\overset{?}{\in} L_2$ for every $i = 1, \ldots, r_q$

$w \in \Sigma^*$ is accepted by $\mathcal{A}^{L_2}$ **iff** w brings $\mathcal{A}$ to the state q such that

$$T_q\left(\mathcal{A}(w) \cdot s_{q,1} \overset{?}{\in} L_2\,,\ \mathcal{A}(w) \cdot s_{q,2} \overset{?}{\in} L_2\,, \ldots,\ \mathcal{A}(w) \cdot s_{q,r_q} \overset{?}{\in} L_2\right) = 1$$

$L_1 \leq_{tt}^{A} L_2$: $L_1 \subseteq \Sigma^*$ is truth-table reducible to $L_2 \subseteq \Delta^*$ **iff**
$L_1 = \mathcal{L}(\mathcal{A}^{L_2})$ is accepted by some Mealy machine $\mathcal{A}^{L_2}$ with oracle L_2

Proposition: *The relation $\leq_{tt}^{A}$ is a preorder.*

Why $L_{\#} = \{0^n 1^n \mid n \geq 1\}$ Is the Simplest DCFL' language?

any reduced context-free grammar G generating a non-regular language $L \subseteq \Delta^*$

is self-embedding: there is a self-embedding nonterminal A admitting the derivation

$A \Rightarrow^* xAy$ for some non-empty strings $x, y \in \Delta^+$ (Chomsky, 1959)

G is reduced $\longrightarrow$ $S \Rightarrow^* vAz$ and $A \Rightarrow^* w$ for some $v, w, z \in \Delta^*$

$$\longrightarrow \qquad S \Rightarrow^* vx^m w y^m z \in L \text{ for every } m \geq 0 \qquad (1)$$

??? a conceivable (one-one) reduction from $L_{\#}$ to L: for every $m, n \geq 1$,

$$0^m 1^n \in \{0,1\}^* \longmapsto vx^m w y^n z \in \Delta^*$$

(the inputs outside 0^+1^+ are mapped onto some fixed string outside L)

since $0^m 1^n \in L_{\#}$ **implies** $vx^m w y^n z \in L$ by (1)

!!! however, the opposite implication may not be true:

Why $L_{\#}$ Is the Simplest DCFL' language? (cont.)

!!! however, the opposite implication may not be true:

for the DCFL' language $L_1 = \{a^m b^n \mid 1 \leq m \leq n\}$ over $\Delta = \{a, b\}$

there are **no** words $v, x, w, y, z \in \Delta^*$ such that for every $m, n \geq 1$,

$$vx^m wy^n z \in L_1 \text{ would ensure } m = n$$

nevertheless, already **two** inputs $a^m b^{n-1} \overset{?}{\in} L_1$ and $a^m b^n \overset{?}{\in} L_1$ decides $m \overset{?}{=} n$

$\longrightarrow$ the truth-table reduction from $L_{\#}$ to L_1 with two queries to the oracle L_1:

$$0^m 1^n \in \{0,1\}^* \quad \longmapsto \quad vx^m wy^{n-1} z \in \Delta^*, \quad vx^m wy^n z \in \Delta^*$$

where $x = a$, $y = b$, $v = w = z = \varepsilon$ is the empty string

satisfying $0^m 1^n \in L_{\#}$ **iff** $(vx^m wy^{n-1} z \notin L_1$ **and** $vx^m wy^n z \in L_1)$

this can be generalized to any DCFL' language L:

The Main Technical Result

Theorem: *Let $L \subseteq \Delta^*$ be a non-regular deterministic context-free language over an alphabet Δ. There exist non-empty words $v, x, w, y, z \in \Delta^+$ and a language $L' \in \{L, \overline{L}\}$ (where $\overline{L} = \Delta^* \setminus L$ is the complement of L) such that*

1. **either** for all $m, n \geq 0$, $vx^m wy^n z \in L'$ ***iff*** $m = n$,
2. **or** for all $m, n \geq 0$, $vx^m wy^n z \in L'$ ***iff*** $m \leq n$.

1.

$m \backslash n$	0	1	2	3	$\cdots$
0	$\in L'$	$\notin L'$	$\notin L'$	$\notin L'$	
1	$\notin L'$	$\in L'$	$\notin L'$	$\notin L'$	
2	$\notin L'$	$\notin L'$	$\in L'$	$\notin L'$	
3	$\notin L'$	$\notin L'$	$\notin L'$	$\in L'$	
$\vdots$					$\ddots$

2.

$m \backslash n$	0	1	2	3	$\cdots$
0	$\in L'$	$\in L'$	$\in L'$	$\in L'$	
1	$\notin L'$	$\in L'$	$\in L'$	$\in L'$	
2	$\notin L'$	$\notin L'$	$\in L'$	$\in L'$	
3	$\notin L'$	$\notin L'$	$\notin L'$	$\in L'$	
$\vdots$					$\ddots$

In particular, for all $m \geq 0$ and $n > 0$,

$$(vx^m wy^{n-1} z \notin L' \ \textbf{and}\ vx^m wy^n z \in L') \ \textbf{iff}\ m = n.$$

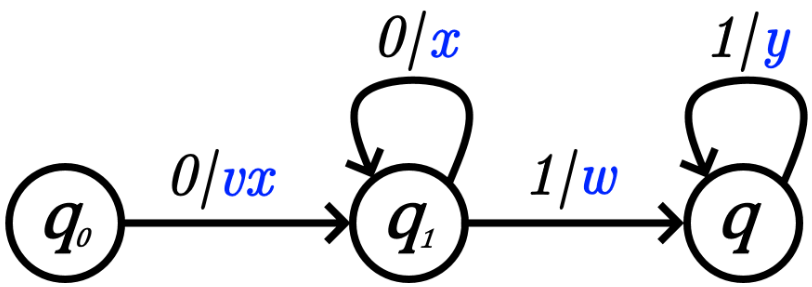
The Truth-Table Reduction From $L_{\#}$ to Any DCFL' L

implemented by a **Mealy machine** $\mathcal{A}^L$ with two queries to the oracle L :

For any DCFL' language $L \subseteq \Delta^*$, Theorem provides $v, x, w, y, z \in \Delta^+$ and $L' \in \{L, \overline{L}\}$, say $L' = L$ (analogously for $L' = \overline{L}$), such that

$$(\, vx^m wy^{n-1}z \notin L \;\; \textbf{and} \;\; vx^m wy^n z \in L \,) \;\; \textbf{iff} \;\; m = n\,. \qquad (2)$$

$\mathcal{A}^L$ transforms the input $0^m 1^n$ to the output $\mathcal{A}(0^m 1^n) = vx^m wy^{n-1} \in \Delta^+$ (the inputs outside 0^+1^+ are rejected), while moving to the state q with $r_q = 2$ suffixes $s_{q,1}$, $s_{q,2}$ and the truth table $T_q : \{0,1\}^2 \longrightarrow \{0,1\}$



$s_{q,1} = z$

$s_{q,2} = yz$

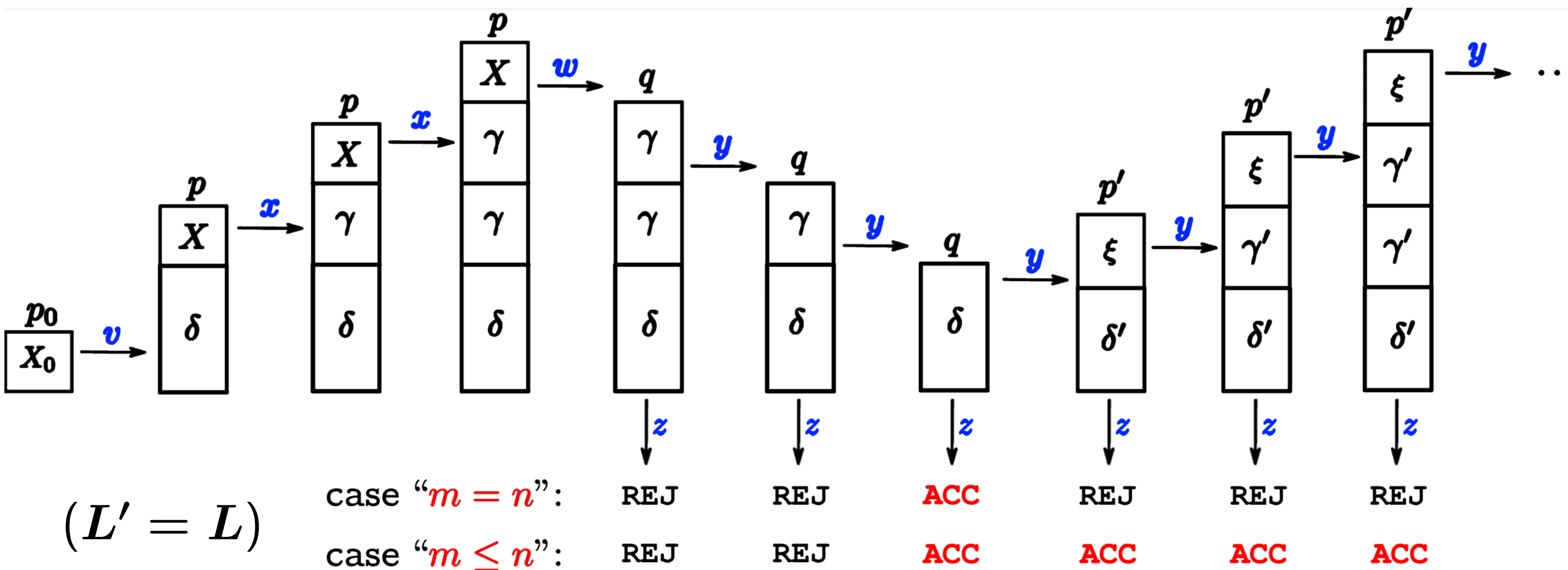
$\mathcal{A}(0^m1^n) \cdot z = vx^m wy^{n-1}z \overset{?}{\in} L$	$\mathcal{A}(0^m1^n) \cdot yz = vx^m wy^n z \overset{?}{\in} L$	T_q
0	1	1
0	0	0
1	1	0
1	0	0

It follows from (2) that $\mathcal{L}(\mathcal{A}^L) = L_{\#}$, i.e. $L_{\#} \leq_{tt}^{A} L$.

Ideas of the Proof of the Theorem

(inspired by some ideas on regularity of pushdown processes due to `Jančar`, 2020)

- any non-regular DCFL language $L \subseteq \Delta^*$ is accepted
  by a deterministic pushdown automaton $\mathcal{M}$ by the empty stack
- since $L \notin$ REG, there is a computation by $\mathcal{M}$, reaching configurations with an arbitrary large stack which is being erased afterwards, corresponding to $v, x, w, y, z \in \Delta^+$ such that $vx^mwy^mz \in L$ for all $m \geq 1$
- in addition, we aim to ensure that for all $m \geq 0$ and $n > 0$,
  $(\, vx^mwy^{n-1}z \notin L' \text{ and } vx^mwy^nz \in L' \,)$ **iff** $m = n$



Ideas of the Proof of the Theorem

(inspired by some ideas on regularity of pushdown processes due to Jančar, 2020)

- any non-regular DCFL language $L \subseteq \Delta^*$ is accepted
  by a deterministic pushdown automaton $\mathcal{M}$ by the empty stack

- since $L \notin$ REG, there is a computation by $\mathcal{M}$, reaching configurations with an arbitrary large stack which is being erased afterwards, corresponding to $v, x, w, y, z \in \Delta^+$ such that $vx^m wy^m z \in L$ for all $m \geq 1$

- in addition, we aim to ensure that for all $m \geq 0$ and $n > 0$,
  $(\, vx^m wy^{n-1} z \notin L'$ **and** $vx^m wy^n z \in L' \,)$ **iff** $m = n$

- we study the computation of $\mathcal{M}$ on an infinite word that traverses infinitely many pairwise non-equivalent configurations

- we use a natural congruence property of language equivalence on the set of configurations (determinism of $\mathcal{M}$ is essential)

- we apply Ramsey's theorem for extracting the required $v, x, w, y, z \in \Delta^+$ from the infinite computation

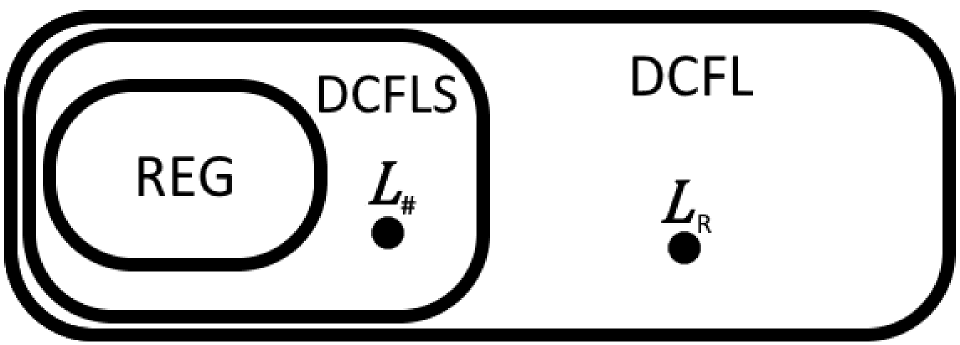
Basic Properties of DCFL'-Simple Problems

DCFLS is the class of DCFL'-simple problems

Proposition:

- REG $\subsetneq$ DCFLS $\subsetneq$ DCFL ,

  e.g. $L_{\#} \in$ DCFLS, $L_R = \{wcw^R \mid w \in \{a,b\}^*\} \notin$ DCFLS



- The class DCFLS is closed under complement and intersection with regular languages.

- The class DCFLS is not closed under concatenation, intersection, and union.

An Application of DCFL'-Simple $L_{\#}$ in Neural Networks

(this application has originally inspired the concept of a DCFL'-simple problem)

The Computational Power of Neural Networks (NNs)

(discrete-time recurrent NNs with the saturated-linear activation function)

depends on the information contents of weight parameters:

- **integer** weights: **finite automaton (FA)** (Minsky, 1967)
- **rational** weights: **Turing machine (TM)** (Siegelmann, Sontag, 1995)
  polynomial time $\equiv$ the complexity class P
- arbitrary **real** weights: **"super-Turing"** computation (Siegelmann, Sontag, 1994)
  polynomial time $\equiv$ the nonuniform complexity class P/poly
  exponential time $\equiv$ any I/O mapping
- increasing **Kolmogorov complexity** of real weights
  polynomial time $\equiv$ a proper **hierarchy** of nonuniform complexity classes **between P and P/poly** (Balcázar, Gavaldà, Siegelmann, 1997)

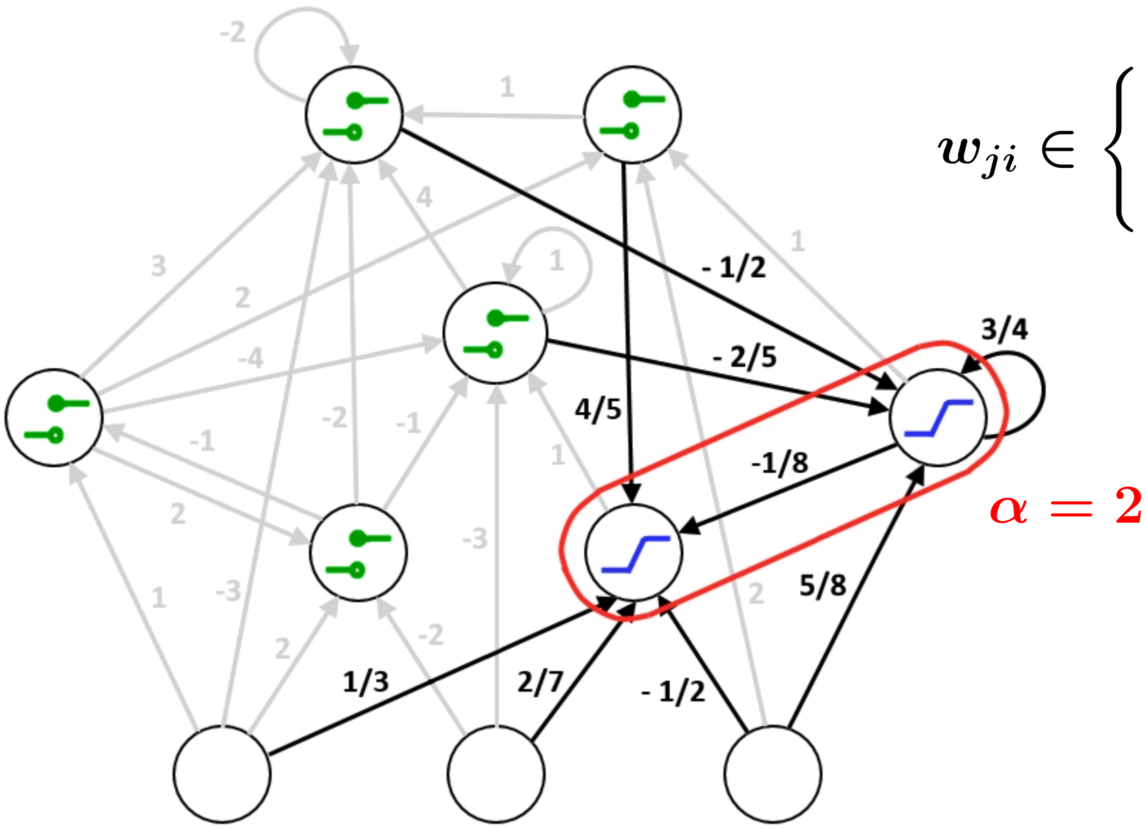
??? the gap in the analysis between realistic **integer** and **rational** weights w.r.t. **Chomsky hierarchy**: **regular vs. recursively enumerable** languages

A Neural Network Model with Increasing Analogicity

from **integer** to **rational** weights

α**ANN** $=$ a **binary-state** NN with **integer** weights

$+$ α **extra** **analog-state** neurons with **rational** weights



$$w_{ji} \in \begin{cases} \mathbb{Q} & j = 1, \ldots, \alpha \\ \mathbb{Z} & j = \alpha + 1, \ldots, s \end{cases}$$

$$i \in \{0, \ldots, s\}$$

A Neural Network Model with Increasing Analogicity

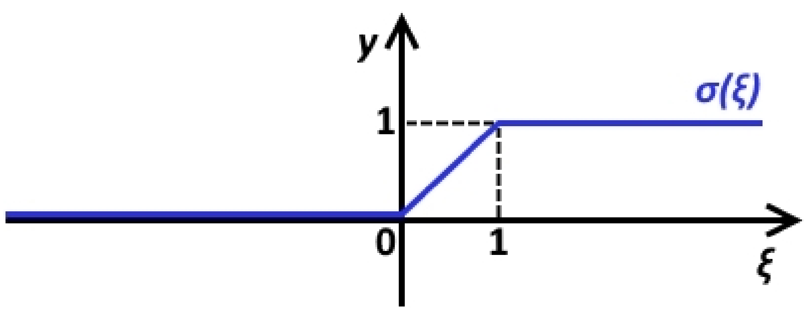
from **binary** $(\{0,1\})$ to **analog** $([0,1])$ states of neurons

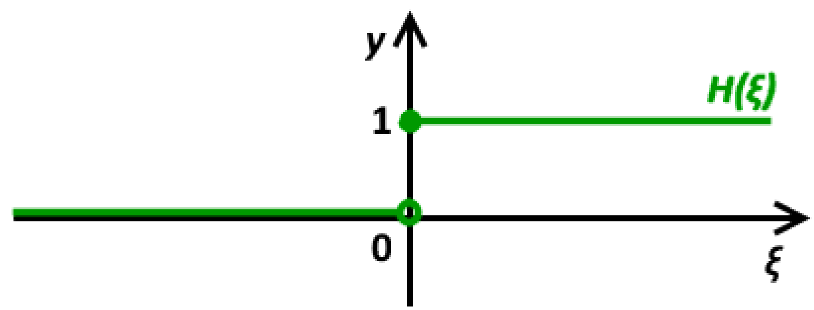
α**ANN** $=$ a **binary-state** NN with **integer** weights

$+\ \alpha$ **extra analog-state** neurons with **rational** weights

$$y_j^{(t+1)} = \sigma_j\left(\sum_{i=0}^{s} w_{ji} y_i^{(t)}\right) \qquad j = 1,\ldots,s \qquad \text{updating the states of neurons}$$

$$\sigma_j(\xi) = \begin{cases} \sigma(\xi) = \begin{cases} 1 & \text{for } \xi \geq 1 \\ \xi & \text{for } 0 < \xi < 1 \\ 0 & \text{for } \xi \leq 0 \end{cases} & j = 1,\ldots,\alpha \quad \text{saturated-linear function} \\ \\ H(\xi) = \begin{cases} 1 & \text{for } \xi \geq 0 \\ 0 & \text{for } \xi < 0 \end{cases} & j = \alpha+1,\ldots,s \quad \text{Heaviside function} \end{cases}$$





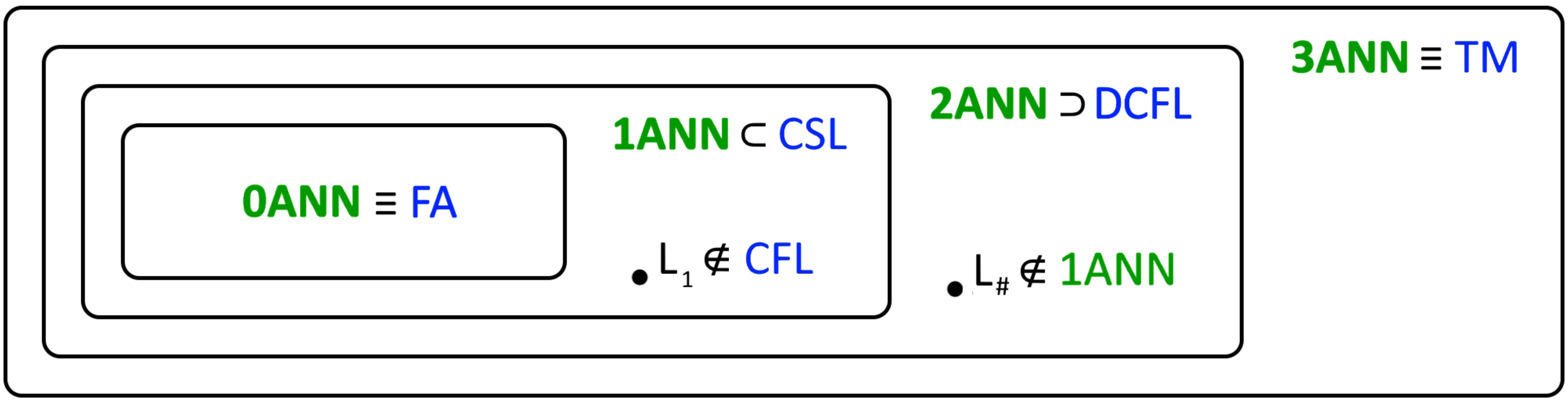
The Analog Neuron Hierarchy (Šíma, 2019, 2020)

the computational power of NNs increases with the number α of extra analog neurons:

$$\text{FA} \equiv \text{0ANN} \subseteq \text{1ANN} \subseteq \text{2ANN} \subseteq \text{3ANN} \subseteq \ldots \equiv \text{TM}$$

$\uparrow$ $\times$ $\uparrow$

integer weights **Chomsky hierarchy** **rational** weights

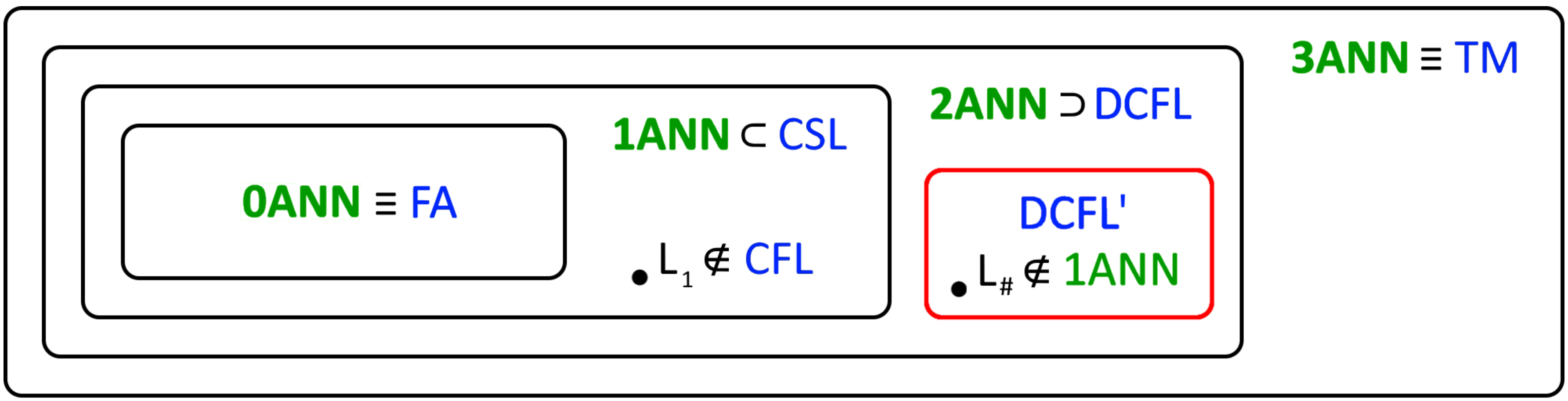


- $L_\# = \{0^n 1^n \mid n \geq 1\} \notin$ 1ANN $\subset$ CSL (Context-Sensitive Languages)
- $L_1 = \left\{ x_1 \ldots x_n \in \{0,1\}^* \,\middle|\, \sum_{k=1}^{n} x_{n-k+1} \left(\frac{3}{2}\right)^{-k} < 1 \right\} \in$ 1ANN $\setminus$ CFL
- DCFL $\subset$ 2ANN
- 3ANN $\equiv$ TM

The Technique of Expanding a Lower Bound

- $L_\# \notin$ 1ANN with a nontrivial proof (based on the Bolzano–Weierstrass theorem) which can hardly be generalized to another DCFL' language
- $L_\#$ is DCFL'-simple under $\leq_{tt}^A$
- the reduction $\leq_{tt}^A$ to any $L \in$ 1ANN can be implemented by 1ANN

$\longrightarrow$ the known lower bound $L_\# \notin$ 1ANN for a single DCFL'-simple problem $L_\#$ is extended to the whole class DCFL' $\cap$ 1ANN $= \emptyset$



Comments:

- If any DCFL' language proves to be CFL'-simple, then CFL' $\cap$ 1ANN $= \emptyset$.
- $L_\#$ is not CSL'-simple since $L_\# \leq_{tt}^A L_1 \in$ 1ANN would imply $L_\# \in$ 1ANN

A Summary

- We have introduced a new notion of $\mathcal{C}$-simple problems which is a conceptual counterpart to $\mathcal{C}$-hard problems.

- We have shown $L_{\#} = \{0^n1^n \mid n \geq 1\}$ to be a DCFL'-simple problem under the truth-table reduction by oracle Mealy machines:

  $\longrightarrow$ $L_{\#}$ is the simplest DCFL' problem

- We have proposed a new proof technique of expanding a lower bound known for a single $\mathcal{C}$-simple problem to the whole class of problems $\mathcal{C}$, which has been illustrated by a nontrivial application to the analysis of neural networks:

  $$\text{DCFL'-simple } L_{\#} \notin \text{1ANN} \quad \longrightarrow \quad \text{DCFL'} \cap \text{1ANN} = \emptyset$$

Open Problems

- Is $L_{\#}$ CFL'-simple or UCFL'-simple (Unambiguous CFL') **?**

  $$(\longrightarrow \ \text{(U)CFL'} \cap \text{1ANN} \stackrel{?}{=} \emptyset)$$

- Examples of nontrivial $\mathcal{C}$-simple problems for other complexity classes $\mathcal{C}$ under suitable reductions **?**